TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

ABBREVIATIONS USED

XML	Extensible Markup Language
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
RMI	Remote Method Invocation
URI	Uniform Resource Identifier
RPC	Remote Procedure Call
HTTP	Hypertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol

CHAPTER 1
INTRODUCTION

1.1 Background

The Internet is a heterogeneous environment that consists of different network devices, operating systems, network technologies and software systems. The heterogeneity in such a system is inevitable and cannot be eliminated. It does, however, pose interoperability issues when software systems are integrated on the Internet [73].

Web Services propose a framework that allows integration of heterogeneous software systems on the Internet. The integration is made possible by the standardisation of interfaces and interactions of software components by means of specifications that are based on open XML standards and Internet protocols [27][37][70][73]. The framework also provides a facility to publish the software components as services on the Internet so that users intending to use the software components can locate and invoke them over the Internet [36][66]. The core specifications of the framework are SOAP, WSDL and UDDI [66][45].

The standardisation of interfaces and interactions could enable interoperability among heterogeneous software components, which will result in different application scenarios that utilise the software integration. By using the Web Services framework, organisations or individuals can therefore create and expose software components as specific services to other organisations. By reducing interoperability issues, this framework aims to facilitate the creation of applications from the available software components on the Internet [25]. This would result in building complex applications by combining multiple software components of different functionalities across organisational boundaries.

Reusing available components to create new applications is an advantage in terms of ease of creation, development and operational costs. On the other hand, integrating components developed by different organisations - and implemented and maintained in different network domains - will create issues that need to be considered and studied. Some of the problems arise due to the difference in the non-functional

attributes of the different components such as their reliability, quality, security, performance, scalability and reusability [27][66][73].

The success of an application developed from different components is based on the functioning of its constituent components. For example, reliability of such an application is dependent on the reliability of its constituent components. The constituent components of the application need not be under the control of the same organisation or local network domain [70]. This may introduce uncertainties about the properties and factors that influence the behaviour of such a constituent entity. As the number of components in an application increases, issues related to for instance security, reliability, quality and performance of the application get complicated. For an entity who intends to use such an application, evaluating *trustworthiness* and establishing trust with a service in any context (including reliability, security, scalability, dependability and quality of service) become relevant.

Trust

This dissertation deals with the technical aspects of trust. Trust is a complex concept and has been studied and presented from different perspectives in the literature [47][40][2][8][6]. Different perspectives about trust have also led to different definitions of trust. Many of the studies of trust focus on distributed systems and e-commerce transactions, with specific emphasis on trust in the context of information security [47][1][8][40][22][6]. In general, studies of trust examine trust properties, trust relationships, issues related to trust and practical implications of trust. They also focus on different processes associated with trust, such as establishing, negotiating, propagating, specifying, monitoring, managing and revoking trust, and analysing trust relationships [47][1][8][40][21][22][82]. Different models of trust are available in the literature [1][2][54][78].

However, trust can be used in any context in the same way as it is used in the context of security. Thus the possible contexts can include the scalability, reliability, and dependability, of the entities. The relevance of trust in different contexts can be illustrated by using the example below.

Consider an application developed by integrating different software entities aimed at establishing the availability of seats for a bus journey and booking seats at the

cheapest fare. For this example, consider that there are only three bus services available, namely `Bus A`, `Bus B` and `Bus C`, and each of them has its own software system to perform booking-related tasks for its own bus service. There is another software system developed and maintained by another organisation, namely `Combined Bus Service`, which interacts with the three bus service software systems to provide a combined service, which in this case is to establish the availability of bus seats and find the cheapest fare. `Combined Bus Service` interacts with another software entity, developed and maintained by yet another organisation named `Credit Card Transaction`, which performs credit card transactions. This example illustrates the kind of application that will be realised by using the Web Services framework. The figure below (*Figure 1-1*) illustrates the entities and the interaction among these entities for the application.

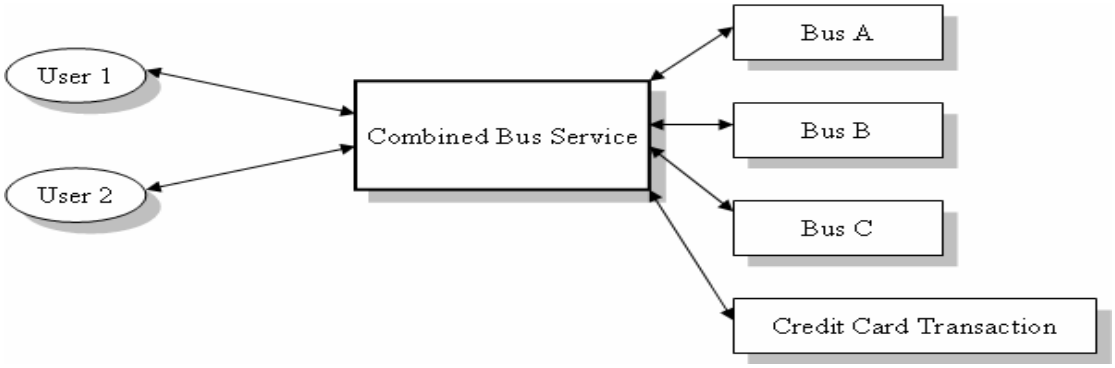

Figure 1-1: A composite application

Each box in the figure represents software entities and a double-headed arrow indicates interaction between the linked entities. The user entities are encircled and they interact with the `Combined Bus Service` for the application. The user entities could also be other software entities.

In terms of the software systems, the functioning of the application depends on the functioning of `Bus A`, `Bus B`, `Bus C`, `Credit Card Transaction` and `Combined Bus Service`. The reliability of the application is therefore based on the reliability of all the constituents, but the security of the credit card transactions is based largely on the functioning of the `Credit Card Transaction` entity. Thus trust, in the context of reliability, will be based on the properties of all the entities involved, while security of the credit card transaction will depend largely on the `Credit Card Transaction` entity.

Trust negotiation

Trust negotiation can be seen as a process towards establishing trust, which specifically deals with the exchange of information among the entities that are intending to establish trust [40]. As discussed before, context of trust influences the information required, which again affects the entities with which negotiations take place. Based on the previous example, in the context of reliability, trust negotiation should be performed with all the entities. However, in the context of secure credit card transactions the negotiation is performed mainly with the `Credit Card Transaction` entity.

A number of current systems utilise trust negotiation to establish trust between entities. The common application of these systems is to implement access control procedures [19][44][71][82]. Because these systems are tightly coupled to a specific application, they do not consider the possibility of generic trust negotiations that can be used for establishing trust in any context. In most of the systems, negotiations occur between two entities and they do not consider trust negotiations with multiple entities in combined application scenarios, such as the application given in *Figure 1-1* [44][71][82]. A thorough survey of literature has revealed that only one of the existing systems provides a negotiation procedure for this scenario [19]. However, the procedure is abstract without providing details on how to conduct such a negotiation procedure. Moreover, it does not consider the possible features of applications such as dynamicity, complexity and anonymity of service entities.

1.2 Problem statement

Evaluating the trustworthiness of a service can influence the interactions with it. One of the primary goals of the trust establishment process is to evaluate the trustworthiness of the entity and trust negotiations are performed as a step to achieve this goal. In order to perform trust negotiations in the Web Services framework, appropriate framework support is required along with negotiation procedures that utilise the framework utilities to conduct such a negotiation.

The purpose of this study is to examine the Web Services framework support for generic trust negotiations and to propose a negotiation procedure that utilises these

framework utilities. The proposed negotiation procedure is aimed at resolving negotiations in the composite service scenario.

The study includes the following:

- Defining trust and trust negotiations
- Understanding the Web Services framework and the concept of composite services in the framework
- The aspects of the composite service that affect trust negotiations
- Analyses of existing trust negotiation systems
- Analyses of the relevant Web Services framework standards to incorporate generic trust negotiations
- A trust negotiation procedure that is suitable within composite service scenario

1.3 Research methodology

A thorough literature review is done to study trust, Web Services framework, trust negotiations and the available trust negotiation systems. Based on a review of literature, arguments are constructed to define trust and trust negotiations, and as well as the implications thereof for composite services created by using the Web Services framework. A comparative study of existing trust negotiation systems is performed to identify different negotiation procedures and their limitations when applied in a composite services scenario. An improved descriptive negotiation procedure for this scenario is subsequently presented.

1.4 Delimitations of the study

This study has been conducted at a theoretical level and the same therefore applies for the suggested negotiation procedure. The study does not include a full-fledged analysis of the implementation of some of the elements that are required for this procedure, such as attribute-based credentials and negotiation strategy algorithms. The study also excludes a comprehensive architecture of a generic trust negotiation system. The features of composite services that affect trust negotiations are limited to anonymity, dynamicity and complexity.

The analysis of the Web Services framework to support generic negotiations focuses only on WS-Trust and WS-Policy specifications. The suggested extensions to these

specifications aim to demonstrate the possibility of extensions and are not intended to be exhaustive. The study excludes vocabularies for digital credentials. The full-scale specification of trust requirements and resources documents is also beyond the scope of this study.

The study is based on one notion of trust and trust negotiation, and it excludes the notion of trust in infrastructure to build applications. It also excludes the details of a computation scheme for quantifying trust. In this study, the focus of trust negotiation is on the interaction between the service provider and service requestor. Web services may or may not be controlled by human entities, but this study has not attempted to model trust as a human aspect.

1.5 Outline of the dissertation

Chapter 1 is a brief introduction to the research area.

The concept of trust is presented in Chapter 2. A summary of the different definitions, properties, processes and applications of trust in the literature is also included in the chapter. Based on the synthesis of the current literature an encompassing trust definition is presented along with its properties.

In Chapter 3 the Web Services framework is discussed as well as the concept of composite services and the framework support for these services. The chapter contains different features of composite services, which should be considered in the trust negotiation procedure.

Chapter 4 contains an analysis of current trust negotiating systems that is aimed at examining the suitability of the existing trust negotiation procedures in composite services.

The Web Services framework support for trust negotiations is evaluated in Chapter 5.

A negotiation procedure for composite services is subsequently presented in Chapter 6.

The conclusions and scope for further research are presented in Chapter 7.

CHAPTER 2
INTRODUCTION TO TRUST

2.1 Introduction

Trust is an essential building block of human society. Although it is in many cases not explicitly specified in our daily interactions and transactions, we somehow implicitly seek the basis to trust individuals, organisations and entities, for our day-to-day transactions. Based on the type and importance of the transaction, which can vary from one individual to the next, information used to base our trust differs. In human society, 'word-of-mouth' recommendations, former experiences and reliable third party references, play an important role in a trust or distrust decision. We use this concept extensively in our day-to-day transactions – often knowingly but, at times, also unknowingly. If this is true, can trust be a valid concept among artificial (machine, software-based) entities specifically in an open, distributed environment? Can trust constitute a decision-making factor for the interactions among artificial entities? To answer these questions, we need to answer the vital question 'what is trust between artificial entities?'

In the computing field, trust is studied and utilised for interactions between human entities and artificial entities, and between different artificial entities. In general these studies can be grouped into two categories; one focusing on the trust itself (which includes defining, generating and propagating trust) and others focusing on the utilisation of trust in various applications. The application and utilisation of trust is prominent in the information security domain [1][8][20][21][47][75].

Public key certificate systems are a key example of trust utilisation in the information security domain. The ultimate purpose of such a system is to facilitate the finding of a trustworthy copy of a public key. To achieve this, absolute or partial trust is placed on certain entities and this trust is utilised or propagated among multiple entities [20][21][75]. As an example in the X.509 certificate system, certificate authorities (CAs) are used to certify key certificates and entities in the system trust key certificates that are signed by the trusted CAs [21]. This system works on the basis that the CAs can be trusted for signing the key certificates. However, how this trust came to exist or how it is built is not specified in detail.

The different views of trust presented in focused studies have led to different definitions of trust [1][8][35][39][40][47][80][59]. The objectives of these studies are to understand, define or generate trust and to explore various processes involved with trust, such as trust establishment, trust negotiation and trust monitoring. Most of these studies limit the context of trust to security and hence the importance of trust in contexts other than security is not examined in depth.

The aim of this chapter is to explore trust in a generic context and the role of trust negotiation in building trust. Section 2.2 contains a discussion of three existing applications in which trust is utilised. For each of the applications the trust context and the trust-building technique are presented. Section 2.3 presents different perspectives and definitions of trust found in the literature. We investigate different perspectives with the purpose of identifying the aspects that are relevant for trust in a generic context and that can be applied among artificial entities. The section concludes with a definition that encompasses the relevant factors to meet the aforesaid goal. This trust definition is further investigated in Section 2.4, where various characteristics of trust derived from the given definition are addressed. Based on these characteristics, a basic framework for a trust generation technique is presented in Section 2.5, while Section 2.6 contains a summary of the processes involved with trust. The latter section includes only the processes associated with the definition, properties and trust quantification provided in the previous sections. The concept of a trust relationship is presented in Section 2.7 and the application of generic trust among artificial entities is discussed in Section 2.8. Some of the requirements of using trust among artificial entities are also dealt with in this section. A conclusion is provided in Section 2.9 to round off the chapter.

2.2 Existing applications of trust utilisation

Three applications in which trust is utilised are described next; two of them are from the security domain and one is based on an online e-commerce transaction.

Firewalls constitute one of the security mechanisms that are used to protect a network from possible malicious attacks. From an abstract point of view, their operation is based on a set of filtering policies. These filtering policies are based on the trust levels assigned to the various logical network segments referred to as zones [75]. As an

example, the network itself that is protected by using the firewall can be assigned a *high* trust level; the network of a partnership business unit can be assigned a *medium* trust level and all the other networks can be assigned a *least* trust level. Based on the assigned trust levels, a firewall can implement a filtering policy which states that data packets from the high trust zone are transmitted to the low trust zone without any restrictions, but all the data packets from the low trust zone to the high trust zone will be monitored and controlled. In this scenario, the trust levels are fixed and they reflect the possibility of malicious attacks from different zones.

In a distributed system, key certificates can be used to bind the identity of an entity with its public key (see Section 2.1). However, such a certificate does not ensure any other characteristic of its owner. For instance, the certificate cannot be used to determine whether an entity is reliable or not. It is up to the entity or the application that receives the certificate to make a decision about the suitability of the public key owner for the intended purposes [21]. In this scenario, key certificates are used to build trust in the public key of an entity [20][21][75]. Such a key certificate can be adequate for authentication procedures, but it may not be the only requirement for authorisation purposes.

In an e-commerce scenario - before a user utilises a service, he/she needs to trust the service provider to deliver the service as advertised [82][85]. Consider an online book-purchasing facility, which allows users to buy books and to make the payment online. The user can make use of different resources to determine the trustworthiness of the service provider. The user can check the reputation of the service provider, which can be based on the recommendations of previous users. Another mechanism is to verify the membership of the service provider with various reputable business organisations. The service provider also needs to trust the user to make the required payment. The service provider can request the credit card details of the user and enquire about the validity of the credit card before the service is granted. Here various information sources are used to determine the trustworthiness of the entities involved.

In all of these examples trust is used within a certain set of contexts. The trust level assigned or built therefore reflects trust in these contexts. For example, the public key certificates intend to answer the question ‘Can the public key be trusted?’ and not ‘Can the owner of the public key be trusted for its reliability?’ The latter question

implies the context of reliability and the former suggests the context of validity of a public key.

In all three examples, trust is generated in different ways. In the case of a firewall system, the trust levels are fixed and this way of assigning trust may not work in another situation such as the e-commerce scenario. In the case of public key certificates, trust in the public key is the function of trust that an entity places on the certificate issuers, even though there is no explicit way of presenting how this trust came to exist. In the e-commerce scenario, users are free to choose any information source to gather information about the service provider and this information is evaluated to determine the trustworthiness of the service provider. Comparing these systems, the trust generation technique used in the firewall system is most rigid and the one used in the e-commerce scenario is most flexible.

2.3 Definitions of trust

One of the common perceptions about trust is that it is subjective. In [59], trust is defined as a subjective expectation that an agent has about another's future behaviour based on the history of their encounters. Diego Gambetta [35] defines trust (or symmetrically, distrust) as a particular level of subjective probability with which an agent assesses that another agent or group of agents will perform a particular action. The subjective perception about trust is most certainly valid in human society, but not among artificial entities. The subjective perception about trust makes it difficult to find a common understanding of trust and to identify ways in which trust can be established among artificial entities. Instead we take an objective-based approach to understand trust.

The definition in [59] suggests one type of information on which to base trust, namely previous interactions with an entity. Gambetta's [35] definition involves assessment of one entity by another entity to establish trust. Among artificial entities it seems to be possible to establish trust by assessing information from different sources-previous interactions being one of them.

In [8], trust is defined as an assessment that a person, organisation or object can be counted on to perform according to a given set of standards in some domain of action. In [83] trust is defined as the assessment by which one individual A expects another

individual B to perform (or not to perform) a given action on which its (A's) welfare depends, but over which it has limited control. As stated in the previous paragraph, trust can be based on the assessment of an entity that can be guided by the guidelines of the entity. It is also worth noting that if there was no benefit or detriment resulting from an interaction, then trust would not be significant. In the online transaction scenario presented in Section 2.2, the user may well suffer financial loss if the service provider does not deliver the ordered books. There is also uncertainty about the service provider because the user cannot monitor or control any of the transactions performed by the service provider. Possible harm resulting from the interaction and the fact that the user has limited control over the service provider highlight the need for trust.

Trust is also sometimes expressed as a belief among entities in different definitions. In [80], which presents trust in peer- to-peer network, trust is defined as a peer's belief in another peer's capabilities, honesty and reliability, based on its own experiences. Here the information source specified for the trust belief is restricted to the interaction history. In [39], trust is presented as the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context. In [47], trust is defined as the belief that a rational entity will resist malicious manipulation. In [40], trust is defined as a quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context. The notion of belief can encompass uncertainties, which can be the result of the limited control and the inability to monitor another entity [65]. The advantage of trust as a quantified value is that it can enable automation and manipulation of trust among artificial entities. The trust definitions given in [39], [47] and [40] are limiting the contexts to which trust is applied. As an example, the definition given in [47] is limited to the security context. This definition would therefore fail when the context of trust is scalability. All these definitions need to be expanded in such a way that they can be applied in a generic context.

Based on the discussion in the previous paragraphs, we define *Trust* as follows:
Trust is a quantified belief by an entity (belief-holding entity) of another entity (target entity) to have a desired property, based on some set of guidelines, for specific purposes.

Trust is not more than a belief, when the belief-holding entity has limited control over the target entity and considering the infeasibilities in knowing all the factors that influence the behaviour of an entity. This definition does not limit trust to any specific context as it can be used in reference to any property of an entity. Thus it can be applied to different contexts, which includes security, scalability, reliability and quality of service. The quantification of trust (explained further in section 2.5) in any context is based on some set of guidelines set by the belief-holding entity and there is a set of ultimate purposes for which trust is established [1]. As an example, in the online transaction scenario presented in Section 2.1, the user may evaluate the trustworthiness of the service provider in the context of security, because the user may have to provide his/her credit card details for financial transactions.

2.4 Trust characteristics

This subsection describes some of the characteristics of trust based on the definition presented in the previous section. The properties explained here are not intended to be comprehensive; instead they highlight the properties that are relevant in the subsequent discussions.

The quantified belief that outlines trust is based on some information about the target entity [2][47][77]. Different types of information sources are the past interaction experiences with the target entity; recommendations from other entities; the reputation of the target entity; system guarantees; system certifications; system specifications, and assessments made by other entities [1][2][3][38][47][54][83]. The information gathering is based on the guidelines of the belief-holding entity and the collected information is then assessed to determine a quantified trust value about the target entity.

As trust has been defined in relation to an entity's guidelines, trust cannot be seen as an inherent property of any entity. Instead it is an observation made by an entity of another entity [8][65]. As each entity can have its own set of standards, different belief-holding entities can have different trust beliefs of the same entity, even in the same context and for the same purpose. Even if two entities are guided by similar guidelines, the trust value can still be different, based on the differences in the information that was collected [33].

Trust could be used with reference to any property or behaviour of an entity, which forms the context. The context within which trust is established depends on the purpose for which the target entity will be used. The context also determines the type of information necessary to quantify trust [8][77]. As an example, when the context is security, the information gathered could infer the extent to which the entity can resist malicious manipulation. On the other hand, when the context is reliability, the information gathered could indicate the failure rate and repair time of the entity [47]. Thus, a belief-holding entity can have different trust values towards the same target entity, based on the different contexts of trust and purpose for which it will be used.

Since information is used as the basis for trust, a change in the information collected will alter the trust value. For instance, as more experience is gained in interaction with the target entity or more recommendations are collected about the target entity, the belief about the entity can change positively or negatively. Thus trust based on information cannot be static and needs to be updated on an ongoing basis [2][8][40][65][80].

It is not always necessary for the belief-holding entity and the target entity to interact directly. The interactions can be mediated through a number of other entities. In such a scenario, the belief-holding entity may use recommendations of the directly interacting entities to establish trust with the target entity. Recommendations do not indicate an implicit transitive property of trust. Instead it is considered as an information source and it is up to the belief-holding entity to make a decision about the recommendations provided by other entities.

In summary, trust in a context is based on the information about an entity. Belief-holding entities are not restricted in choosing their information sources and in deciding on the significance of information contributed by each source. Moreover, trust can be dynamic and need not be transitive.

2.5 Trust quantification

The quantification of trust is utilised in various trust models and different trust computation algorithms are used for the quantification process [1][2][3][8][40]. The trust computation techniques that quantify trust vary in complexity in terms of the variables used and therefore in terms of the input parameters required for the

computation. Direct experiences that the belief-holding entity had with the target entity and/ or recommendations of other entities made about the target entity are two factors that are included in the majority of trust computation schemes. Based on the computation scheme, these factors are mapped into different trust values and various calculations are made available to combine these trust values. The models that use the quantified values use threshold values (trusted/distrusted) or a range of values (highly trusted/trusted/distrusted/highly distrusted) to make a trust-based decision. Some trust computation schemes also include factors like utility, risks, benefits and the role of an entity in the computation schemes [79][2][78][54].

Based on the discussion of the characteristics of trust in the previous section, an abstract representation of a computation scheme that can be used for building a computation algorithm is presented in *Figure 2-1* below. Such a computation scheme makes use of multiple information sources and assigns different weights to different information sources. These weights will reflect the value that the belief-holding entity places on each of the different sources. The computation scheme illustrated is not intended to be an exhaustive list of information sources that can be used to build trust, as each of the sources can also be sub-categorised into more specific information sources.

The goal of such a computation scheme is to compute trust in a specific context based on available information. In order realise such a computation scheme, there should be techniques to gather the selected information, to appropriately map this information into values and to perform appropriate calculations on these values to present a trust value, which should encompass all the relevant information selected in the computation scheme. As an example, an entity may choose ‘direct experience’ as the only information source for computing trust. Here it gathers all its previous interactions with the specific entity, maps each interaction experience to a numeric value and perform calculations to combine these values to compute a trust value.

Although the computation scheme is not further developed in this dissertation, the concept is used in Chapter 04. In this dissertation, whenever the term trust value is mentioned, it refers to the quantified representation of trust which is computed by using the scheme given in *Figure 2-1*.

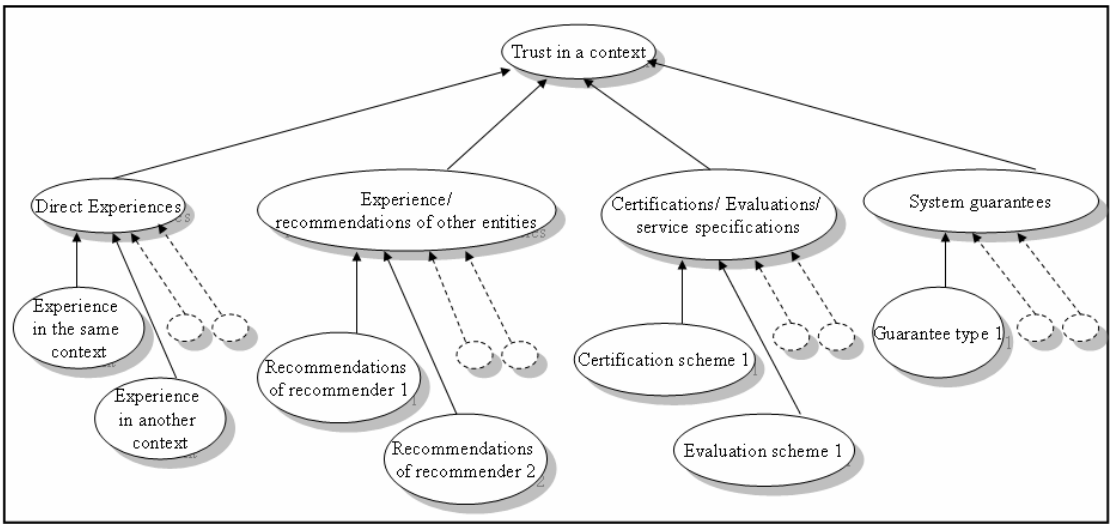

Figure 2-1: An abstract representation of a computation scheme that can be used for trust quantification

2.6 Trust processes

A trust process refers to any activity involved with the generation and updating of trust. *Trust establishment* encapsulates all the activities of establishing trust with an entity. These could include the specification of information required for trust computation, as well as the techniques to gather the appropriate information, to verify the information collected, to utilise the computation scheme for computing the trust value and, if feasible, for determining the trust relationships (discussed in Section 2.6) with the target entity.

In general, *trust negotiation* is a sub-process of trust establishment aimed at gathering information about the target entity. It focuses specifically on collecting information from the target entity itself, which is normally guided by a negotiation protocol. The process of trust negotiation is discussed further in Chapter 04.

Trust Monitoring is the process that performs all activities to update trust once it has been successfully established. It could include performing various sub-activities of the trust establishment process to update trust and trust relationships. For example, information is collected and the trust computation scheme is invoked at regular time periods to compute updated trust values. The trust negotiation process can also be included as an activity in the trust monitoring process.

2.7 Trust relationship

A trust relationship exists between two entities when the interaction between them is influenced by trust. Such a relationship asserts the computed trust value, the context for which trust is held, the purpose of interaction between the entities and the conditions or other trust relationships that need to exist before the specified trust relationship is activated [40]. A trust relationship can be specified as:

$TR_{(\text{Belief holding} \rightarrow \text{Target})}$ [Trust_context, purpose, ConditionSet]

A trust relationship is specified in terms of the trust value for a specific context, the purpose of establishing trust and the set of conditions that should be satisfied to activate the relationship. The relationship also identifies the belief-holding entity and target entity for which the relationship is specified.

The conditions that are explicitly specified may include aspects that are not considered in the trust computation scheme but that are relevant to the belief-holding entity to define a specific trust relationship. Some of the factors are risks, cost factors, and benefit of the interaction. For instance, the condition may specify that the cost of the interaction should be less than a certain value. The conditions may also specify other trust relationships that should exist in order to activate this trust relationship. This is significant when trust between two entities is influenced by the trust with its mediating entities. Here the trust relationship allows the explicit specification of trust relationships with such mediating entities.

The trust relationship summarises the trust in a context for a specific purpose. Thus an entity can have different trust relationships to the same entity in different interactions. It is important to note that since trust is dynamic, the trust relationship may also vary with the trust value. The existence of a trust relationship of the belief-holding entity to the target entity does not guarantee a trust relationship of the target entity to the belief-holding entity. Thus a trust relationship between entities need not be bi-directional [39] and every trust relationship is established uniquely. *Figure 2-2* given below illustrates two distinct trust relationships between two entities.

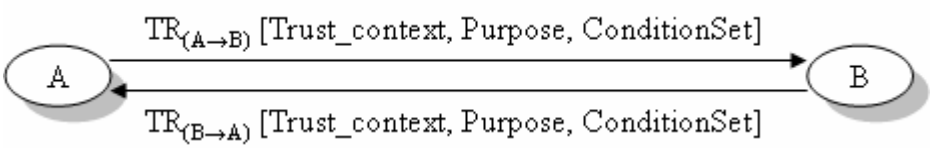

Figure 2-2: Trust relationship between two entities

2.8 Application of trust in artificial entities

Trust can be employed as a decision-making factor in the interactions among artificial entities. For example, it can be used to determine the suitability of an entity for a specific purpose and to compare different entities for a specific purpose [47][78][83]. Based on the type of information collected on which to base trust, it can be used for different applications if appropriate models are developed to include trust in various applications. Examples of such applications are access control mechanisms, to determine credibility of services and quality of service [47].

Requirements for using trust among artificial entities

The implementation and usage of trust among artificial entities would require a common vocabulary of trust [8]. Such a vocabulary would help in conveying and processing trust information among entities clearly and without ambiguity. In addition to the unified semantics, basic frameworks need to be in place for various trust processes. These would include protocols to communicate and carry trust-related information and a wide range of property-based digital credentials to convey information that is required to build trust among entities [1]. A few examples of such credentials are those required to convey recommendations, reputation and previous experiences. When entities are enhanced with trust-reasoning capabilities, additional systems should be in place to support the entities to establish, update and make decisions based on trust. These aspects are further developed in Chapter 4, 5 and 6.

2.9 Conclusion

Trust should be used in reference to a context and security is just one context among a range of other possible contexts. A computation scheme for computing trust should be flexible enough to allow each entity to choose the preferred information sources and threshold values. In order to make use of trust in various applications, appropriate models to include trust and frameworks to support various trust processes should be made available.

CHAPTER 3
WEB SERVICES AND THE CONCEPT OF TRUST NEGOTIATION IN COMPOSITE SERVICES

3.1 Introduction

The Internet is a heterogeneous distributed system that is enabled by the co-operation of various operating systems, network devices, software applications developed by using different programming languages and network configurations. These elements that contribute to the heterogeneity have their own capabilities and various application fields benefit from this heterogeneity [73]. Thus heterogeneity is an inevitable feature of the Internet system and it cannot be eliminated.

The concept of distributed computing allows the distributed completion of tasks based on the interaction of software components over a distributed system [37]. The full-scale implementation of distributed computing over the Internet involves the interaction and integration of software systems irrespective of their location and implementation [73]. This takes us to a level where the integration of software systems is not constrained within a single computer or within a set of computers limited by organisational boundaries. Instead, software systems can be built by integrating software components of different implementations from different computer locations on the Internet. One of the challenges in integrating software systems in a distributed system like the Internet is interoperability. In order to achieve interoperability, solutions need to be developed to address issues arising from the heterogeneity of the system [73].

Some of the available distributed technologies are CORBA, DCOM and RMI [37]. They do, however, have some limitations when applied over the Internet. Moreover, the difference in their object models and application level communication protocols result in low interoperability between them. Although some integration solutions were developed to improve their interoperability, the proposed integration solutions were proprietary standards. These solutions are also complex and require high communication costs and overheads [34][37][73].

Web Services propose a framework for the implementation of distributed computing on the Internet [66]. It aims to tackle the interoperability problems in a heterogeneous

distributed system so as to promote easier interaction among various software components. To achieve this goal, Web Services propose standardised interfaces and interaction between software components by using open protocols and standards [27][37][70][73]. For an existing software application, this will add an additional layer above the current middleware infrastructures to support the Web Services framework [5]. This framework supports the concept of discovering services offered by software components on the Internet. To accomplish this, the availability of software components is published on the Internet through its directory service. Thus the users who intend to make use of the service can locate the appropriate application components from the directory service and invoke them over the Internet [36][66].

By using the framework, the services advertised by the various Web services can be utilised to create larger applications or services. The constituent services of such an application, referred to as *composite service*, may not always be under the control of the same organisation or network domain [69]. The constituent services can also vary in their functional and non-functional attributes. Moreover, composite services can become complex and dynamic due to the possibility of a large number and possible changes in the constituent services. These aspects of a composite service affect the process of trust establishment with the service, which will in turn affect the trust negotiation with a composite service.

The discussion in this chapter commences with the Web Services framework and the concept of service composition using the framework. Then the concept of trust is applied in composite services and the factors that need to be considered for trust establishment with such a service are listed. We also discuss how trust negotiation is affected by these factors. In Section 3.2, a discussion of Web services is presented. This contains a definition of Web services, its architecture, essential specifications and a brief summary of its remaining specifications. The concept of composite service and a classification of composite services are presented in Section 3.3, as well as the infrastructure and specifications required to specify and execute the composite services. The concept of trust in composite services is discussed in Section 3.4, along with the factors that need to be considered in establishing trust and how these factors affect the process of trust negotiation with composite services. A conclusion of the chapter is provided in Section 3.5.

3.2 Web Services concepts, architecture and essential specifications

The World Wide Web Consortium (W3C) defines *Web services* as [84] :

"*a software application identified by a URI (Uniform Resource Identifier), whose interfaces and bindings are capable of being defined, described, and discovered by XML (Extensible Markup Language) artifacts. A Web service supports direct interactions with other software agents using XML-messages exchanged via Internet-based protocols.*"

Based on the above definition, one of the key features of the Web Services framework is the use of XML artifacts for the interaction, definition, description and discovery of software applications identified by URIs. XML therefore forms the core data representation in this framework. The specifications of the framework are based on open XML standards and Internet protocols [15][27][37][70][73][61][5]. As the infrastructure of many of the existing networks supports Internet protocols and XML standards, the deployment of the Web Services framework will greatly utilise the existing infrastructures. A software application should conform to the specifications of the framework to qualify as a Web service.

Focusing on software components that conform to this framework, they can be viewed as modular applications of various implementations residing at various network domains of the Internet. Each of the applications can provide a specific set of functionalities, referred to as *services*, which can be exposed outside the specific domain of implementation [25][36][66]. This will enable the utilisation of the service by other applications or services, which facilitates the creation of larger applications.

In the Web Services framework, the heterogeneity is dealt with by the standardisation of interfaces and interactions using XML standards. The framework specification also allows the discovery of software components on the Internet. To accomplish this, the availability of the software components is published on the Internet. Users who would like to use the application components can locate these components and invoke them over the Internet [36][66]. The architecture that assists in publishing and locating Web services is discussed in the Section 3.2.1 below.

It is important to note that the framework supports interoperability through the standardised interfaces and interaction, which do not necessarily imply complete interoperability [5]. For example, if an application employs RPC (Remote Procedure

Call) procedures and another application uses a message queuing system, the framework does not address the complete interoperability and integration of these two applications. It is left to the application to use compatible applications or to implement necessary middleware to integrate dissimilar systems. Though it does not provide a complete solution for such situations, the directory service (discussed in Section 3.2.1) in the framework can assist in finding suitable applications.

3.2.1 Web Services architecture

Two architectures are discussed in this section: an abstract architecture of the Web Services framework and an abstract architecture of the software applications conforming to the framework, i.e. the Web services.

The general Web Services architecture consists of three entities, categorised in terms of the roles needed to support Web services. These entities are the *service provider*, *service requestor* and the *service registry*. *Figure 3-1* [5][36][66] illustrates these entities and the interactions between them.

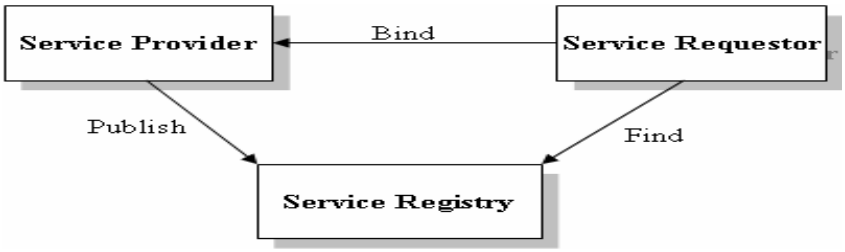

Figure 3-1: General Web Services Architecture

An organisation wishing to supply Web services takes the role of a service provider. The service provider publishes the services in the service registry so that the availability of the service is visible to the intended Web service users. In publishing a Web service it provides the service description in the service registry. The service description is published in the format of meta data and it includes the details of the interface, its implementation and network location. Thus, service descriptions should include sufficient information for a service requestor to access the service [66][36][56][5]. The service registry is a look-up registry that holds the list of services published by the service providers. Hence the service registry is the key entity that facilitates the discovery of services. Once a service is published in the service registry, the service requestor can search and find specific service descriptions from the service directory. If the service requestor finds a suitable service in the service

registry, it can use the details contained in the service description to locate and invoke, and hence to use the Web service offered by the service provider [66][36][56][5].

Though Web services can have their own internal implementation architectures, in order to support the interactions and integrations in the Web Services framework they will all have a common additional tier on their existing infrastructure. This is abstractedly illustrated in *Figure 3-2* below [5]:

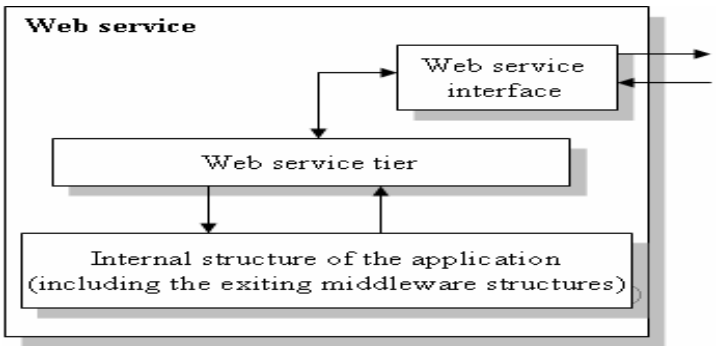

Figure 3-2: Internal architecture of Web services

All the interactions in and out of the Web service occur through the service interface, which hides the application details of the Web service. The internal implementation of the application component need not be modified but an additional tier is added into the infrastructure to add the additional capabilities to work in the Web Services environment. For example, the XML format of the service invocation that occurs in the framework may have to be restructured to conform to the application specific invocation format [5]. This is just one of the functionalities that should be included in the additional tier. Some other functionalities are discussed later in the chapter.

3.2.2 Essential Web Services specifications

The essential Web Services framework specifications are divided into areas of communication protocol, service description and service directory [27]. The core specifications in these areas are Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL) and Universal discovery, description and integration (UDDI) [66][36][27][73][61][5] respectively. The following figure (*Figure 3-3*) illustrates the basic framework architecture with these specifications.

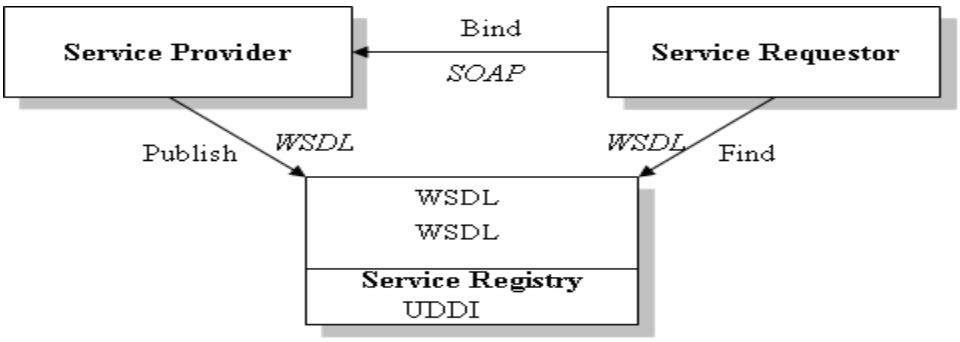

Figure 3-3: General Web Services architecture with framework specifications

SOAP

SOAP is the communication protocol for the exchange of XML-based data between applications [36][66][61][5]. It relies on existing protocols like HTTP and SMTP as transport protocols [5][61][66][36][27][28]. SOAP can be viewed in two aspects; message format and processing model [41].

The message format of SOAP specifies an envelope for transmitting messages. The SOAP message is an XML document, which contains a parent element `envelope` and child elements `header` and `body`. The `envelope` element contains the encoding details of the message; the `body` element contains the information for the recipient; while the `header` element can contain information related to security and transaction [28][41][5][61][66][45]. The following code fragment is a simple SOAP message.

```
<SOAP:envelope
    <SOAP:header>
    <t:transactionID xmnls:t=  “http://id.book.com”>
    5678
    <t:transactionID>
    </SOAP:header>
    <SOAP:body>
    <m:OrderBook xmnls:m= “http://order.book.com”>
    <m:BookName> Database system concepts </m:BookName>
    <m:Author> Silberschatz </m:Author>
    <m:Quantity> 2 </m:Quantity>
    </OrderBook>
    </SOAP:body>
</SOAP:envelope>
```

The message contains an XML document to order books. As given in the example, the ordering of the book is specified in the `body` element, while the `header` element contains the transaction id of this order in order to distinctly identify the transaction by the requestor and service provider.

In the SOAP-processing model a message contains a sender and receiver, and any number of intermediaries through which the message is routed to the receiver. The ultimate receiver processes the body of the message. The information for the intermediary nodes is included in the header and these nodes process the SOAP message accordingly (which may involve including headers in order to route the message to the ultimate receiver) [5][41].

In brief, SOAP provides a template for representing XML data, which is then sent by using the existing transport protocols [28]. Additional specifications are available to include more capabilities to the SOAP messaging. For example, WS-Security [60] is a specification to include security aspects such as integrity and confidentiality in the SOAP message. For a detailed description of SOAP, refer to [42].

WSDL

WSDL is an XML-based specification for providing a description of Web services. This description includes the data type definition, input/output messages, message formats, protocol binding, network addresses and the operations supported by the Web service [27][66][61][5]. A WSDL document is intended to provide sufficient information required for the interactions with a Web service.

A WSDL document contains an application-level service description and the specific protocol-dependent details of the service. The application-level description includes the information content of the messages exchanged in a service interaction. This would include the structure of the message, the content of the message and the operations performed on the data. The protocol-dependent details include the type of communication protocol, data format specification and network address [27]. An abstract representation of the WSDL specification is given in *Figure 3-4* below [5]:

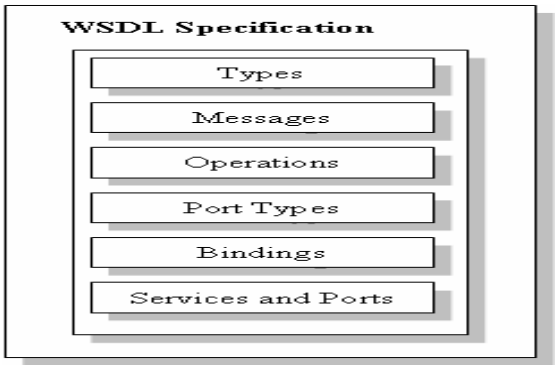

Figure 3-4: Abstract representation of WSDL specification

The following example is a partial WSDL document for service named `BookService`, according to which the above SOAP message is created:

```
<message name=  "OrderMsg">
  <part name= "BookName" type= "xs:string">
  <part name= "Author" type="xs:string">
  <part name= "Quantity" type= "xs:integer"/>
</message>

<portType name= "BookOrderPortType">
  <operation name= "OrderBook">
    <input message= "OrderMsg"/>
 </operation>
</portType>

<binding name= "BookOrderSOAPBinding" type=
"BookOrderPortType">
<binding:style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
   <operation name= "OrderBook">
<!—elements to further describe the SOAP message for this
operation- ->
    </operation name>
</binding>

<service name= "BookService">
   <port name= "BookServicePort" binding=
"BookOrderSOAPBinding">
     <address: location= http://order.book.com/>
  </port>
</service>
```

As is illustrated in the example, the message (`OrderMsg`) is defined by means of the defined data types. Operations are subsequently defined using a combination of the messages. In the example, the operation `OrderBook` consists of one message `OrderMsg`. A `portType` definition includes a combination of operations. All these elements detail the application-level specifics. The `binding` element specifies the protocol bindings. In this example, it is detailed that the SOAP messaging with HTTP protocol is used. After that a `service` element is used to detail the address at which the service is accessible.

In brief, a WSDL document describes the service interface and the address at which the service is accessible. These documents are included in the service directory as part of the service descriptions. For a detailed description of WSDL, refer to [26].

UDDI

UDDI is the specification for the Web Services registry facility. The specification includes the facilities for registering, publishing, updating, modifying and searching descriptions of Web services [66][36][73][61][5].

A service description is specified by using the XML data structures of the UDDI, namely `businessEntity`, `businessService`, `bindingTemplate` and `tModel`. The `businessEntity` data structure is used to register the details of the organisation that provides services. The `businessService` data is used to convey the services offered by the organisation. The `bindingTemplate` data structure includes the binding details of the services. The `tModel` data structure is an abbreviation for technical model, which is used to convey generic information about a service [5][61].

When a `tModel` is referenced in a service description, it means that the service complies with details of the corresponding `tModel` [5][61]. For example, consider that there are standard industry-wide interfaces and bindings for electronic book-ordering services. Then individual software components implementing such a service can follow this standard description. In this scenario, a `tModel` for such a description is registered and identified by a unique key in the UDDI. When a Web service complies with this `tModel`, it is referenced using its unique key in the `bindingTemplate` data structure of the UDDI registry entry [5][61].

The service registry in the Web Services framework is also modelled as a Web service, which provides standard interfaces for performing various functions associated with the registry. It also provides standard SOAP messages to support registering, publishing, updating, modifying and searching descriptions of Web services using different APIs. Two of these APIs are UDDI Inquiry and UDDI Publishers [5][61]. *Figure 3-5* below illustrates an abstract representation of the service registry [5].

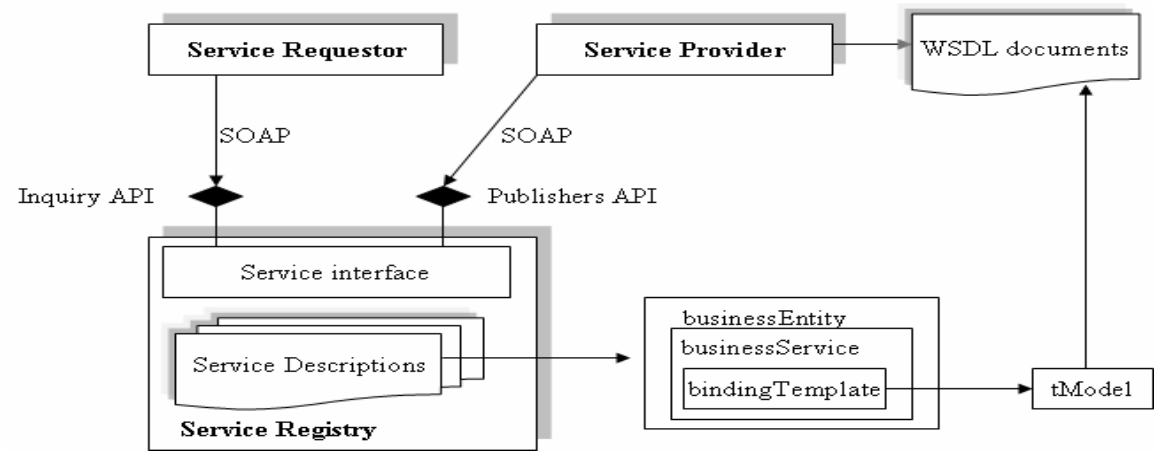

Figure 3-5: Abstract representation of the Web Services registry

In brief, the UDDI specification forms the core specification for the service registry in the Web Services framework. For a detailed description of UDDI, refer to [12].

3.2.3 Added specifications of the Web Services framework

The core specifications of the framework are SOAP, WSDL and UDDI. Many specifications were appended to the framework to add greater functionality to the framework. For instance, the initial version of SOAP specification did not address any security aspects. To address this issue WS-Security was proposed, which is an extension of the SOAP specification to address the integrity and confidentiality issues of SOAP messages [60].

The specifications in the Web Services framework can also be grouped into categories to address security, messaging, reliable messaging, transactions, business and management aspects [58]. *Figure 3-6* summarises the available specifications according to this grouping.

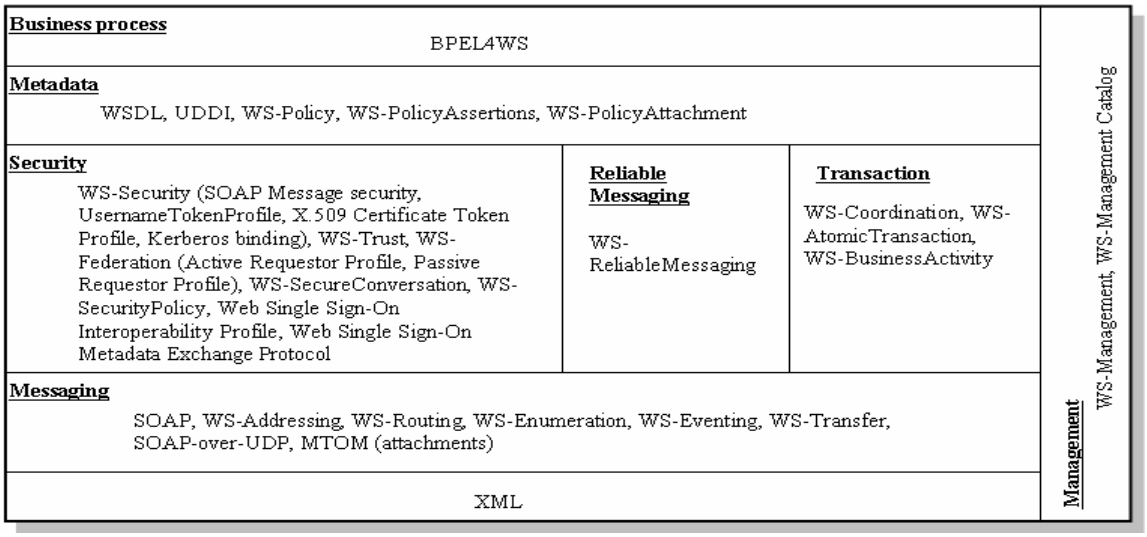

Figure 3-6: Specifications of the Web Services framework

The specifications that are further considered in this study are Business Process Execution Language for Web Services (BPEL4WS) (discussed in Section 3.3.1) [7], WS-Trust (discussed in Chapter 5) [6] and WS-Policy (discussed in Chapter 5) [9].

3.3 Composite services and service composition in Web Services framework

The Web service that utilises only one Web service to provide a service is referred to as an *elementary* service. On the other hand, the Web service that utilises the functionalities of two or more Web services to provide a service is referred to as a *composite service* [5]. Each of the services in a composite service is referred to as a *constituent* service, which in turn can be elementary or composite. The process of creation, which includes specification and execution, of a composite service is called *service composition* [5]. *Figure 3-7* illustrates a composite service, which provides the service of booking bus tickets.

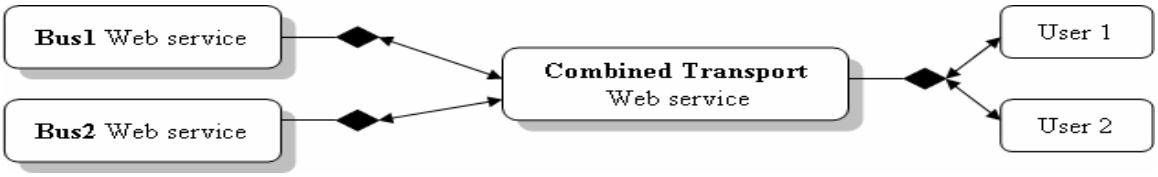

Figure 3-7: A composite service

In this example, `Bus1` and `Bus2` are elementary services, which are then utilised by the composite service `Combined Transport`. `Bus1` and `Bus2` Web services provide booking facilities for the respective bus services. However, `Combined Transport` service utilises the functionalities of these two services so that users can make bookings on either of the buses directly through this service. Here the

`combined Transport` service is the service requestor, which is also referred to as a *directly interacting entity* of the service providers `Bus1` and `Bus2`. In a similar way, `User1` and `User2` are service requestors of the `Combined Transport` service. `User1` and `User2` are referred to as *indirectly interacting entities* of the `Bus1` and `Bus2` Web services. *Figure 3-7* above is only intended to illustrate the possible interactions between different entities.

The process of service composition mainly forms part of the internal infrastructure of the composite service, which is illustrated as the additional layer in *Figure 3-2* [5]. Two significant elements of service composition are the specification and the execution of the composite service based on the specification. The specification of the service composition includes the constituents of the composite service, the order in which they are invoked, data exchanged between the constituents, the control logic and handling exceptions in the application. Such a specification defines how each of the constituents contributes to the logic of the application that it is realising. The run-time environment of the composite service uses this specification to execute the appropriate application logic, which would include invoking appropriate constituents and handling exceptions [5]. The run-time environment is part of the internal infrastructure of the Web service and it is the responsibility of each service provider to support such a system. On the other hand, the Web Services framework supports the specification of service composition using the BPEL4WS specification [15][76].

3.3.1 Specification of service composition using BPEL4WS

The BPEL4WS specification is used to describe the business process implemented by a service. In describing the process, it provides the ability to include other Web services to accomplish the tasks described by the process. It also supports the description of data exchange between these services, the control logic of the process and exceptional handling in a process. In this specification, the interactions between services are described as interactions between the WSDL interfaces [5][61][64][6].

Consider the composite service illustrated in *Figure 3-7*. The process of accomplishing a booking enquiry can subsequently be abstractedly represented as follows (*Figure 3-8*):

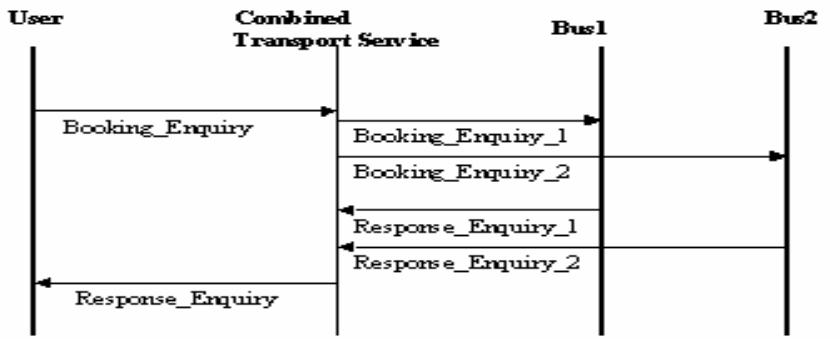

Figure 3-8: Message sequence diagram for the composite service depicted in Figure 3-7

In defining this process (specified using the `process` element) in BPEL4WS, the services that are utilised are specified using the `partnerLink` element. Such an element is used to define the relationship between two interacting interfaces in the process [5][61][64][7]. Here `user` and `Combined Transport Service`, `Combined Transport service` and `Bus1`, and `Combined Transport Service` and `Bus2` each form a distinct `partnerLink`. For each `partnerLink`, it defines the role of the participating interfaces from the perspective of the service that implements the process. For example, in the `partnerLink` defined for `user` and `combined Transport Service`, `user` takes the role of "Requestor" and `Combined Transport Service` takes the role of the "Provider".

The BPEL4WS specifies the order in which the `partnerLink` is invoked by using the constructs `sequence`, `switch`, `pick`, `while` and `flow` [5][61][64][7]. For example, in *Figure 3-8* `Booking_Enquiry_1` will sequentially follow the `Booking_Enquiry` invocation. Thus they are specified in the `sequence` construct of the `process`. On the other hand, `Booking_Enquiry_1` and `Booking_Enquiry_2` occur in parallel, which is specified in the `flow` construct. The invocation of a service is specified using the `invoke` construct, where it can specify which `operation` (as provided in the WSDL description) of the `partnerLink` is invoked along with the data (specified in `inputVariable`) that is provided to the operation. The `receive` construct is used to specify the data that is expected from the operation of the `partnerLink`.

The above description of the `process` in BPEL4WS specification uses only a limited number of constructs. In brief, BPEL4WS specification allows the description

of a process in terms of the WSDL interfaces that participate to perform various tasks in the process. For a detailed description of BPEL4WS, refer to [7].

3.3.2 Composite services: features and a classification

Service composition enables the creation of services that encompass several other Web services. The service provided by a composite service involves multiple interactions between different services according to the predetermined service specification. Such a service need not reveal the constituent services and the possible interactions to the service requestors [5]. Thus the interfaces provided by the composite Web services are capable of abstracting different interactions in a composite service. The service requestor of the composite service need not be aware of the constituent services [70] or of the underlying interactions.

Service composition can realise complex and dynamic Web services. Here complexity refers to the large number of constituent services and the possibility of multilevelled indirect interactions that can occur within a composite service. Composite services can also be dynamic considering the possibility of adding, removing and substituting constituents without necessarily affecting all the interactions in the service.

Let us consider another composite service that utilises the `Combined Transport` Web service as one of its components. This composite service implements an application to make reservations for a theatre show. According to the theatre reservation, the service provides appropriate accommodation and travel enquiries. *Figure 3-9* illustrates the possible interactions in such a Web service, where all constituent services are assumed to be Web services.

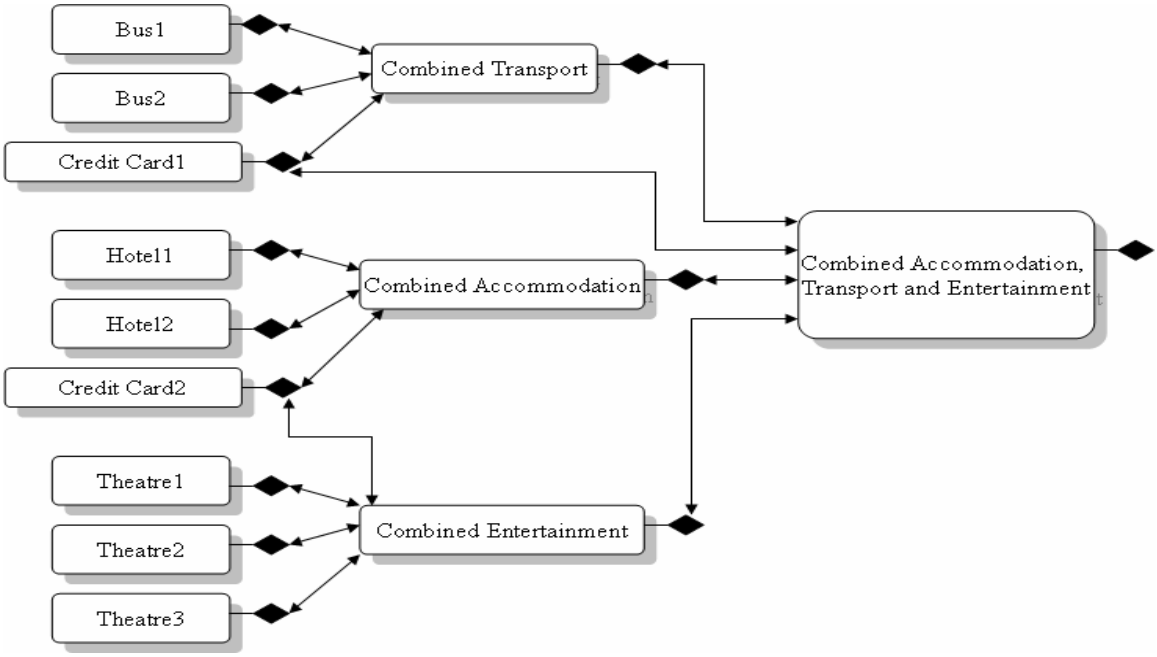

Figure 3-9: A composite service

The example illustrates the complexity of a composite service. There are thirteen constituent services in the composite service to accomplish the specific application and the application has multiple layers of interactions. The service requestor utilising this composite service interacts with only one service namely, the `Combined Accommodation, Transport and Entertainment` Web service. This service acts as an entry point to the whole application and the multilevelled interactions can be hidden from the user entity. When the user entity interacts with the `Combined Accommodation, Transport and Entertainment` Web service, it triggers a number of other interactions, which cause multiple software components to be executed, possibly across different network locations and organisational boundaries [57][70].

Composite services are capable of utilising various services but every service invocation need not result in the utilisation of all the constituent services. For example, the `Combined Accommodation, Transport and Entertainment` service can provide interfaces to utilise a subset of the services, for instance, an interface to provide just the entertainment and accommodation service. Therefore, when a service requestor requests such a service at the composite service called `Combined Accommodation, Transport and`

`Entertainment` service, it will result in utilising only the constituent services, `Combined Transport` and `Combined Entertainment`.

The composite service illustrated in *Figure 3-9* can exhibit dynamicity. For example, the `Combined Transport` Web service can deploy another credit card service namely `credit card3` for its credit card transactions. Such a replacement may affect the interaction details of the `Combined Transport` Web service with this credit card transaction module. Although this may not affect other interactions in the service, it could affect the overall non-functional attributes of the service. For instance, the performance of the service could be different after the modification.

The composite services can be classified into two categories based on the way in which constituent services are coordinated. They are referred to as mediator and peer-to-peer structure. In the mediator structure, a mediator is used to coordinate and provide the overall service. The mediator also has the global knowledge of the constituent entities and their interfaces. An example of this type is the composite service illustrated in *Figure 3-7*. In the peer-to-peer structure there is no single central service coordinating all the constituents. It also means that constituents have limited knowledge of other constituents [43]. An example of this type is the composite service illustrated in *Figure 3-9*.

3.4 Factors affecting trust establishment and trust negotiation in the composite service scenario

The features of the composite service described in the previous section cause the trust establishment to be quite distinct compared from the trust establishment with an elementary service. This section contains a list of some factors that influence trust establishment with the composite service.

Context

As mentioned in Chapter 02, the information required to build trust is based on the context. For example, when the context of trust is the reliability of the service, this may require some information about the reliability of all the constituent services. When the context of trust is the security of the credit card transaction performed by a composite service, this context does not require relevant information regarding all the

entities in a composite service; instead only information pertaining to a set of services that handle credit card transactions is required. This means that the context influences the information collected and the number of constituents from which the information is collected.

Complexity

A complex composite service can make use of any number of constituent services to realise the ultimate service. When the complexity increases, the number of indirect interactions and the depth of the levels of interaction may increase. When the complexity increases based on the context, there will be an increased number of entities from which the information is collected. In such a case finding out from which entities the information needs to be collected gets complicated- especially in a peer-to-peer structure, where constituents have only partial knowledge about other constituents.

Anonymity

In a composite web service, the identity of constituent services may not be made known to the user entity. This presents a problem to any user who wants to establish trust with the service, because information needs to be collected from the constituents but the user is not aware of the constituents in such a service.

Dynamic behaviour

Dynamic behaviour refers to the fact that constituent services could be added to, deleted from and substituted in the composite service at any time. When such changes occur, they can influence the trust values of different entities. Composite services are capable of changing dynamically in many different ways that could affect the trust values of different entities without changing much of the interaction details.

As trust negotiation is a process aimed at establishing trust, the negotiation procedure for a composite service should handle the above factors appropriately. For a specific context, the trust negotiation procedure should decide about the constituents with whom the negotiation should be performed and about what should be communicated to the negotiating entities. In order to decide on the negotiating entities, which can be a subset of the constituent services based on the trust context and information

collected, there should be techniques to resolve the issues of anonymity and complexity in a composite service. The dynamicity also influences the frequency at which trust negotiation needs to be performed with the composite service.

3.5 Conclusion

The Web Services framework aims to provide interoperability by standardising interfaces and communication between software components. Many standards are developed in this framework to achieve the set goal. One of the distinct outcomes of using the framework is the possibility of accomplishing composite services. Although such services can realise complex applications they cannot be viewed as a single unified entity due to the fact that their constituents can be of different implementations from different providers and they can vary in their capabilities. This makes the trust establishment with the composite service distinct. Depending on the trust context, the trust relationship with the composite service should encompass the trust relationships with their constituents. Factors such as complexity, anonymity and dynamicity should be considered in the trust establishment process, since they influence the trust negotiation procedure with the composite service.

CHAPTER 4
TRUST NEGOTIATION CONCEPTS AND AN ANALYSIS OF THE CURRENT NEGOTIATION SYSTEMS

4.1 Introduction

In a trust negotiation, entities disclose or exchange information in a systematic order to prove their trustworthiness [82][74][67]. This systematic order may vary from one negotiation to the next based on the participating entities, the trust context for which the negotiation is performed and the sensitivity of credentials exchanged.

One of the common applications of trust negotiation is to implement access control procedures [17][71][82]. These trust negotiation systems have a strong basis on existing security systems such as public key certificates and trust management. Hence it is possible to present an abstract comparison between public key certificate systems, trust management systems and trust negotiation systems in terms of their key elements and the way in which they are utilised in access control procedures.

One of the key elements of a trust negotiation system is digital credentials. Digital credentials follow the underlying principles of certificates in public key certificate and trust management systems. When a public key certificate binds a public key to an entity, a certificate in a trust management system binds a public key to the access rights of an entity. In a trust negotiation system, the capability of a certificate was extended to express various attributes of entities using a predetermined vocabulary. Trust management systems were introduced to realise an access control mechanism that uses capability-based certificates in a more flexible manner than the public key certificate framework, which uses identity-based certificates [21][22]. On the other hand, the trust negotiation concept was introduced to implement an access control procedure that uses general attribute-based certificates [16].

From the perspective of implementing access control procedures, both trust management and trust negotiation systems try to resolve the access rights of an entity by analysing the set of revealed credentials to prove that an entity may obtain access to a resource. Both these systems use policies and credentials as basic elements but they differ significantly in the type of credentials used and how the credentials are exchanged. This difference is discussed later in Section 4.2. Examples of trust

management systems are PolicyMaker, KeyNote, REFREE, Delegation Logic and simple public key infrastructure/simple distributed security infrastructure (SPKI/SDSI) [82]. Examples of frameworks that are based on trust negotiation principles are TrustBuilder, Trust-X, Trust-Serv, PRUNES, IBM Trust Establishment Framework, RT framework and the framework in [24].

As discussed in Chapter 2, security is just one among many possible trust contexts. In this study, the trust negotiation is therefore not examined with respect to any specific security application. Instead, the goal is to generalise the trust negotiation concept so that it can be used for negotiating trust in any context. Trust negotiation can indeed be used to establish access control mechanisms in a similar way that it can be used for other applications, for example to compare various services.

A trust negotiation is based on certain guidelines of the participating entities that possess attribute-based credentials to describe them. The essential elements of such a negotiation, as will be discussed in Section 4.3, are *digital credentials*, *policies* and *negotiation protocol* and *strategy*. *Digital credentials* are employed to state information about an entity in terms of its attributes and characteristics that are digitally signed by third parties [16][82]. Thus these credentials function as templates for communicating various aspects about entities in a negotiation. The role of *policies* is to guide a negotiation in terms of the credentials required from the participating entities, the evaluation schemes for various credentials, the disclosure policies for revealing one's own credentials and the setting of threshold values for the trust computation scheme. The *negotiation strategy* defines the order of credential exchanges in such a way that a trust negotiation can be theoretically achieved [82]. Such strategies are significant in cases where credentials are sensitive and protected by disclosure policies. The role of the *negotiation protocol* is to define the semantics and ordering of messaging required for a trust negotiation.

When analysing the current negotiation systems in terms of these essential elements, many of them focus on a subset of these elements. For example, PRUNES provides a certain negotiation strategy [85]. On the other hand, Trust-X provides support for policies, credentials and negotiation strategies [19]. In addition to support for essential elements, a number of the existing trust negotiation systems provide system

architecture [17][72][82], which describes the structure and implementation of the trust negotiation system.

Most of the existing negotiation systems focus on the trust negotiation between two entities and do not consider the composite service scenario where there is more than one independent service. The only two systems that take this into consideration are Trust-Serv and Trust-X. Trust-Serv simply identifies the requirement to unify policies of constituent services into the policy of the composite service [71]. On the other hand, Trust-X provides an abstract procedure for negotiating with the composite service [19].

The aim of this chapter is to discuss the concept of trust negotiation in detail, as well as to analyse the existing trust negotiation systems. This analysis is based on their support for essential elements of generic trust negotiation and for trust negotiation procedure in composite services. A brief summary of the evolution of trust negotiation systems is provided in Section 4.2. This discussion is aimed at providing a background to public key certificates and trust management systems, and at pointing out some commonalities and differences among these and trust negotiation systems. The concept of trust negotiation and its essential elements are next discussed in Section 4.3. In Section 4.4, the essential functionalities that should be supported by trust negotiation systems are summarised. The existing trust negotiation systems are introduced in Section 4.5, followed by an analysis to examine their support for the essential elements of general trust negotiation. The analysis also contains a discussion on how these systems support negotiations in a composite service scenario. The chapter is rounded off by a conclusion in Section 4.7.

4.2 Evolution of trust negotiation systems

In access control procedures, public key certificates can be used to identify the public key of an entity. The access rights of that public key are then determined by the application that receives the certificates [21].

Trust management systems were introduced to extend the ability of certificates to directly convey the access rights of a public key instead of binding the public key to an identity [21]. The goal of this extension is to implement more flexible access

control and authorisation systems. Application of these systems occurs in large distributed systems, where it is not feasible to be familiar with all the entities and to determine their access rights in advance [22]. Instead the systems rely on certificates issued by authorities that are capable of authorising an entity for a specific action. Here the credential issuer has some preexisting relationship with the credential owner. An example of such a relationship is that they belong to the same security domain.

The key difference between public key certificate systems and trust management systems is the way in which the access permissions are determined for a specific public key. Using a public key framework, an entity determines that the public key indeed belongs to an identity and, by using a predetermined access control list, the access rights of a key are determined [22]. In a trust management framework, access rights are directly mapped to a public key based on the certificates of an entity. From the perspective of access control procedures, public key systems are suitable in a system where the number of entities is relatively small and the access rights can be previously determined beforehand. On the other hand, trust management systems are suitable in a large decentralised system with several security domains, where it is not realistic to determine the access rights of all the entities prior to an actual service or resource request [22].

Two of the essential elements of a trust management system are *policies* and *credentials*. The *policies* in a trust management system specify the actions permitted by certain public keys and/or delegate this task to trusted third parties. Such a system uses *credentials* to express the authorisation of a key to a task. It can also contain the delegation of this task to a trusted third party. Thus, in a trust management system, when requesting for a resource *r*, the system examines whether the set of credentials *C* prove that the request *R* complies with the policy *P* [22]. Consider a simplified example:

An entity A has a security policy that delegates the task of giving access to Resource R1 to a trusted third party T1. Requestor B, requesting for the resource R1, has to obtain a credential from the third party T1. When requesting for the credential at T1, B receives a credential C1, which delegates the task of authorising access to R1 to another trusted third party T2. Assume that the third party T2 issues a credential, say C2, which authorises B to gain access to the resource R1. Now, B can submit credentials C1 and C2 to A for obtaining resource R1.

In the above example, *A* evaluates *C1* and *C2* with its policy in order to find an appropriate chain of credentials to grant *B* access to *R1*.

The functionalities of a trust management system are centered on the specification and interpretation of policies and credentials [22]. A number of systems exist that utilise the principles of trust management, where some provide full implementations of a trust management system and others provide partial support, such as policy and credential language. As mentioned earlier, examples of systems that are based on the trust management principles are PolicyMaker, KeyNote, simple public key infrastructure/simple distributed security infrastructure (SPKI/SDSI), REFREE and Delegation logic [82].

For demonstration purposes, the PolicyMaker language is chosen to briefly discuss policy and credential structure in trust management systems. In this language, policies and credentials are expressed as *assertions*. The general structure of an assertion is as follows:

Source ASSERTS *AuthorityStruct* WHERE *Filter*

The `Source` field of an assertion contains the public key of a third party when it is a credential, or the keyword `policy` to represent a policy. The `AuthorityStruct` field contains the public keys for which the assertion is applied. The `Filter` field represents a predicate that should be satisfied for the requested action, to make the assertion valid [39][21].

Consider a simple policy in PolicyMaker that states that the public key `0xfabc0001` is allowed to authorise entities to access Resource *R*.

```
policy ASSERTS 0xabcf0001 WHERE predicate= “Expression 1”
```

A credential that expresses a valid authorisation of the public key `0xfffa1111` to access Resource *R* is:

```
0xabcf0001 ASSERTS 0xfffa1111 WHERE predicate= “Expression 2”
```

`Expression 1` and `2` specified as predicates are explicit conditions that should be satisfied in order to activate the assertions.

Based on the above policy and credential, by presenting the following query to the PolicyMaker interpreter, `0xfffa1111 REQUESTS R` should return `true` if Expression 1 and Expression 2 of the above policy and credential are satisfied [39].

In trust negotiation systems, certificates describe various characteristics of an entity. These certificates are referred to as general purpose digital credentials, which conform to accepted vocabularies to describe attributes of an entity. The following XML code fragment represents such a credential [55].

```
<credential type="formal" name="ISO9000">
<business uri="..."/>
<authority uri="..."/>
<!--Elements to further describe the entity or credential-->
<issue date="2002-12-31"/>
<expiry date="2004-01-31"/>
<valid/>
</credential>
```

Trust negotiation systems realise a more flexible access control procedure than those realised by trust management systems. Therefore, instead of making access control decisions based on identity or capability, the negotiation systems uses attributes of an entity to determine access rights [81]. The attributes are stated in credentials and entities use these credentials to describe their attributes.

When comparing public key frameworks, trust management systems and trust negotiation systems, all these systems employ credentials. However, the content of the credentials vary in each system. The public key framework uses credentials to convey a trustworthy copy of an entity's public key. The trust management systems use credentials to communicate the access rights of a public key while trust negotiation systems use credentials to communicate attributes of an entity.

When comparing policy structures, trust management systems bind public keys directly to access rights or delegate this task to a third party. This delegation logic is a key aspect in the policies of these systems. In the case of policy structures (discussed further in section 4.3) used in a trust negotiation system, it expresses access rights in terms of the attributes of an entity. Even though third parties are used to issue general purpose credentials, the task of authorising an entity is not delegated to a third party. In trust management systems, the set of credentials is used to examine whether an

entity has access to a resource. On the other hand, in trust negotiation systems the credentials are used to examine whether an entity has the desired attributes and access rights can then be determined later based on the attributes of the entity.

For establishing trust in a general perspective, negotiation systems are more suitable than trust management systems and public key frameworks. This is because trust management systems and public key frameworks implement concrete security applications, while trust negotiation systems can be used for trust negotiations in any context. This obviously makes trust negotiation systems suitable for implementing diverse applications.

4.3 Essential elements of trust negotiation

Digital credentials

In trust negotiations, the attributes of entities are presented and conveyed by means of general purpose digital credentials [81][16]. These credentials describe the attributes of the credential owner using a predetermined vocabulary. The following code fragment represents a possible credential.

```
<credential type="formal" category="Certificate">
<issuer uri="..."/>
<QualityOfService>
<!--Elements to further describe Quality of the service-->
</QualityOfService>
<issue date="2002-12-31"/>
<expiry date="2004-01-31"/>
<valid/>
</credential>
```

The credentials are issued and digitally signed by authorised third parties [82][16]. For example, a credential that states an entity to be a valid credit card processing authority can only be valid when it is issued and signed by a globally accepted credential authority. On the other hand, a globally accepted credential authority need not sign recommendations in the form of a credential about an entity. It can be signed by the entity that issued such a credential, as long as the participating entity in the negotiation can resolve the credential signature.

Policies

Based on each entity's guidelines, referred to as *policies*, and the trust computation scheme, a certain number of credentials are required from the participating entities in a negotiation. These credentials are then evaluated by the entity or by third parties to determine a trust value in a specific context. The evaluation scheme and the threshold value for trusting the entity are also based on the policies of the negotiating entity. For instance, consider the policies of an entity, referred to as *A*, participating in a trust negotiation.

```
Policy1:   TrustNegotiation[(Credential1, Credential2,
           Credential3), ParticipatingEntity]
Policy2:   TrustComputation    [(Credential1,    Credential2,
           Credential3)] > (ThresholdValue=N)
```

`Policy1` states that in the trust negotiation, `credential 1`, `2` and `3` should be obtained from the participating entity. According to `Policy 2`, these credentials should be given as parameters to the trust computation scheme (see Section 2.5, Chapter 2). It also specifies that the computed trust value should be greater than the threshold value in order to consider it for an application or for further evaluation. The policies given above are partial descriptions of certain aspects of trust negotiation and they do not follow any policy language.

On the other hand - digital credentials may have sensitive information that cannot be revealed unconditionally to other entities. These credentials are therefore protected by *disclosure policies*, which outline the conditions under which a credential can be revealed [16][19]. The conditions involved can be specified in terms of the credentials of the participating entity that must be revealed or collected at a specific state of negotiation [16]. The conditions can further state the attribute values of the credentials revealed so far, or even a threshold trust value based on the evaluation of the credentials revealed so far by the participating entity. Consider that the entity A has a set of credentials named `Credential 1`, `Credential 2`, `Credential 3`, `Credential 4` and `Credential 5`. Consider that `Credentials 2`, `4` and `5` contain some sensitive information and they are protected by disclosure policies.

```
Policy3:   REVEAL:Credential2 [(Credential1),
           ParticipatingEntity]
```

```
Policy4:   REVEAL:Credential4 [((Credential1, Credential2),
           ParticpatingEntity)& (Credential1<attribute1= "..">]
Policy5:   REVEAL:Credential5[<Credential1, Credential2>,
           ParticipatingEntity) &
           (TrustComputation[Credential1, Credential2] >
           Threshold)]
```

`Policy 3` states that in order to reveal `Credential 2` of A, the participating entity should reveal `Credential 1`. `Policy 4` states that in order to reveal `Credential 4`, the participating entity should reveal `Credentials 1` and `2`. This policy places an additional restriction on `credential 1`, which states that the given attribute should match a specific value. Hence it is not sufficient to just reveal two credentials but they should also satisfy the attribute constraint to get access to `Credential 4`. In contrast, `Policy 5` states that to reveal `Credential 5`, the trust value resulting from the evaluation of `Credentials 1` and `2` should be greater than the threshold value.

In a negotiation, parts of the policies will be communicated to the participating entity. For instance, based on `policy 1`, `Credentials 1`, `2` and `3` will be requested from the participating entity. It is not necessary that the policy be sent as it is; it can be modified to convey only the essential aspects. Thus, if the participating entity requests for `Credential 4` of entity A, it may not reveal the complete disclosure policy, `Policy 5`. This is because `Policy 5` states the credentials required from the participating entity along with the evaluation of the credentials. The entity may not want to reveal how these credentials are evaluated. Hence revealing a complete policy is dependent on the strategic decision of the respective participating entity.

In a trust negotiation, the credentials that are required from the participating entity for the establishment of trust are referred to as *trust requirements*. Likewise, the credentials that an entity possesses, which can be used by the participating entity to evaluate its trustworthiness, are referred to as *trust resources*. According to the policy descriptions (`Policy 1` and `2`) of the entity A, `Credentials 1`, `2` and `3` are trust requirements and `Credentials 1`, `2`, `3`, `4` and `5` are trust resources.

When policies contain sensitive information and it is essential to reveal them in full then entities may set restrictions on revealing such policies [68]. In the same way as digital credentials are protected by disclosure policies, policies can also be protected

by disclosure policies. These disclosure policies may state the credentials revealed so far, as well as the attributes and evaluation outcomes of the credentials, to reveal a confidential policy.

It is important to note that trust negotiation is a procedure aimed at evaluating trustworthiness. Therefore, revealing all the credentials need not guarantee trust establishment and these credentials need to be further evaluated by the participating entities to define trust relationships.

When credentials have disclosure policies attached to them, negotiations are intricate. This is because every time a credential is requested, the disclosure policies need to be satisfied before it can be revealed to the participating entity [17][82][85]. Consider an example where two entities, namely *A* and *B*, are involved in a trust negotiation. It is assumed that A and B have a set of credentials to describe their attributes.

```
A←   {Credential1, Credential2, Credential3, Credential4,
     Credential5}
B←   {Credential1, Credential2, Credential3}
```

The policies of A for the negotiation are `Policy 1` and `Policy 2` given above. The policy of B is stated as follows:

```
Policy1: TrustNegotiation[(Credential5), ParticipatingEntity]
```

The policy states that `Credential 5` is required from the participating entity. Consider the scenario where the credentials of both the entities are not protected by any disclosure policies. When B is initiating the negotiation, a possible negotiation sequence can be represented as follows:

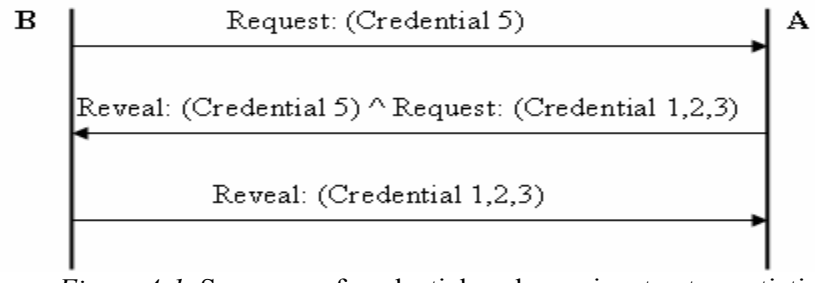

Figure 4-1: Sequence of credential exchange in a trust negotiation

Consider the same negotiation when the credentials are protected by disclosure policies. The disclosure policies of A are `Policies 3`, `4` and `5` given above. Let `Credential 3` of B be protected by the following policy:

```
Policy 2:       REVEAL:Credential3 [(Credential1),
                ParticipatingEntity]
```

A possible negotiation sequence can be represented as follows:

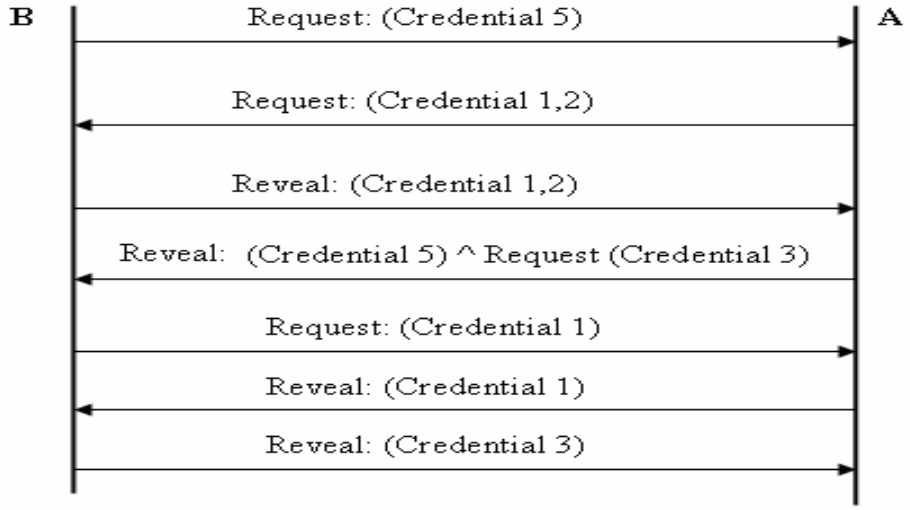

Figure 4-2: Sequence of credential exchange in a trust negotiation when credentials are sensitive

Comparing *Figures 4-1* and 4-*2*, it is evident that when credentials are protected by disclosure policies, the negotiation may not be straightforward and it might entail more credential exchanges [85]. Since these representations of negotiations are meant to demonstrate only the sequence of credential exchanges other aspects related to trust establishment, such as the credential evaluation, have been omitted from the diagram.

Negotiation protocol

Besides digital credentials and policies, another significant element of trust negotiation is the *negotiation protocol*. A negotiation protocol should enable a common understanding of negotiation among participating entities [31]. To achieve this, such a protocol should define the semantics along with the structure and ordering of negotiation messages. The protocol should be able to provide negotiation constructs for all the possible requirements of a negotiation, which includes requesting for a credential, revealing a credential, reporting a successful negotiation, revealing multiple credentials and revealing a complete policy. For example, based on *Figures 4-1* and *4-2*, the negotiation protocol defines the message semantics for `Request` and `Reveal` along with the content of these messages. It can also state the respective response messages of each message. A response to a `Request` message

can therefore be a `Reveal`, `Request`, `Error,` or `Request` and `Reveal` message.

Negotiation strategy

The aim of a negotiation strategy is to suggest a possible credential disclosure sequence that may result in a successful negotiation. The strategy is important because there is no significance in exchanging credentials if the negotiation is not theoretically achievable. The credential sequence is determined by evaluating policies of the participating parties [17][82][85]. Consider for instance the requirements of two entities participating in a negotiation as illustrated below.

Entity *A*:
Credentials of A:
`A← {Credential1, Credential2, Credential3}`
Disclosure policies of A:
`REVEAL: Credential2[(Credential3), ParticipatingEntity]`
`REVEAL: Credential3[(Credential1), ParticipatingEntity]`

Consider that A requires `Credential1`, `Credential2` and `Credential3` from the participating entity in a trust negotiation.
`TrustNegotiation[(Credential1, Credential2, Credential3), ParticipatingEntity]`

Entity *B*:
Credentials of B:
`B← {Credential1, Credential2, Credential3}`
Disclosure policies of B:
`REVEAL:Credential1[(Credential3), ParticipatingEntity]`

Assuming that A starts the negotiation, it will request for `Credentials 1`, `2` and `3` of B. B can reveal `Credential 2` and `3` but not `1`, because it has a disclosure policy attached to it. According to B's policy in order to reveal `Credential 1`, A should reveal `Credential 3`. Hence B will request for `Credential 3` of A but A's policy states that `Credential 3` cannot be revealed without receiving `Credential 1` of B. This creates a deadlock and negotiation cannot be carried out successfully, assuming that there are no alternative policies. As the example demonstrates, there is no significance in exchanging credentials if there is no feasible credential disclosure sequence.

When there is more than one possible credential disclosure sequence based on the entity's preferences, the strategy should be able to put forward the sequence with minimal credential disclosure, so as to avoid unnecessary disclosures [16][17][82][85]. A credential disclosure sequence presents a possible way of conducting a negotiation and not a successful trust relationship.

4.4 Trust negotiation system

A *trust negotiation system* is an abstract component that performs all aspects of trust negotiation in a trust system. This section contains a discussion of the main components and functionalities of a trust negotiation system in terms of the essential elements of the trust negotiation.

Policies constitute the core element of any trust negotiation. In [17][19], the collection of disclosure policies is referred to as a policy base. We generalise the term *policy base*, to refer to the collection of policies of an entity and it forms an abstract component of the trust negotiation system. In a trust negotiation, only certain aspects of a policy are revealed to the participating entity. For instance, only trust requirements are communicated to the participating entity and not the evaluation scheme for the requested credentials. It is also possible that the policies are specified by using a policy language that may not be in an interoperable format for communicating with the participating entities. Thus, it might require converting policies or even parts of policies into an accepted interoperable format [16] before they are communicated to the negotiating party. This can be achieved by using a *translation system* in the trust negotiation system.

The policies of a trust negotiation seldom remain the same for the entire life period of a service. The change can occur in respect of any aspect, including the credentials required, the evaluation schemes used and even the threshold trust values used. This means that the trust negotiation system should be able to support the process of updating policies and mechanisms to handle existing negotiations that are running when the policies are updated. This functionality of a trust negotiation system is referred to as *policy management* [71][72].

The *management of credentials* is another functionality of a trust negotiation system [82]. Whenever credentials are submitted by the participating entity, these credentials

have to be verified for validity. This process includes, among many aspects, verifying whether the credential indeed belongs to the negotiating party and verifying that it has not been revoked [16][17][82]. This component can be referred to as a *credential verifier*. In some instances certain credentials may not be revealed by the negotiating party; it might be collected from a third party. For example, the recommendation credentials or certification credentials have to be collected from a credential repository [16][72], and then the *credential collection* becomes another functionality of the trust negotiation system.

Ideally every negotiation should be followed using a negotiation strategy, which forms the basis for a credential disclosure sequence. The component of the trust negotiation system that is responsible for analysing and presenting a feasible strategy is referred to as the *negotiation strategy module* [82]. If a feasible strategy can be theoretically put forward and accepted by the entities in the negotiation, then the strategy module should steer a negotiation in terms of the credential requests among the participants.

In order to increase the efficiency of negotiations, trust negotiations systems can cache the negotiation strategy and the credentials revealed in a negotiation, so that these can be reused in similar negotiations or in a negotiation with the same participating entity for another context. In a similar way the trust negotiation systems can support the functionality of caching successful trust negotiations so that it can be reused in similar negotiations [16][17][19].

Among the existing trust negotiation systems, Trust-X, TrustBuilder and Trust-Serv provide trust negotiation system architectures. However, these architectures differ from each other in terms of their components, functionalities and abstract layout of these components. A brief summary of the various architectures is presented in Section 4.5.

4.5 Introduction and analysis of the current trust negotiation systems

The aim of this section is to introduce and analyse existing trust negotiation systems. These systems are viewed in three aspects: its support for the essential elements of negotiation; the trust negotiation system architecture; and the systematic procedure for performing negotiations using the essential elements and the system architecture.

Afterwards an analysis is presented to show the extent to which each system supports generic trust negotiations in a composite service scenario.

4.5.1 Current trust negotiation systems

The frameworks that are included in the section are Trust-X, TrustBuilder, Trust-Serv, IBM Trust Establishment (TE) Framework, PRUNES, RT Framework and the framework in [24]. Although these systems have their basis in the trust negotiation principles not all of them have a system architecture.

4.5.1.1 Trust-X

Trust-X is an XML-based framework for trust negotiations to support the process of iterative disclosure of credentials to establish trust. The purpose of trust negotiations in this framework is to implement access control procedures in order to establish trust between entities even when they do not have prior knowledge of each other. The framework works in a system where every entity can independently initiate and direct a negotiation. For this purpose, each entity is equipped with its own trust negotiation system and manages its own credentials and disclosure policies. This framework provides support for negotiating strategies, and the specification of policies and credentials [17][19].

Trust-X introduced an XML-based language named *X-TNL* for specifying both credentials and policies. A credential in Trust-X is referred to as *Trust-X certificate*, where credentials are grouped into different types and each type is defined as a distinct DTD (document type definition). Thus, a valid credential is an XML instance of the specific DTD. By means of unique credential types, each credential is uniquely defined to convey the same semantics among the entities [17][19][18]. The following example illustrates the XML encoding of a Trust-X credential that represents a digital library card [19].

```
<Library1 credID='1234', SENS=' NORMAL'>
<Issuer HREF='http://www.DigitalLibrary.com'
Title=DigitalLibrary_Repository>
<name>...</name>
<address>..</address>
<card-number code=345/>
<e_mail>..</e_mail>
<position>...</position>
```

```
</Library1>
```

The disclosure policies are also encoded in XML, which are specified as policy rules. The policy rules are expressed in terms of the credentials received from the participating entity and the constraints on the credentials to reveal a credential or grant access to a service. The rules also support specification of disclosure policies for policies, in the form of preconditions to restrict disclosure of sensitive policies [18]. The following XML fragment shows a simple policy using X-TNL.

```
<PolicySpec>
<properties>
<certificate targetCertType=Library1>
<certCond> /…/ [position = professor]</certCond>
</certificate>
</properties>
<resource target=”Online_Collections”/>
<type value=’SERVICE’/>
</PolicySpec>
```

This policy states that in order to access the `Online_collections` service, the requestor should provide credential `Library1`. This credential will be examined to check if the `position` element contains the value `professor` in order to give access to the service.

Trust-X also supports negotiation strategies. Therefore, before credentials are exchanged in a negotiation, a possible credential sequence is determined. Such a credential sequence is determined in the policy evaluation phase, which is explained later in the ‘Trust-X negotiation procedure’ section. The exchange of credentials in a negotiation is based on the credential sequence accepted by the participating entities [17][19].

Trust-X negotiation procedure

A Trust-X negotiation is carried out between two entities that are described by a set of credentials. A negotiation is carried out in different phases namely an *introductory phase*, a *policy evaluation phase* and a *credential exchange phase* [17][19].

The introductory phase comprises of exchanging credentials that are specified in the introductory policies. It is not necessary to have an introductory phase, but if introductory policies are specified, then it is mandatory for the entities to satisfy these

policies to set off a trust negotiation. The aim of this phase is to ensure that the participating entities have mandatory attributes to access the service or resource. After the successful completion of the introductory phase, the negotiation proceeds to the policy evaluation phase. In this phase, the participating entity will disclose the disclosure policies for the requested resource. The receiving entity will evaluate these policies to examine whether the credentials specified in the policy are available. If so, a response message is sent with disclosure policies that specify the credentials required from the participating entity to satisfy the earlier request. If the credentials are not available, then the participating entity reveals an alternative policy, if any. The aim of this phase is to determine a possible credential disclosure sequence for the negotiation. The third phase is the credential exchange phase, where credentials are exchanged based on the credential sequence determined in the policy evaluation phase [17][19].

If the negotiation was successful, then the negotiating parties issue *trust tickets* to each other. These tickets are XML documents to describe a successful negotiation, and they are presented in future negotiations to speed up the negotiation either by skipping one or two phases in the negotiation or even the complete negotiation [19].

Trust-X negotiation procedure for composite services

One of the features of Trust-X is the trust negotiation procedure for composite resources. When a resource *R* consists of *n* number of resources (each being elementary or composite), a successful negotiation with *R* represents successful negotiation with all *n* of its constituent resources [19].

In this procedure, when a composite resource is requested, the negotiation starts with the root element. The introductory and policy evaluation phases are executed with the root element and, if feasible, a possible credential sequence is determined. If a possible credential sequence can be determined with the root resource, the requestor performs the introductory and policy evaluation phase with the next resource in the composite resource *R*. Thus, if credential sequences can be determined for each resource, then a credential exchange phase is executed with each resource in the composite resource *R* according to the determined credential sequence [19].

Trust-X negotiation architecture

The proposed architecture for the Trust-X negotiation system is illustrated in *Figure 4-3* below [17][19]:

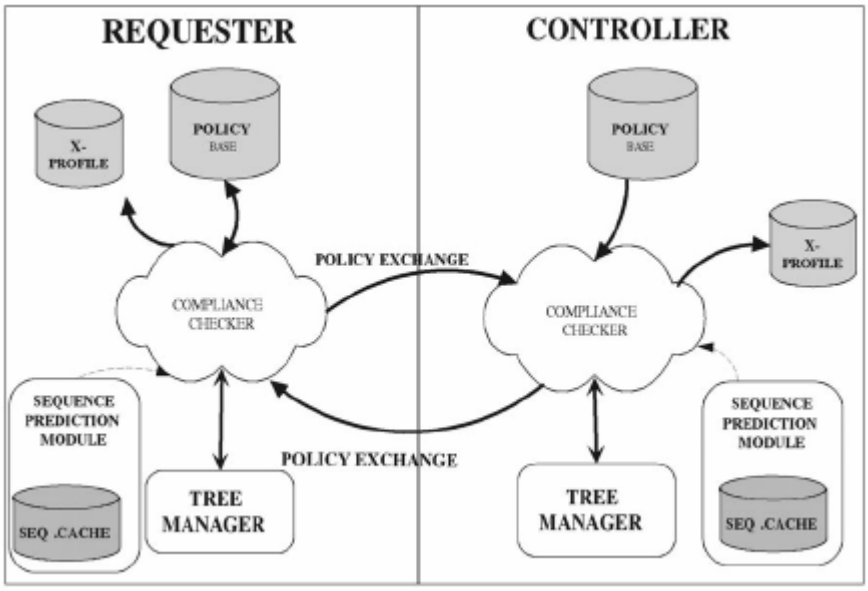

Figure 4-3: Architecture of Trust-X negotiation system

The main components of the architecture are the policy base, X-Profile, Tree Manager, Compliance checker and Sequence prediction module. The policy base represents the disclosure policies of an entity, whereas X-Profile represents its set of credentials. The tasks of the compliance checker is to check if the credentials in the X-Profile satisfy the policy of the participating entity in the policy evaluation phase and to formulate a response message based on the participating entity's request and credential verification. The role of the Tree Manager is to maintain the credential sequence determined in the policy evaluation phase and to guide the credential exchange phase according to the predetermined credential sequence. This architecture has a sequence prediction module, which aims to predict credential sequences by caching the credential sequences, for example, from previous negotiations [17][19].

4.5.1.2 TrustBuilder

TrustBuilder is a trust negotiation system for implementing access control mechanism for applications on the Internet. The aim of the system is to allow two previously unknown entities to establish trust by the iterative disclosure of credentials. This framework utilises the existing credential frameworks and policy languages. For example, X.509v3 attribute certificates and signed XML statements are used as

credentials and policies are specified using TPL (see Section 4.5.1.4) [82]. The contribution of this system is its support for negotiation protocol and strategies.

The negotiation protocol aims to define message ordering for trust negotiation. The following statements illustrate an abstract structure of a message defined in the protocol [82].

```
TrustBuilder_next_message(m,R){
Let M be the message sequence so far
Let L be this party's local resources and policies
m'=Local_strategy(M,L,R)
Send m' to the other party
TrustBuilder_check_for_termination(m',R)
}
```

The above message structure is used to formulate a message to a participating entity. The content of such a message is based on the sequence of messages exchanged between the entities so far, its local resources (credentials) and policies. In this system, the exchange of credentials is based on the predetermined credential sequence using the negotiating strategy. When there is more than one possible sequence the implementation allows the entities to choose among different sequences based on its preference for a negotiation. For example, the entity can choose a sequence with less credential exchange [82].

TrustBuilder negotiation procedure

In this negotiation model, the negotiation is carried out by a different component referred to as a *security agent* for each participating entity[82]. Appropriate policies for the requested resource, service or credential are exchanged between the security agents in order to obtain successful credential sequences. Based on a sequence, credentials are exchanged to get access to a resource, service or credential [74].

TrustBuilder negotiation architecture

Figure 4-4 below [16] represents the architecture for the TrustBuilder system. The core components of this system are the compliance checker, the negotiation strategy module and the credential verification module. The compliance checker verifies that the credentials revealed by the participating entity satisfy a policy and that they also satisfy the participating entity's policies to determine the credential sequence for a

negotiation. The negotiation strategy module is used to steer a negotiation, based on the predetermined credential sequence. The credential verifier verifies the credentials revealed by the participating entities [82].

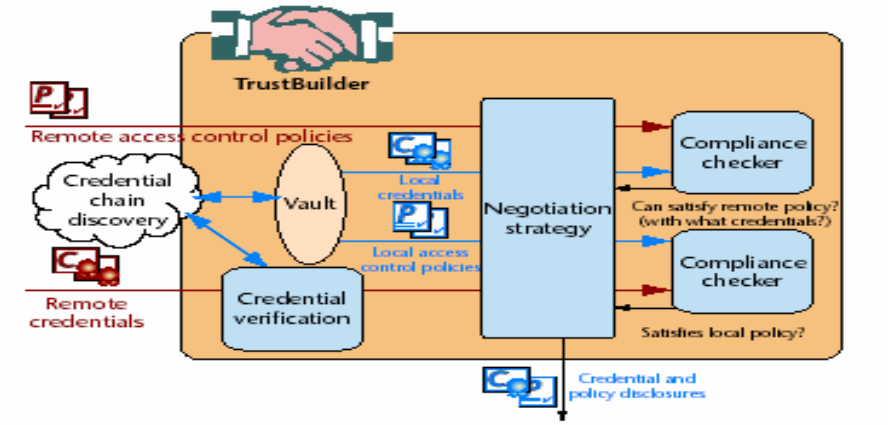

Figure 4-4: Architecture of TrustBuilder negotiation system

4.5.1.3 Trust-Serv

Trust-Serv is a trust negotiation framework proposed for Web Services. This framework also views trust negotiation as a mechanism for implementing access control procedures. The trust negotiation is presented as a better approach to traditional access control mechanism in the Web Services environment due its dynamic nature and the large number of requesting entities. This framework presents a model for specifying policies and for the lifecycle management of policies [71][72]. The lifecycle management of policies in a trust negotiation context is not discussed in other frameworks.

In the Trust-Serv framework, the policies are modelled using state machines, where states represent the protected resource and the transitions between these states represent the credentials required to access the resources. The policies describe the resources that can be revealed at a certain point of negotiation. These resources can be protected services, resources or even the sensitive credentials of the negotiation entity [72]. The following state diagram illustrates a policy that is described by using state machines.

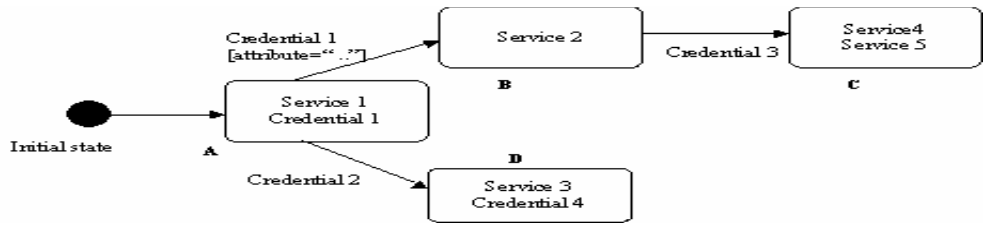

Figure 4-5: A policy described using state machines

According to the state diagram, `Credential 1` and `Service 1` can be accessed without any credential submissions. In order to access `Service 2`, the participating entity has to reveal `Credential 1` and the attribute value of the credential should match the one specified in the transition. If the participating entity reveals `Credential 2`, then access to `Service 3` and `Credential 4` is granted. If the participating entity has already reached the state marked B, revealing `Credential 3` will allow access to `Services 4` and `5`.

The second aspect of the framework is its support for the management of policies. For this, it provides strategies for updating policies and migrating ongoing negotiations to the updated policies. The three suggested ways of migration are *concurrent to completion*, *migration to new policy* and *abort*. In the *concurrent to completion* strategy, all the current negotiations are completed according to the existing policy and the updated policies are used only in the new negotiations. In the *migration to new policy*, all the existing negotiations are moved into the new policy. In doing so, certain credentials exchanged will not be valid any more and in some cases the existing negotiations have to restart from the beginning. In the *abort* strategy, all the negotiations are cancelled and new negotiations are started with the updated policy [71][72].

Trust-Serv also uses attribute-based credentials but does not provide a credential specification. Instead, it uses the existing credential structures such as SAML, X.509v3 and SPKI [72]. The Trust-Serv framework uses SOAP as the negotiation protocol but it does not provide any details of the negotiation message structure. Moreover, it does not put into perspective the extent to which the standards provided by the Web Services framework can be used for trust negotiations.

In terms of composite services, it is pointed out that there are problems in formulating policies for such services. One of the problems is formulating policies for a composite service, which should encompass policies of the constituent service. Another problem is validating a policy when a requestor reveals a credential to the composite service, which entails the process of ensuring that the policies of all the constituent services are satisfied before a service is granted [71].

Trust-Serv negotiation architecture

The Trust-Serv system provides a system architecture, which is depicted in *Figure 4-6* given below [72]:

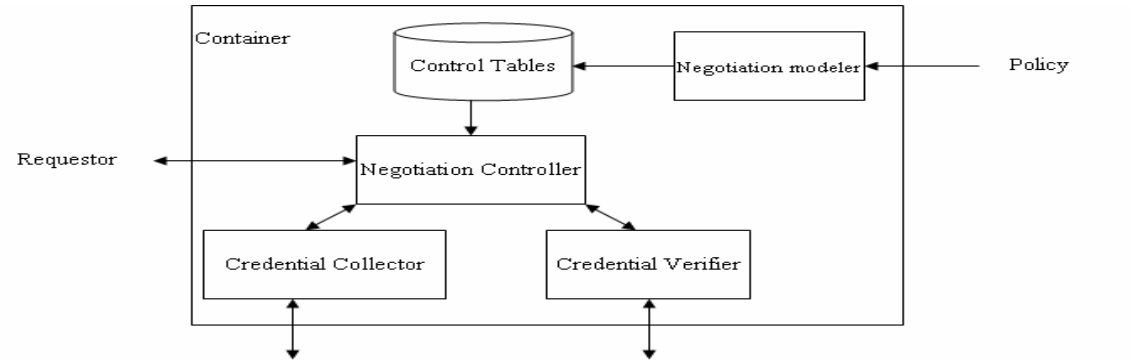

Figure 4-6: Architecture of Trust-Serv negotiation system

The system components are the negotiation modeler, control tables, negotiation controller, credential collector and verifier. The negotiation modeler is a tool to assist policy developers to specify and modify policies using state diagrams. These policies are converted into XML format, and they are collectively referred to as control tables. The component credential verifier performs the verification of the submitted credentials of the requestor. Unlike other negotiation systems, this architecture includes a credential collector component, which collects credentials of the participating entities in case the credentials are not managed by the participating entities. The negotiation controller controls the negotiation, receives messages from the requestor, requests for credentials and gives access rights based on the control tables. In this architecture, the credential collector and verifier may be independent Web services [72].

4.5.1.4 IBM trust establishment (TE) framework

The application of this proposed framework is to support role-based access control procedures by using attribute-based certificates specifically in an e-commerce environment [44][39].

One of the aspects of this framework is an XML-based policy language named Trust Policy Language (TPL). In this language, a policy is formulated by defining different groups of members. To be considered as a group member, requesting entities should submit certificates and the policy could state the conditions under which the certificates are valid. Such conditions include specific attribute values in the

certificate and inclusion of external functions to evaluate the revealed certificates. Thus, when a user submits certificates, they are evaluated according to the policies and the requesting entity is classified into a group. Based on the group, access rights are given [44][16]. The following is a simple policy specified using TPL.

```
<POLICY>
<GROUP NAME=”Group1”>
<RULE>
<INCLUSION ID= “EmpIDCert”, TYPE= “EmployeeID”, FROM=
“EducationInstitution”/>
</RULE>
</GROUP>

<GROUP NAME=”Group2”>
<RULE>
<INCLUSION ID= “EmpIDCert”, TYPE= “EmployeeID”, FROM=
“FinancialInstitution”/>
<FUNCTION>
<AND><EQ>
<FIELD Position=”MANAGEMENT”/><FIELD Institution= “BANK-A”/>
</EQ>
</AND>
</FUNCTION>
</INCLUSION>
</RULE>
</GROUP>

</POLICY>
```

According to this policy there are two possible groups namely `Group1` and `Group2`. If the requesting entity submits an `EmployeeID` credential from an Education Institution, it will be grouped into `Group 1`. However, to become a `Group2` member, the `EmployeeID` certificate should be issued from a financial institution and the fields `Position` and `Institution` in this certificate should match to `management` and `Bank-A` respectively.

In TPL, a policy specifies the conditions under which the entity can be grouped into specific groups and the mapping of groups into access rights is done separately [39]. The TE framework does not provide a credential specification, but it uses the existing credential frameworks such as the X.509v3 attribute framework. This framework also lacks support for negotiation protocol or strategy. Moreover, TPL does not make provision for handling sensitive credentials or policies [16].

4.5.1.5 PRUNES

PRUNES provide an algorithm for implementing a negotiation strategy, which provides a credential sequence whenever feasible with minimum credential disclosures. This algorithm takes into account the disclosure policies of credentials and policies when it determines the credential sequence [85].

4.5.1.6 RT framework and the framework in [24]

The RT framework is a logic-based system that provides a set of languages that can be used for specifying policies and credentials for implementing role-based access control procedures. This framework has combined the concepts of role-based access control systems and trust management systems. The concept of roles is taken from the role-based access control system where the roles of an entity are determined by using its attributes. These roles are then mapped into access rights. The concept of credentials is taken from trust management systems. In this framework credentials are used to certify the attributes of an entity. The system also delegates the authority of certifying attributes to other entities. When an entity submits a set of credentials, the RT system, based on its policies, evaluates the credentials to infer the attributes and determines the roles of an entity [51][52]. A simple example, based on [52], is given below.

An entity, say *A*, grants service S1 to entities that are preferred customers of its parent organisation P and IEEE members. This policy can be logically stated as follows:

$$A.S1 \leftarrow P.\ PreferredCustomer \cap IEEE.Member \qquad (1)$$

However, to be a preferred customer of the parent organisation P, the customer should possess an Excellent Customer certificate, which is issued to outstanding customers. This policy is represented as:

$$P.PreferredCustomer \leftarrow P.ExcellentCustomerCertificate \qquad (2)$$

For a requesting entity B to access the Service S1, it should have credentials that convey the following:

$$IEEE.Member \leftarrow B \qquad (3)$$
$$P.ExcellentCustomerCertificate \leftarrow B \qquad (4)$$

The service entity A will grant S1 to B based on its policy (1) and P's policy (2), and by evaluating credentials 3 and 4.

The framework in [24] provides support for implementing access control for service access and for limiting information disclosure on the Web, using the principles of trust negotiation. This is achieved by using attribute-based credentials and policies that specify the access rights in terms of these credentials, and that disclose policies and credentials. The contribution of the paper is that it provides a logic-based language to reason about policies and credentials for service access and information disclosure [24].

4.5.2 An analysis of the current negotiation systems

The current trust negotiation frameworks are summarised in Section 4.5.1. The common aspect of these systems is that trust negotiation is mainly used to implement access control procedures by using attribute-based credentials.

From the generic trust negotiation point of view, as discussed in Sections 4.1 and 4.3, the role of credentials is to represent information about the entities. The concept of attribute-based credentials used in the existing trust negotiation systems is appropriate for the credentials in generic trust negotiations. All of the discussed systems employ attribute-based credentials, where some of them provide their own credential specification, for example Trust-X, and the others reuse the existing credential frameworks, for example TrustBuilder.

In order to fully realise general purpose negotiations, more vocabularies for credentials and trust contexts are required. Some examples are vocabularies to convey Quality of Service, Reliability of services and new credential types (such as the recommendation credential) to convey the recommendation of an entity about another entity.

The role of the policies in generic trust negotiation is to guide a negotiation in terms of credentials required from the negotiating entity limit the disclosure of own credentials and policies and set preferences in computation schemes and threshold values. Hence policies should form the ultimate authority in a trust negotiation system. As the existing trust negotiation systems use trust negotiation to implement access control, they limit the capability of the policies to express only what is required from the negotiating entity and the disclosure policies for sensitive credentials and policies. These systems do not use the concept of computing trust based on the

credentials. Therefore they lack the flexibility to include the utilisation of external functions, not just to validate a credential or attributes in a credential, but also to include a computation scheme, which can consider, if required, factors such as risks, cost factors and benefits to provide an all-inclusive trust value. The policies in a general purpose trust negotiation should be able to express the invocation of functions to perform such evaluations. Among the existing trust negotiation systems, the TPL of the IBM TE framework provides for the facility to include functions to be utilised in a trust negotiation [44].

A realistic aspect of a trust negotiation system is the possibility of limiting the disclosure of sensitive policies and credentials. Most of the trust negotiation systems, except IBM TE and Trust-Serv, include the notion of sensitive credentials and policies. When credentials and policies are sensitive, the need for a negotiation strategy to predetermine the credential sequence becomes necessary. This concept is also addressed in existing frameworks like Trust-X, PRUNES and TrustBuilder. The negotiation protocol is furthermore an essential element of trust negotiation and the only system that addresses this in an abstract way is the TrustBuilder.

In terms of the trust negotiation system architecture, the three systems that provide concrete structures are Trust-X, TrustBuilder and Trust-Serv. All these systems use policies to guide the process of negotiation, but only Trust-Serv has taken into account the process of managing policies. The systems that support negotiation strategy modules are Trust-X and TrustBuilder. On the other hand, caching of trust sequences in order to increase efficiencies of negotiations is included only in Trust-X. All these three systems support credential verification functionality but only Trust-Serv supports credential collection.

The lacking feature in all of these systems is an in-depth negotiation procedure that can be used in a composite service scenario. The only two systems that consider such a scenario to a certain extent are Trust-Serv and Trust-X. Trust-X provides an abstract trust negotiation procedure for a composite service scenario, but leaves out the details like policy composition, the coordination of negotiations with different elementary services and how the dynamic nature could affect the negotiations. On the other hand, in [71], it is pointed out that policy composition in composite services should be based on the policies of its elementary services.

Table 4-1 below presents a summary of the existing trust negotiation systems in terms of their support for essential elements of negotiation, trust negotiation system architecture and negotiation procedure for composite services.

Trust Negotiation system	Utilisation and/or support for essential elements of trust negotiation				Support for trust negotiation system	Support for negotiation procedures for composite services
	Digital Credentials	*Policies*	*Protocol*	*Strategy*		
Trust-X	√	√		√	√	√
TrustBuilder	√	√	√	√	√	
Trust-Serv	√	√			√	
IBM TE	√	√				
PRUNES				√		
RT	√	√				

Table 4-1: A summary of various trust negotiation systems

4.6 Conclusion

Trust negotiation can be viewed in a general perspective without binding it to an application. Even though the existing negotiation systems are mainly used for access control implementation, they can be reused extensively to implement generic trust negotiations. However, these systems do not provide adequate techniques or procedures for trust negotiations in composite service scenario. This suggests that there is a need for negotiation procedures that work in a composite service scenario, but which can also be applied for elementary services.

CHAPTER 5
A FRAMEWORK FOR TRUST NEGOTIATIONS IN WEB SERVICES

5.1 Introduction

As discussed in Chapter 4, a framework for trust negotiations in Web Services is based on three facets: its support for essential elements of negotiation (digital credentials, policies, negotiation protocol and negotiation strategy); negotiation system architectures, and negotiation procedures that are suitable for elementary and composite services. Some of these facets should be incorporated within the framework and remaining by Web services itself.

Among the essential elements of trust negotiation, digital credentials and negotiation protocol should be common to all Web services to facilitate trust negotiations. Thus support for these two elements should be included in the Web Services framework. With regard to specifying policies, Web services may deploy different policy languages, but the aspects of the policies that are exposed to the negotiating parties, e.g. trust requirements and resources, should adhere to an accepted specification. The implementations to support negotiation strategy and negotiation system architecture can be Web service specific. Likewise, the trust negotiation procedure can also be service specific, but it should facilitate the use of the framework support for negotiations.

In the Web Services framework, WS-Policy [9] is the policy specification standard and WS-Trust [6] is the specification that closely fits the description of a negotiation protocol. However, the Web Services framework does not have a digital credential framework specification.

The Web Services Policy (WS-Policy) framework is an XML-based specification, which provides the syntax to specify the policies of the Web services. In brief, the policies are intended to specify two main aspects of a Web service: the *service requirements* and the *capabilities of the service provider*. The *service requirements* include the requisites that the intended service user must provide to utilise the service. For example, the service requestor can state that the service provider must be able to provide an X.509 certificate. The second aspect of WS-Policy is to state *service*

capabilities, which refer to the properties or capabilities that emerge from the configuration of the service. An example of service capability is the various facets of the service to express the quality of the service provided. The set of encryption algorithms supported by the service can also be considered as a capability [9].

As part of the service requirements specified in the service policy, the service provider can state the credential issuers that it trusts to issue credentials. The intended service user can therefore request and receive credentials from the specified credential issuer. The WS-Trust framework specifies the syntax to support this process of requesting and receiving credentials by using SOAP messages [6].

As discussed in Chapter 4, a trust negotiation policy specification should be able to specify trust requirements and resources among various other aspects of negotiation. The ability of WS-Policy to specify service requirements and capabilities suggests its suitability for specifying, mostly, trust requirements and resources.

The elements in the WS-Policy specification emphasise the specification of security-related policies of a Web service. The WS-Trust specification also focuses on the exchange of security-related credentials between the credential requestor and issuers. However, to support generic trust negotiations, policies should be flexible enough to incorporate any trust context. Likewise, negotiation protocol should be able to support the exchange of any digital credentials, including security credentials between the service provider and requestor. With appropriate extensions, WS-Policy and WS-Trust can be utilised in generic trust negotiations. A number of the possible extensions are discussed later on in this chapter.

The aim of Chapter 5 is to analyse the Web Services framework for its ability to support generic trust negotiations. In Section 5.2, an overview of WS-Policy is presented. Several requirements for specifying trust requirements and resources in Web Services are discussed in Section 5.3, along with an analysis to examine the extent to which WS-Policy satisfies these requirements. Based on this analysis, a number of possible extensions to WS-Policy to specify trust requirements and resources are discussed in Section 5.4. An overview of WS-Trust is presented in Section 5.5, while Section 5.6 contains a few illustrations of how WS-Trust

specification can be used as a negotiation protocol. The chapter is concluded in Section 5.7.

5.2 An overview of the WS-Policy specification

The Web Services Policy (WS-Policy) is the XML-based policy specification of the Web Services framework. This specification provides the base constructs to specify policies in terms of the requirements and capabilities of the services. The discussion below is taken largely from [9].

The WS-Policy specification defines a policy to convey conditions concerning interaction between two Web service endpoints. These conditions guide service requestors to choose suitable services. A policy of a Web service is represented as a collection of *policy alternatives* where each policy alternative refers to a collection of *policy assertions*. An *assertion* is used to express a distinct unit of requirement or capability of a service. The WS-Policy provides constructs to group these assertions to outline policies of the Web services.

A policy assertion contains domain-specific conditions or information, where domain can be seen as analogous to a context. The vocabulary and semantics of assertions of each domain (or context) are defined in separate domain specifications, referred to as domain vocabularies.

The following (*Policy 1*) illustrates how a policy document is specified:

```
1 <wsp:policy>
2 <wsp:ExactlyOne>
3 <wsse:SecurityToken>
4 <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
5 </wsse:SecurityToken>
6 <wsse:SecurityToken>
7 <wsse:TokenType>wsse:X509v3</wsse:TokenType>
8 </wsse:SecurityToken>
9 </wsp:ExactlyOne>
10 </wsp:policy>
```

Policy 1 states that either a Kerberos ticket or X.509 certificate is required. Though the purpose of this requirement is not specified in the policy document, it can be assumed that these are some requirements for authentication, based on the contents of

requirements or the way in which the policy is attached to the WSDL elements of the service, for example, using WS-PolicyAttachment specification [10].

The above policy is expressed as two distinct policy assertions, given in lines `3-5` and lines `6-8`. The grouping of these policy assertions are specified by using the element `<wsp:ExactlyOne>`, which states that only one of the given assertions need to be satisfied. The policy assertions used in this policy are based on the WS-SecurityPolicy vocabulary [29].

In WS-Policy, a policy document is specified by using the XML element `policy` and all the policy assertions are contained as subelements of the `policy` element. When multiple policy assertions are provided, policy operators defined in WS-Policy can be used to facilitate groupings of policy assertions into policy alternatives. These policy operators are `<wsp:All>` and `<wsp:ExactlyOne>`. The former specifies that all the policy assertions and/or policy alternatives grouped in this element must be satisfied, while the latter (`<wsp:ExactlyOne>`) specifies that exactly one of the policy assertions and/or policy alternatives must be satisfied. The elements `<wsp:All>`, `<wsp:ExactlyOne>` and `<wsp:Policy>` can be nested within the elements `<wsp:All>` and `<wsp:ExaclyOne>`.

In general, the domain vocabularies are identified using distinct namespaces. The WS-Policy specification assists the grouping of assertions of different domain specifications uniformly to outline policies. *Figure 5-1* below illustrates the concept.

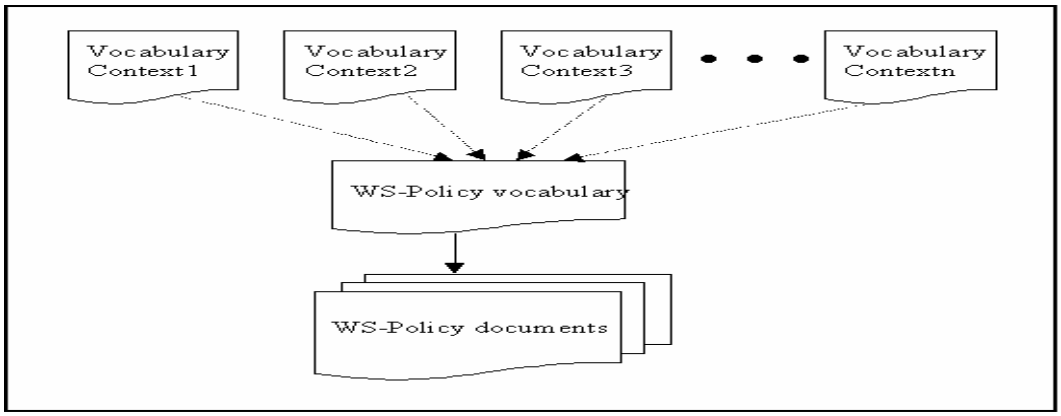

Figure 5-1: Abstract representation of WS-Policy documents

Referring to *Policy 1* (page 71), the domain of the policy assertions is security and the vocabulary for the assertions is specified in the XML vocabulary named WS-

SecurityPolicy specification, which is identified by the namespace http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd. The namespace is referred to by the prefix *wsse* [29]. In general, WS-SecurityPolicy specifies assertions for different facets of security, which includes integrity and confidentiality aspects. For further information on WS-SecurityPolicy, refer to [29].

5.3 Specification of trust requirements and resources in Web Services

In order to specify and present trust requirements and resources in a trust negotiation specifically in Web Services, we introduce the concept of a *trust requirements document* and a *trust resources document*.

The description of the trust requirements and resources of a service is referred to as a trust requirements document and a trust resources document respectively. A trust requirements document (*TRtD*) is a document that describes the partial or complete trust requirements that a Web service require from the negotiating entity in the trust negotiation. A *trust resources document* (*TReD*) is a document that describes the partial or complete trust resources of a Web service, which can be revealed to the negotiating entity in a trust negotiation, if required. A TReD should also be able to specify the trust requirements or reference to a TRtD for those trust resources that are protected by disclosure policies. The TRtDs and TReDs are revealed to the negotiating entities to convey the trust requirements and resources needed in a trust negotiation.

In general, these documents specify trust requirements and resources in terms of the respective credentials. Thus, it should specify aspects such as the details of the credentials, the frequency of credential submissions and address of credential submission or request. The details of the credentials include the content, accepted credential issuers and format. The frequency of credential submission indicates how often credentials are to be revealed to the participating entities. This may range from a once-off submission to submissions at every service request. The frequency may also correspond to the frequency at which trust relationships are updated. Specifying the address of the credential submission and request gives the flexibility to include third parties for credential evaluation and retrieval. In summary, these documents not only represent trust requirements and resources, but also a complete description of

how the respective credentials should be presented and the necessary details to submit these credentials.

In Web Services, TRtDs and TReDs should be specified on the basis of an accepted XML vocabulary. Once they have been specified they should be revealed to the negotiating party either by publishing it through UDDI or sending the documents to the negotiating parties during negotiations. It is also possible that when the number of trust requirements and resources is large, they are specified in subdocuments and, in this case, some of them are published in the UDDI and the rest are exchanged during negotiations.

When comparing the features of WS-Policy and the requirements of TRtD and TReD specifications, it is evident that WS-Policy has certain limitations with respect to the requirements. These requirements can be met with appropriate extensions to WS-Policy. In the discussion below we illustrate a few extensions in this direction. One of the key goals of the extension is to incorporate various trust contexts in WS-Policy to specify TRtDs and TReDs. The suggested extensions are by no means complete, but they aim to illustrate the possibility of increasing the capability of WS-Policy for specifying TRtDs and TReDs.

The specification of the full structure of the TRtD and TReD is beyond the scope of this dissertation. We do however show how some of the aspects can be specified.

5.4 Possible Extensions to WS-Policy to specify TRtDs and TReDs

Extending vocabularies to accommodate further information resources and contexts

To support generic trust negotiations it is necessary to have vocabularies for different contexts and information sources, which are specified using formal and/or informal credentials. The formal credentials are those credentials that are globally recognised and signed by authorities that are trusted to do so. ISO certification is an example of such a credential. On the other hand, informal credentials are those that need not be globally recognised. Examples of such credentials are those issued by a service to state its service guarantee or recommendations of another service.

One way of including various contexts and information sources in policies is illustrated in *Figure 5-2*. Here vocabularies for various information sources are defined in separate vocabularies and they are reused in specific context vocabularies.

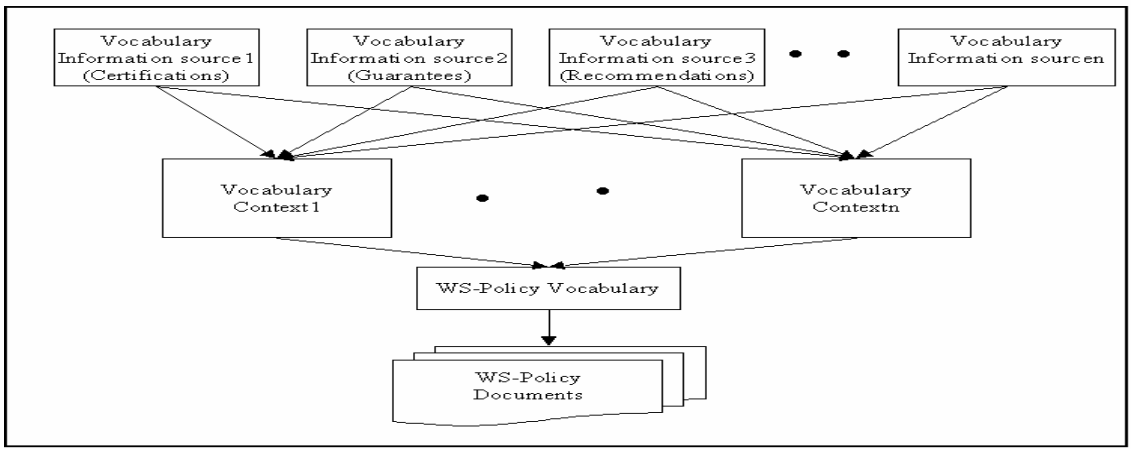

Figure 5-2: Abstract representation of WS-Policy documents with various contexts and information source vocabularies.

An example of this concept is to define a vocabulary for specifying recommendations, which will consist of a collection of credential formats to detail various forms of recommendations. This vocabulary can then be used in any context vocabulary to specify recommendations in the specific context, e.g. reliability. The details of such vocabularies are not considered in this dissertation but it is assumed that they can be made available.

Figure 5-2 closely relates to the aspects depicted in *Figure 2-1* (Chapter 2). Each information source in *Figure 2-1* has a specific vocabulary as depicted in *Figure 5-2*. By using this technique, credentials representing various information can be included in the TRtDs and TReDs.

To specify trust requirements and resources separately

When vocabularies for various contexts are available, certain credentials can become part of both trust requirements and resources of a Web service. In generic trust negotiations, it is vital to identify TRtDs and TReDs distinctly. One possible way of realising this is by using WS-Policy is illustrated in the *Figure 5-3*. The goal of this concept is to distinctly identify assertions as trust requirements or resources of a service.

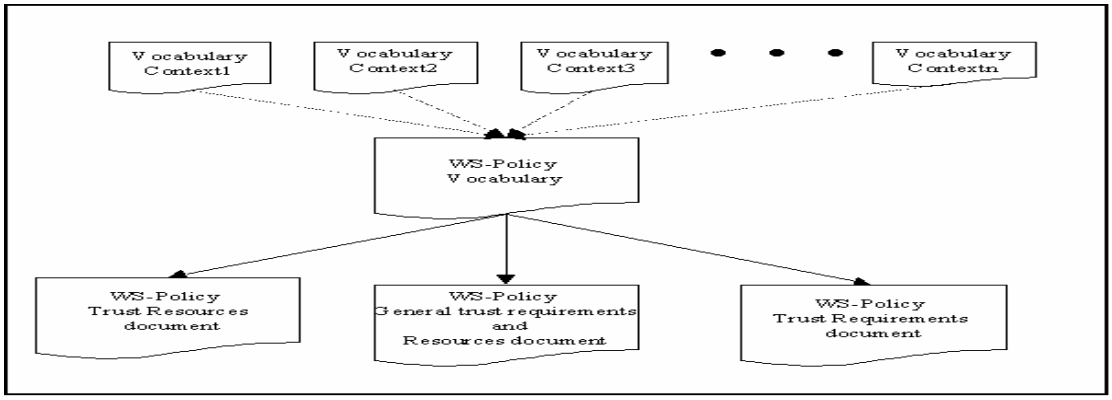

Figure 5-3: Abstract representation of trust requirements and resources documents using WS-Policy

In general, an assertion can be identified as a trust requirement or resource in two different ways. One technique is based on the content or semantics of the assertion. In this technique, if the assertion can be of both requirement and resource, then they are distinguished either by distinct assertion names or by the content of the assertion. The second technique is to create a single assertion and to identify it as a requirement or resource in the policy construct. Between these techniques, the second one allows the specification of trust requirements and resources more explicitly in the TRtD and TReD specification.

The WS-Policy specification uses the first technique to distinguish service requirements and capabilities. Thus, the distinction of service requirement and capability is made at the assertion level and not at the policy construct level. Consider a simplified version of a policy (*Policy 2*) that states Kerberosv5TGT as a security token.

```
<wsp:Policy>
<wsse:SecurityToken>
<wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
</wsse:SecurityToken>
</wsp:Policy>
```

In this policy, `<wsse:SecurityToken>` is an assertion to specify a security token, which is either required from the requestor or provided by the service in a response message. However, the content of the `<wsse:TokenType>` element identifies this assertion as a requirement.

To identify a policy explicitly as a trust requirement or resource in WS-Policy, an optional attribute named `Type` can be added to the element `<wsp:policy>`. This

requires an amendment of the current version of the specification. The attribute `/wsp:policy/@Type` can be of type `xs.string`, which can be assigned values such as "`Requirement`" or "`Resource`" to identify policies as requirements or resources respectively. These two values need not be the only values for this attribute and more values can be identified and added. The inclusion of such an attribute to a policy is illustrated below:

```
<wsp:Policy wsp:Type= "wsp:Requirement">
<wsp:All>
<!--Assertion 1-->
<!--Assertion 2-->
</wsp:All>
</wsp:Policy>
```

The absence of the `Type` attribute indicates that the semantics of the assertion should determine whether it is a requirement or resource. It is important for the semantics of the assertion not to contradict the `Type` attribute. For example, in the above policy, when the `Type` value is set to `Requirement`, the semantics of assertions 1 and 2 should not refer to them as resources in their domain vocabulary.

As illustrated in the above policy, when the `Type` attribute is specified for the root `policy` element, the policy document can be readily acknowledged as a TRtD or TReD by processing the content of this attribute.

As the recursive looping of `<wsp:Policy>` element is allowed in itself, each assertion can also be identified distinctly as a requirement or resource in a policy document. Consider the example below:

```
<wsp: Policy>
<wsp:All>
<wsp:Policy wsp:Type= "Requirement">
<!--assertion 1-->
</wsp:Policy>
<wsp:Policy wsp:Type= "Resource">
<!--assertion 2-->
</wsp:Policy>
</wsp: All>
</wsp: Policy>
```

This policy document contains two assertions, where one dictates a trust requirement and the other a trust resource. This structuring allows the inclusion of requirements and resources in a single document with the ability to identify them distinctly.

Identifying TRtD and TReD subdocuments

When trust requirements and resources are specified as a sequence of TRtDs and TReDs, each document in the sequence is referred as a subdocument. When TRtDs and TReDs are specified as subdocuments, it is necessary to identify each uniquely, as well as to indicate whether a given document is a subdocument, if so required.

In WS-Policy each policy can be uniquely identified by using `xml:base` and `wsu:ID` attributes of the `<wsp:Policy>` element. Given below is *Policy 3*, which uses these elements.

```
<wsp:Policy     xml:base=     http://creditservice.com/policies
wsu:Id= "TRtD1">
<wsp:All>
<!-- Assertion 1-->
<!-- Assertion 2-->
</wsp:All>
</wsp:Policy>
```

The above policy is identified by the URI http://creditservice/policies#$TRtD_1$.

Consider another service policy, *Policy 4*, specified as a separate document and identified by the URI http://creditservice.com/policies#$TRtD_2$.

```
<wsp:Policy xml:base= http://creditservice.com/policies
wsu:Id= "TRtD2">
<wsp:All>
<!-- Assertion 3-->
<!-- Assertion 4-->
</wsp:All>
</wsp:Policy>
```

Consider an example where the TRtD of a service consists of the above two TRtDs namely $TRtD_1$ (*Policy 3*) and $TRtD_2$ (*Policy 4*). Among these documents consider that $TRtD_1$ is revealed in the UDDI while $TRtD_2$ is revealed later on in the negotiation. A negotiating entity that accesses $TRtD_1$ should be informed that the list of the trust requirements might not end with $TRtD_1$ and further trust requirements may be revealed by the service later in the negotiation.

Adding a Boolean attribute `/wsp:policy/@Sequence` could serve the purpose of indicating a document as a part of sequence of documents. When given the value `true`, it indicates that the given document is part of a sequence of documents, which

means further documents can be expected. When given the value `false`, it indicates that it is the sole TRtD or TReD document. Based on this discussion, the above TRtDs can be modified to indicate that they are part of a sequence of documents:

```
<wsp:Policy     xml:base=     http://creditservice.com/policies
wsu:Id= "TRtD1" wsp:Sequence=true >
<wsp:All>
<!-- Assertion 1-->
<!-- Assertion 2-->
</wsp:All>
</wsp:Policy>

<wsp:Policy xml:base= http://creditservice.com/policies
wsu:Id= "TRtD2" wsp:Sequence=true >
<wsp:All>
<!-- Assertion 3-->
<!-- Assertion 4-->
</wsp:All>
</wsp:Policy>
```

5.5 Overview of WS-Trust specification

The Web Services Trust (WS-Trust) is an XML-based specification for facilitating the establishment of trust in the *Web Service trust model*. In this model, a service provider has a set of requirements that need to be satisfied by the requestor in order to utilise the service. These requirements are stated in the policy (using WS-Policy) document of the service provider. Based on them, the service requestor may exchange credentials with the provider to prove that they satisfy the requirements. As part of stating the requirements, the service provider can also state which credential issuers it trusts to issue credentials. The service requestor can therefore request a credential at the specified issuer and the credential issuer can issue the requested credentials. In this trust model, the credential issuer is assumed to be a Web service and it may, in turn, have its own set of requirements to issue credentials to the service requestor. *Figure 5-4* below illustrates an abstract representation of this trust model. The discussion in this section is taken largely from [6].

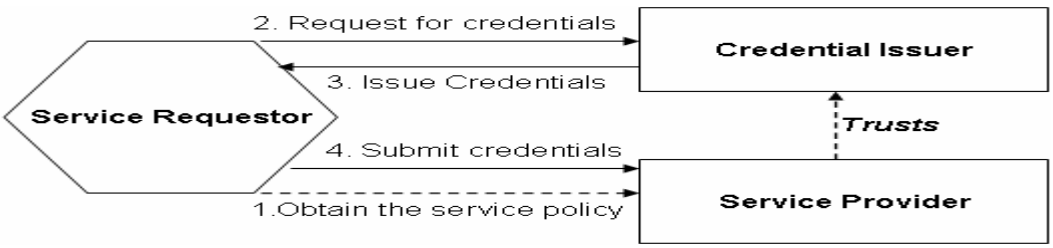

Figure 5-4: WS-Trust trust model

In the trust model the request for credentials at the credential issuer can be of different types. The four request types considered in the trust model are *issue*, *validate*, *renew* and *cancel* - with obvious meanings. Based on the service provider's trust implementation system, certain requests can be made directly to the credential issuer. For instance, when the service requestor submits a specific credential, the service provider can request the credential issuer to validate it. The credential issuer may respond directly to the service provider. It is also possible for the credential issuer to request other credential issuers to validate the credential on behalf of the service provider. In this trust model, trust is established between the service provider and requestor through an exchange of credentials issued by trusted issuers(s).

The WS-Trust specification provides constructs for credential requests and responses for the request types mentioned above. As part of the process of establishing trust, a number of credential requests and responses may occur between the service requestor and provider. As discussed above, there can also be credential requests and responses between the service provider and credential issuer, and between service requestor and credential issuer. Thus, in general, WS-Trust can be summarised as a specification that provides constructs for credential requests and responses that can be incorporated in the SOAP messages as a step towards establishing trust within Web services.

In WS-Trust, the `wst:RequestSecurityToken>` element is used to specify the request of credentials and a response to this request is specified in the element `<wst:RequestSecurityTokenResponse>`. Irrespective of the type of request or response, these elements are used as root elements, and the details of the request or response are then specified in the respective child elements. Given below is a simple credential request using WS-Trust, which is incorporated into the SOAP message structure.

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wst:RequestSecurityToken>
<wst:TokenType>
http://example.org/mySpecialToken
</wst:TokenType>
<wst:RequestType>
http://schemas.xmlsoap.org/ws/2005/02/trust/issue
</wst:RequestType>
</wst:RequestSecurityToken>
```

```
</S11:Body>
</S11:Envelope>
```

In this example, the prefixes `S11` and `wst` refer to the SOAP and WS-Trust specification namespaces http://schemas.xmlsoap.org/soap/envelope and http://schemas.xmlsoap.org/ws/2005/02/trust.

As shown in the above example, `<wst:TokenType>` is used to specify the type of credential requested. Examples of credential types in the security context are X.509 certificate, Kerberos ticket, any standard security credentials and service-defined credentials. The element `<wst:RequestType>` is used to specify the request type of the message. As explained before, there are different types of requests and in this example the request is to *issue* a token of type that is specified by the URI `http://example.org/mySpecialToken`. A possible response to this request can be represented by using the following SOAP message.

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wst:RequestSecurityTokenResponse>
<wst:TokenType>
http://example.org/mySpecialToken
</wst:TokenType>
<wst:RequestedSecurityToken>
<xyz:Token id=" ">
<xyz:issuer> </xyz:issuer>
.
.
</xyz:Token>
</wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
</S11:Body>
</S11:Envelope>
```

As illustrated in the example, the response is specified in the `<wst:RequestSecurityTokenResponse>` element, while its child element `<wst:TokenType>` specifies the type of the credential returned. The requested credential is returned in the element `<wst:RequestedSecurityToken>`, where it can either include the credential itself or a reference to an URI where the credential can be obtained. In this example a fictitious partial XML credential is included just to demonstrate how this can be done.

The above examples illustrate only the basic constructs required for a request credential and response message. The WS-Trust provides constructs for specifying more detailed requests and responses. For example, the specification provides constructs to support more than one credential request in a single SOAP envelope. It is also able to specify the lifetimes of the credentials. In an issue request, the service requestor may use these constructs to specify the desired lifetime of the credential. Some of these specialised elements are discussed further in the analysis of the specification.

In WS-Trust, credentials represent and convey security aspects and the purpose of credential exchange between the requestor and the provider is almost exclusively stated as to secure the communication among them. The WS-Trust specification extends WS-Security [60] to support the issuance and distribution of security credentials to establish trust for various security purposes (e.g. authentication, securing the confidentiality of SOAP messages). As was discussed before, there are other trust contexts that should also be included in generic trust negotiation. In the following section, the extension and usage of WS-Trust to a generic trust negotiation is discussed.

5.6 An analysis of WS-Trust in a general trust negotiation process

With respect to *Figure 5-4*, the focus of this dissertation is on the trust negotiation between the service provider and the requestor. The process of the requestor requesting for credentials from the credential issuer and how the service provider came to trust the credential issuer does not constitute the focus of this study. In order to successfully carry out trust negotiation, these aspects are necessary, but to focus on the exchange of credentials between the service provider and requestor, it is assumed that these aspects are already in place or can be made available. In *Figure 5-5* below, the section identified by the ellipse illustrates the focus of general trust negotiation and hence WS-Trust specification is examined to use in this negotiation.

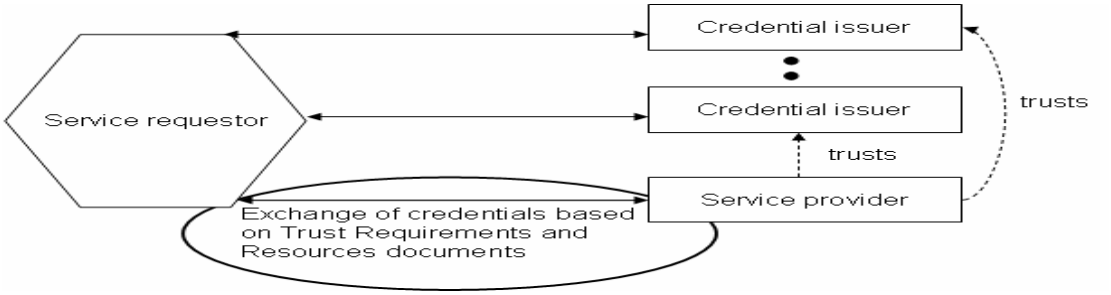

Figure 5-5: Illustration of the focus of this study with respect to the WS-Trust trust model

In general, a generic trust negotiation centers on the exchange of the TRtDs, TReDs and digital credentials. Below are a few illustrations of WS-Trust in generic trust negotiation. These illustrations are not intended to be complete, but to demonstrate the possibility of using WS-Trust in such a negotiation.

Requesting and receiving diverse credentials

In WS-Trust, the `<wst:RequestSecurityToken>` element is used to specify the credential request. The credential and request types are specified in its child elements `<wst:TokenType>` and `<wst:RequestType>` respectively [6]. The `<wst:TokenType>` element can refer to credentials of any XML vocabulary, which provides the flexibility to include any predefined credential in the request message. If vocabularies are defined as explained in Section 5.4, then they can be referred to in the credential request messages. The `<wst:RequestType>` element is used to specify the type of credential request. In addition to the request types (issue, validate, renew and cancel) specified in WS-Trust, more request types can be added to the vocabulary, if required [6]. For the credential exchange in a trust negotiation, one of the possible request types is to *reveal* the credential to the negotiating entity. This kind of request and its corresponding response do not precisely fit into the request types detailed in the WS-Trust specification, which suggests a need to add such request types to support generic trust negotiation.

The `<wst:RequestSecurityTokenResponse>` element is used to specify the response to a credential request. The `<wst:RequestedSecurityToken>` element, which is its child element, can contain the credential itself if it is an XML-based credential or a reference to the credential in terms of where it can be accessed [6]. For instance, if the credential is not in XML format, then it can be sent as an

attachment to the SOAP message [11]. A reference to the credential is then included in the <wst:RequestedSecurityToken> element of the response message.

To reference the TRtD for which the credentials are submitted

When each TRtD has been uniquely identified, the negotiating party should be able to include a reference to the TRtD for which the credentials are submitted in the credential submission message.

In WS-Trust specification an attribute named Context is defined, which is used to correlate requests and responses. This attribute is defined for both <RequestSecurityToken> and <ResponseSecurityTokenRequest> elements [6]. When credentials are revealed in trust negotiation, this element can be used to reference the TRtD that requests specific credentials.

The example given below contains a TRtD that is identified by the URI http://creditservice/policies#TRtD$_1$.

```
<wsp:Policy wsp:Type= "Requirement" xml:base=
http://creditservice.com/policies wsu:Id= "TRtD1">
<wsp:All>
<!-- Assertion 1-->
<!-- Assertion 2-->
</wsp:All>
</wsp:Policy>
```

Based on this policy - if the negotiating entity is providing credentials as specified in Assertion 1 and 2 of the policy, the policy identification URI can be included in the Context attribute of the response message. The following XML fragment illustrates a possible response message.

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wst:RequestSecurityTokenResponse Context="
http://creditservice/policies#TRtD1">
<wst:TokenType>…</wst:TokenType>
<wst:RequestedSecurityToken>…</wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
</S11:Body>
</S11:Envelope>
```

The semantics of this message is that the credential or the reference to the credential included in the message is based on the trust requirements specified in TRtD, which is identified by the content of the `Context` attribute.

To combine a credential request with a credential submission

In trust negotiation, a negotiating entity may request for credentials of the service provider as stated in its TReD, along with its submission of credentials to the service provider. Using WS-Trust this can be achieved by combining the `<RequestSecurityToken>` and `<ResponseSecurityTokenRequest>` elements in the SOAP message [6].

Consider a TRtD and a TReD:

```
<wsp:Policy wsp:Type= “Requirement” xml:base=
http://creditservice.com/policies wsu:Id= “TRtD1”>
<wsp:All>
<!-- Assertion 1-->
</wsp:All>
</wsp:Policy>

<wsp:Policy wsp:Type= “Resource” xml:base=
http://creditservice.com/policies wsu:Id= “TReD1”>
<wsp:All>
<!-- Assertion 1-->
</wsp:All>
</wsp:Policy>
```

Based on these documents, a service requestor can submit the credential specified in the `Assertion 1` of TRtD and request for the credential in the TReD. The following fragment of a SOAP message illustrates how this can be achieved in a single SOAP message structure.

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wst:RequestSecurityTokenResponse Context=”
http://creditservice/policies#TRtD1”>
<wst:TokenType>…</wst:TokenType>
<wst:RequestedSecurityToken>…</wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>

<wst:RequestSecurityToken Context= ”
http://creditservice/policies#TReD1”>
<wst:TokenType>…</wst:TokenType>
<wst:RequestType>..</wst:RequestType>
</wst:RequestSecurityToken>
```

```
</S11:Body>
</S11:Envelope>
```

The semantics of this message is that it contains the credential that is specified in the TRtD along with the request for the credential that is specified in the TReD, which is identified by the URI http://creditservice/policies#TReD$_1$.

To send TRtDs and TReDs in a negotiation

Provision should be made to exchange TRtDs and TReDs in trust negotiation.

The WS-Trust specification defines mechanisms to attach policies in the <RequestSecurityToken> element. In this specification, the aim of attaching a policy in a credential request is to refer to a specific policy that indicates the desired features of the requested credential. For example, the following partial SOAP message illustrates how this is achieved [6].

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wst:RequestSecurityToken>
<wst:TokenType>
http://example.org/mySpecialToken
</wst:TokenType>
<wst:RequestType>
http://schemas.xmlsoap.org/ws/2005/02/trust/issue
</wst:RequestType>
<wsp:Policy>
<!—Assertion 1 -->
</wsp:Policy>
</wst:RequestSecurityToken>
</S11:Body>
</S11:Envelope>
```

This message requests a token of type http://example.org/mySpecialToken, where the desired settings for the requested token is provided in the policy. This technique is significant when a credential can be issued with different settings. Hence the service providers can specify these settings in their policies so that the service requestors can refer to the specific policies when requesting for a credential at the credential issuer.

In the generic trust negotiation process, the service provider may send TRtDs to the service requestor. Here policies outline the credentials that need to be provided by the

service requestor. This can be achieved by including policies using <wsp:Policy> element in the SOAP Body. The SOAP message given below illustrates how this can be achieved.

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wsp:Policy wsp:Type= “Requirement” xml:base=
http://creditservice.com/policies wsu:Id= “TRtD1”>
<wsp:All>
<!-- Assertion 1-->
<!—Assertion 2-->
</wsp:All>
</wsp:Policy>
</S11:Body>
</S11:Envelope>
```

This message expresses a policy, which summarises the trust requirements that needs to be satisfied by the requestor. The trust requirements are specified in the assertions and the policy is uniquely identified by the URI http://creditservice.com/policies#TRtD1. This URI can be included in the Context attribute of the <wst:ResponseSecurityTokenRequest> element in the corresponding response message.

Multiple credential submissions in a single response message

When a TRtD states the submission of more than one credential the service requestor should be able to provide the collection of credentials or credential references in the response message. In WS-Trust specification, the element <wst:RequestSecurityTokenResponseCollection> is used to group different credentials into a single response. This element contains more than one <wst:RequestSecurityTokenResponse> child element, each containing a credential or credential reference. The following partial SOAP message illustrates the structure of the <wst:RequestSecurityTokenResponseCollection> element [6].

```
<S11:Envelope>
<S11:Header> </S11:Header>
<S11:Body>
<wst:RequestSecurityTokenResponseCollection>

<wst:RequestSecurityTokenResponse>
<wst:TokenType>…</wst:TokenType>
```

```
<wst:RequestedSecurityToken>…</wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>

<wst:RequestSecurityTokenResponse>
<wst:TokenType>…</wst:TokenType>
<wst:RequestedSecurityToken>…</wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
...
<wst:RequestSecurityTokenResponseCollection>

</S11:Body>
</S11:Envelope>
```

Frequency of credential submission

The credentials may have validity time frames, which might drive repeated credential exchanges between negotiating entities. Based on the trust implementation system and the type of credentials exchanged, this resubmission may trigger a renewal of trust relationships.

In WS-Trust specification, `<wst:Lifetime>` is the construct that is used to specify a time period. This element has two subelements named `<wsu:Created>` and `<wsu: Expires>`, which are defined in the utility schema of Web Services. The element `<wst:Lifetime>` can be used in both credential request and response. When used in `<wst:RequestSecurityToken>`, it indicates the validity time period requested by the requestor to the credential issuer and when used in `<wst:ResponseSecurityTokenRequest>`, it indicates the validity of the credential issued by the issuer. It is not necessary for the lifetime requested to be the same as the lifetime of the issued credential [6].

In trust negotiations, the validity of the credentials exchanged will be determined by the lifetime specified in the credential itself, if any. If the lifetime is not specified by the credential then it should be possible to validate the lifetime of the credential at the appropriate credential issuer.

Error handling

Errors may occur during the process of negotiating trust, and there should be techniques to handle these errors appropriately. The errors can be due to incorrect credentials, credential formats, unknown credentials and invalid credentials. As part

of the error-handling technique, the service provider and requestor may inform each other about the errors that occurred.

The WS-Trust defines certain fault codes that can be included in the SOAP messages to specify errors. Two examples of such fault codes are `wst:InvalidRequest` and `wst:RenewNeeded`, which refer to an invalid request and a renewed credential request respectively [6] . In addition to these error messages, elements can be added to convey details of the errors and, if possible, ways of recovering from the error. In trust negotiation, it is also required to inform the negotiating party about the result of negotiation, whether success or failure of negotiation and the implications of the negotiation success or failure.

5.7 Conclusion

In the Web Services framework WS-Trust and WS-Policy closely match the requirements of a negotiation protocol and those for the specification of trust requirements and resources respectively. These specifications mainly concern the specification and exchange of security credentials. In order to utilise them in generic trust negotiations, several extensions are suggested. These extensions are not exhaustive, but they demonstrate how the specifications can be generalised.

CHAPTER 6
A TRUST NEGOTIATION PROCEDURE FOR COMPOSITE SERVICES

6.1 Introduction

A trust negotiation procedure for composite services should deal with two viewpoints - the unified outlook of the service and the non-unified view - because of the possible distinct trust negotiations with its constituents. A possible way to handle this is to reconcile the trust requirements and resources of the constituents with those of the composite service, so as to provide a unified outlook to trust requirements and resources of the composite service. These trust requirements and resources can then be utilised in negotiations with the service.

The trust requirements and resources of a composite service should encompass each of its constituents' trust requirements and resources. This can be done by reconciling the TRtDs and TReDs of the constituents into those of the composite service. When a unified TRtD and TReD is available, then the composite service should have a mechanism to present and manage these documents by a set of services referred to as *trust-negotiating nodes*. By including and coordinating these nodes, a composite service can provide a unified approach towards trust negotiation with its users.

The aim of this chapter is to develop a negotiation procedure that works in composite Web services by utilising trust requirements and resources of composite services. In Section 6.2, the trust requirements and resources of composite services are discussed. Two techniques for reconciling the TRtDs and TReDs of the constituents with those of a composite service are discussed in Section 6.3. Section 6.4 deals with the presentation of the reconciled TRtDs and TReDs by the trust-negotiating nodes, and Section 6.5 with the association between the services and the trust-negotiating nodes. The coordination of the trust-negotiating nodes by the trust-negotiation coordinator is discussed in Section 6.6, while Section 6.7 describes how a user entity can carry out trust negotiations with an elementary and composite service by using the trust negotiating-nodes and trust-negotiation coordinator. The extent to which this negotiation procedure handles trust context, anonymity, dynamicity and complexity is discussed in Section 6.8. The chapter is concluded in Section 6.9.

6.2 Trust requirements and resources of composite services

A constituent service can have trust requirements for directly and indirectly interacting entities. These requirements can be the same, with or without some overlap, or there may not be any trust requirements for the indirectly interacting entity. The trust requirements of a composite service should include the trust requirements of the constituents for indirectly interacting entities, except for the constituent that interacts directly with the user. The TRtD of the composite service should therefore contain the trust requirements of the constituent services as explained above.

On the other hand, the complete trust resources of each constituent should be integrated or unified with the trust resources of the composite service and the TReD of the composite service should contain the trust resources of its constituent services. If the trust resources are sensitive, then certain trust requirements can be attached to these resources in such a way that the trust resources are revealed only when the attached trust requirements are satisfied. When specifying the trust resources of a composite service, these attached trust requirements should also be included.

6.3 Reconciling the TRtDs and TReDs of constituent services

In general, the TRtDs and TReDs of the constituent services can be integrated in two different ways. In the first form or technique of unification, the documents are unified as a sequence of subdocuments with no modification in the content and structuring of the individual TRtD and TReD of the constituent services. The second technique involves an analysis and combination of the contents of the TReD and TRtD of each constituent to create a unified TRtD and TReD, which may contain a sequence of subdocuments.

The first technique is the simplest, but may not be the best way of specifying trust requirements and resources of a composite service. Some requirements may be redundant and the same trust requirement may be specified in all the TRtDs. The second technique may avoid redundancy by analysing the documents of the constituents. For example, if all the documents specify the same trust requirement, then the TRtD of the composite service can specify this requirement only once. The

effort required to implement the second technique is obviously greater than to implement the first technique.

Unification Technique 1

In this technique the TRtD and TReD of the composite service consist of a sequence of subdocuments where each document represents the TRtD or TReD of a constituent service. For example, the TRtD of the composite service given in *Figure 6-1* will consist of two TRtDs where these documents are `TRtD1` (for a directly interacting entity) and `TRtD2` (for an indirectly interacting entity). Likewise, the TReD of the composite service is also presented as two sub-TReDs.

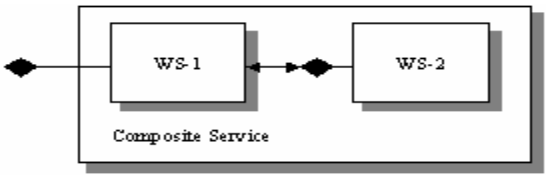

Figure 6-1: A composite service with two constituent services

Consider the following WS-Policy documents, which represent the `TRtD1` and `TRtD2`.

```
<wsp:Policy wsp:Type= "Requirement" xml:base=
http://service1.com/policies wsu:Id= "TRtD1">
<wsp:All>
<!-- Assertion 1-->
<!-- Assertion 2-->
<!-- Assertion 3-->
</wsp:All>
</wsp:Policy>

<wsp:Policy wsp:Type= "Requirement" xml:base=
http://service2.com/policies wsu:Id= "TRtD2">
<!-- Assertion 1-->
</wsp:Policy>
```

Using the above technique, the Web service `WS-1` can generate the TRtD for the composite service as given below.

```
<wsp:Policy wsp:Type= "Requirement" xml:base=
http://service1.com/policies wsu:Id= "TRtD1" wsp:Sequence=true>
<wsp:All>
<!-- Assertion 1-->
<!-- Assertion 2-->
<!-- Assertion 3-->
```

```
</wsp:All>
</wsp:Policy>

<wsp:Policy wsp:Type= “Requirement” xml:base=
http://service2.com/policies wsu:Id= “TRtD2” wsp:Sequence=true>
<!-- Assertion 1-->
</wsp:Policy>
```

Here the policies identified by the URI http://service1.com/policies/#$TRtD_1$ and http://service2.com/policies/#$TRtD_2$ will be presented to the user entity as a sequence of TRtDs. These TRtDs are described as a constituent in a sequence of documents by setting the XML element `wsp:sequence` (refer to Chapter 5) to `true`; thus indicating that further TRtDs can be expected in the trust negotiation.

Unification Technique 2

In the second unification technique, the integration of TRtDs is based on the content of assertion to specify each trust requirement and resource. This means the assertions are analysed to create a unified TRtD and TReD. Even in this form of unification the resultant TRtD and TReD can be a sequence of documents, where the sequencing can be based on criteria such as the context or sensitivity, of credentials specified as using assertions, rather than on the elementary service to which the document belongs.

Consider the example where the policies identified by http://service1.com/policies/# $TRtD_1$ and http://service1.com/policies/# $TRtD_2$ are merged using this technique. Given below is a policy document that illustrates such a unified document.

```
<wsp:Policy wsp:Type= “Requirement”>
<wsp:All>

<wsp:Policy xml:base= http://service1.com/polcies  wsu:Id=
“TRtD1”>
<wsp:All>
<!-- Assertion 1-->
<!-- Assertion 2-->
<!-- Assertion 3-->
</wsp:All>

</wsp:Policy>
<wsp:Policy xml:base= http://service2.com/polcies wsu:Id=
“TRtD2”>
<!-- Assertion 1-->
</wsp:Policy>
```

```
<wsp:All>
</wsp:Policy>
```

The XML fragment specifies two distinct policies in a single document. The specification of the WS-Policy element `<wsp: All>`, states that both the policies need to be satisfied.

Both these unification techniques are useful as demonstrated in the following sections.

6.4 The presentation of the unified TRtDs and TReDs by the trust-negotiating nodes

Once the TRtDs and TReDs are reconciled, a subset of the constituents can present these documents to the user. The services are known as trust-negotiating nodes. For instance, in the composite service illustrated in *Figure 6-1*, if the service `WS-2` allows the integration of its TRtD (for an indirectly interacting entity) with that of `WS-1`, then the unified TRtD can be presented by the service `WS-1`. Here `WS-1` becomes the trust-negotiating node. The service `WS-2` may not be involved in presenting its trust requirements or resources, but it can still be responsible for other activities of trust establishment. For example, it can still be responsible for evaluating the credentials submitted by the user entity stated in the trust requirements.

It is possible that not all the constituents allow unification of their trust requirements and resources documents with those of other services, when they are utilised in a composite service. One of the reasons may be the partial delegation of the negotiation process to the other services. In this case, these constituent services will become independent trust-negotiating nodes. A trust-negotiating node can therefore be responsible for presenting TRtDs and TReDs of itself or on behalf of a set of services.

It emerges from this discussion that, in a composite service, there can be a number of trust-negotiating nodes, which are determined by the following criteria:

Form 1:

In this form, the documents are not reconciled and every constituent is responsible for negotiating trust with the user entity individually. Hence each of the constituent services will act as a trust-negotiating node.

Form 2:

The subset of constituents in the composite service collaborates to create a unified TRtD and TReD for the specific subset. Thus, only a subset of the constituents in the composite service constitutes trust-negotiating nodes.

Form 3:

This form is a type of Form 2 above, where the TRtDs and TReDs of all the constituent services are reconciled into one, which is managed by the root element (constituent that interacts directly with the user) in the composite service. Here there is only one trust-negotiating node for the composite service.

When there is more than one trust-negotiating node (*Forms 1* and *2*), the root element of the composite service takes the role of coordinator in order to coordinate the trust-negotiating nodes. Each of the forms represents a scenario possible in a composite service, as will be demonstrated below.

Consider the following composite service (*Figure 6-2*) to demonstrate the above-discussed forms:

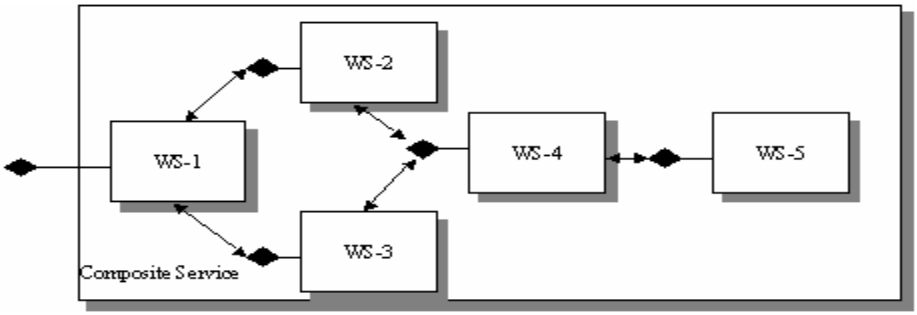

Figure 6-2: A composite service with five constituent services

The figure illustrates a composite service, where the service offered by the composite service is the combination of services offered by Web services `WS-1`, `WS-2`, `WS-3`, `WS-4` and `WS-5`.

Using *form 1*, `WS-1`, `WS-2`, `WS-3`, `WS-4`and `WS-5` are trust-negotiating nodes, which means that every node will present its own TRtD and TReD, and they negotiate directly with the user entity. In *form 2* a subset of the constituent services becomes trust-negotiating nodes. For example, consider `WS-4` as the negotiating node for `WS-4` and `WS-5`. In the same way `WS-1` is the negotiating node for `WS-1`, `WS-2` and

WS-3. Thus there are the two negotiating nodes namely WS-1 and WS-4 in the composite service. In the above two forms WS-1 becomes the trust-negotiation coordinator. Using *form 3* the service WS-1 becomes the sole trust-negotiating node. Here WS-1 will be responsible for presenting the reconciled TRtD and TReD of the complete composite service.

6.5 The association between the services and the trust-negotiating node

The reconciliation of trust requirements and resources documents allows the partial delegation of the negotiation process to the other services. However, for example, a constituent service entity can still be responsible for evaluating credentials submitted by the user entity even if it is not a trust-negotiating node. This is based on the extent to which a service entity delegates its trust negotiation aspects to the trust-negotiating node.

Consider the composite service in *Figure 6-2*, where WS-4 is a trust-negotiating node for WS-5. Here WS-4 has a unified TRtD and TReD consisting of the TRtD and TReD of WS-4 and WS-5. Based on the trust relationship between these two services, the service WS-5 can delegate more aspects of negotiation to WS-4. For example, to handle credential submissions by the user entity in response to a requirement stated in the TRtD of WS-5, the service WS-4 can perform any of the following:

- It can receive the credential on behalf of WS-5 and forward it to WS-5.
- It can specify in the reconciled TRtD that the credential needs to be submitted directly to WS-5.
- It can receive and evaluate the credential on behalf of WS-5 to determine its trustworthiness. In this case, the service WS-4 has to use the policies of WS-5 to evaluate trustworthiness. In this case, WS-5 is delegating all its tasks of trust negotiation to WS-4.
- It can receive and evaluate the credential for WS-5 and can subsequently provide a recommendation regarding the trustworthiness of the credential to the service WS-5.

These cases are not exhaustive, but they summarise some of the obvious options.

Likewise, to facilitate the access of trust resources of WS-5, WS-4 can perform any of the following:

- Forward the trust resource request to WS-5 and WS-5 provides the resource to WS-4. Then the user entity receives the resource from WS-4.
- Forward the trust resource request to WS-5 and WS-5 provides it to the user entity directly.
- Request the user entity to make the trust resource request directly to WS-5 and receive the trust resource directly from WS-5.

Here it is assumed that when a request is made for a sensitive trust resource the trust-negotiating node has presented its attached TRtD.

6.6 The role and functioning of the trust-negotiation coordinator

The trust-negotiation coordinator bases its coordination activities on the list of trust-negotiating nodes in the composite service.

When the user entity initiates a negotiation, the coordinator selects the first trust-negotiating node in the composite service. It forwards the relevant information to the user entity so that it can initiate a trust negotiation with the trust-negotiating node. If the negotiation is successful, then the negotiation node can provide a token that represents the status of the negotiation. The user entity presents this token to the trust-negotiation coordinator. The coordinator analyses the token and if it represents a successful negotiation with the trust-negotiating node, it reveals the second negotiating node. This process is carried out until the negotiation is complete with all the negotiating nodes, or until the negotiation fails - either because the user was unable to create a feasible trust relationship or because the negotiating node could not make a trust relationship with the user entity. The following step-by-step procedure summarises the tasks that a trust-negotiation coordinator fulfils in trust negotiation.

Step 1: Initiates trust negotiation.

Step 2: Obtains the trust-negotiation node list.

Step 3: Reveals the trust-negotiating node to the user entity.

Step 4: Waits for a response from the user entity regarding the trust negotiation with the negotiating node. The wait will lapse after a predetermined time period if a response is not obtained in the time interval.

Step 5: Analyses the token provided by the negotiating node to the user entity in the response message. If the token informs a failure in negotiation, the negotiation ends. If the token informs a success in negotiation, then trust-negotiation coordinator proceeds to *step 6*.

Step 6: Analyses the user's intention to continue the trust negotiation. If `true`, repeats s*teps 3-6* for each trust-negotiation node or `else` the negotiation ends.

6.7 Trust negotiations of the user entity with a service

In this section, the unified negotiation structure discussed in the previous section is used to examine how the user entity can negotiate with a service.

Trust negotiations with an elementary service

In order to establish trust with the service, a user can request the credentials that are listed as trust resources in the TReD of the service.

The following procedure illustrates how a user entity negotiates trust with an elementary service. The negotiation procedure is carried out with the assumption that the user entity has determined the context(s) for establishing trust, the information required to compute trust and the resources from which the information can be gathered (as discussed in Chapter 2).

Step 1: Select the trust resources required from the service TReD.

Step 2: Request the trust resources.

Step 3: Based on the requested trust resources, satisfy the TRtDs to obtain the trust resources.

Step 4: Obtain the trust resources.

In order to use a negotiation strategy, the user and service can first exchange appropriate policies regarding trust resources and requirements. This is then followed by the exchange of credentials that represent trust resources and requirements.

Trust negotiations with a composite service

One of the challenges in this scenario is to determine the services from which the information needs to be collected for a specific context and also the trust-negotiating nodes that are negotiating on behalf of these services.

Consider the composite service illustrated in *Figure 6-3* below:

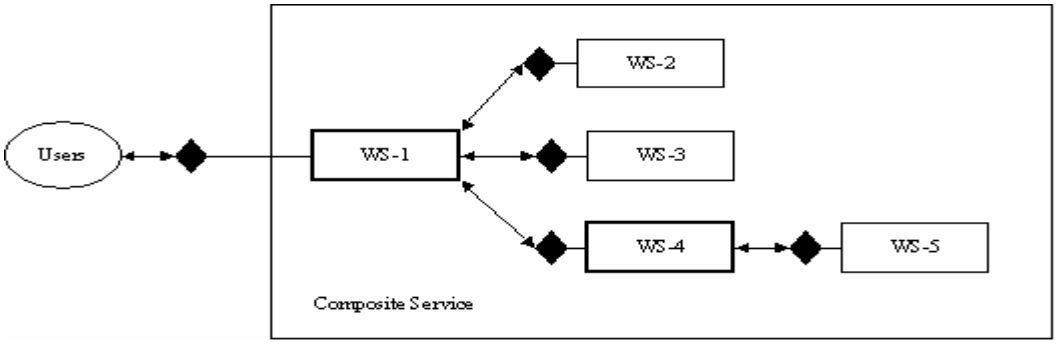

Figure 6-3: A composite service with five constituent services

The service offered by `WS-1` is a combination of the services offered by `WS-1`, `WS-2`, `WS-3`, `WS-4` and `WS-5`. Consider that `WS-1` and `WS-4` are trust-negotiating nodes for the service (as depicted using thick boxes), where `WS-4` negotiates for `WS-4` and `WS-5`, and `WS-1` negotiates for `WS-1`, `WS-2` and `WS-3`.

In order to carry out a negotiation, the user has to determine the services from which the information needs to be collected. Based on the trust context, information can be collected from a subset of the services or from all the services. The first step for the user entity is to determine the services from which the trust resources have to be collected. The second step is to further resolve the trust-negotiating nodes of the respective services. For example, if the information needs to be collected from `WS-5`, then the user has to negotiate with the negotiating node `WS-4`.

One way to determine the respective services and their trust-negotiating nodes is to find some information about the structure of the service. This can be achieved only if the service reveals its structure to the user entities. Such a structure can summarise the following information:

- The contributing services in the composite service.
- The general structure of the composite service, which shows the directly interacting and indirectly interacting entities of every service.

- The role of each service.
- The general invocation and data flow between these services.
- The list of trust-negotiating nodes.
- The trust-negotiation coordinator of the service.

The description can incorporate some of the aspects specified in the BPEL4WS (See Chapter 3) specification of the composite service, along with aspects such as the trust-negotiating nodes and coordinator. When and how this service structure is revealed to the user entity may vary from one service to another. In one form, the complete service structure may be published in the UDDI. In another form, some information regarding the service structure is published in the UDDI and the complete service structure is revealed only when it is explicitly requested by the user entity. The decision to choose one of these forms can be based on the size and complexity of the composite service, as this will influence the size of the service description document. Moreover, not all user entities require all the details of a service structure.

The following procedure illustrates how a user entity negotiates trust with a composite service. The negotiation procedure is carried out with the assumption that the user entity has determined the context(s) for establishing trust, the information required to compute trust and the services from which the information can be gathered (as discussed in Chapter 2).

Step 1: Select the trust resources required from the TReD.

Step 2: Request for the trust resources.

Step 3: Satisfy the TRtDs to obtain the requested trust resources.

Step 4: Obtain trust resources.

Repeat steps 1 to 4 for all the trust-negotiating nodes that negotiate on behalf of the services from which trust resources need to be gathered.

This procedure is similar to that of negotiating with an elementary service, with the exception that the procedure may be repeated, based on the relevant number of trust-negotiating nodes.

In order to use a negotiation strategy, the user and trust-negotiating node can first exchange appropriate policies regarding trust resources and requirements. This is then

followed by the exchange of credentials that represent trust resources and requirements.

6.8 Handling issues of anonymity, dynamicity, trust context and interlinks in composite services

In this section the previously discussed trust negotiation process is analysed to examine how it deals with the issues of anonymity, dynamicity, trust context and complexity of a composite service (see Chapter 3). This discussion focuses on how the negotiation procedure helps the user entity to handle these issues when establishing trust relationships with the service.

Handling anonymity in a composite service

Certain aspects of the service structure are made available to the user entity in the trust negotiation procedure. This service structure describes the service in terms of the constituent services, their roles and the trust-negotiating nodes, along with other service aspects. When the service structure is made available to the user entity, then the abstract structure of the service becomes transparent to the user entity.

Handling trust contexts

In the negotiation procedure, the trust contexts are managed through the use of uniquely defined vocabularies for various contexts and the associated credentials. The request by a user entity for a credential is based on the trust resources outlined in the TReD of the service. These trust resources are structured as policy documents, where resources are specified as assertions. The semantics of an assertion is based on the predefined vocabularies.

Handling complexity in a composite service

The service structure provided to the user entity contains the invocation and data flow among the constituent services. By analysing the structure of the service, as well as the flow of data among different services, the user entity can determine the possible interlinks and restrict its trust relationship with the service accordingly. In this negotiation structure the responsibility for analysing possible effects of complexity is therefore transferred to the user entity.

Handling dynamicity in a composite service

The changes in a composite service can be grouped mainly into two categories: one dictates changes in TReDs and TRtDs, and other dictates changes in the overall service structure. For example, change in certifications or change in service guarantees may result in modified trust resources, which may result in changes in the respective TReDs. On the other hand, replacing constituent services with other services or adding new services will result in a modified service structure and the unified TRtD and TReD.

If changes occur in the trust resources, then the user entity must be able to access the modified trust resource either by requesting the resource at every service request or at regular time intervals. Another option is for the service to send the modified trust resource to its user entities. To accomplish this with every trust resource listed in the TReD, the service can list the preferred options to convey the frequency of obtaining the trust resource where the options include the various alternatives indicated above. The user entity may choose one of the alternatives and renew the trust relationships with the service, based on the frequency of acquisition of the trust resources.

If the changes occur in the service constituents, they trigger change in the service structure and in the unified TRtDs and TReDs. Such changes may result in the renewal and re-establishment of trust relationships between services in the composite service. Moreover, they can also result in the re-establishment of trust relationships of user entities with the service. To handle such a change, the services themselves must initially renew or re-establish trust relationships with the new services - along with a reshuffling of the TRtDs, TReDs, service trust-negotiating nodes, even the trust-negotiation coordinator and the service structure. Once this has been achieved, the user entities of the composite service are informed about the change in the service structure and given access to the new service structure. This will allow them to re-establish the trust relationships with the service by using the new structure.

To accomplish the suggested methods to handle dynamicity, further extensions to WS-Policy and WS-Trust specifications are required.

6.9 Conclusion

One possible trust negotiation procedure for composite services, which utilises the unified TRtDs and TReDs of the composite service, was presented in this chapter. The procedure suggests one way of handling the issues of anonymity, complexity, trust contexts and dynamicity.

CHAPTER 7
CONCLUSIONS AND FURTHER WORK

The aim of this study was to analyse the Web Services framework for generic trust negotiations and to suggest a generic trust negotiation procedure that applies to composite services. To achieve this goal, we commenced with the concept of trust and then progressed to the concept of trust negotiation.

The core concept dealt with in this dissertation is trust, for which a solid perspective is presented in Chapter 2. Trust is more than security; it is a generic concept that can be applied in various contexts. Security is just one context among other possible contexts. However, trust is an important concept in various security applications and hence the literature concentrates on it.

In Chapter 2 an abstract trust establishment technique is presented, which has the potential to be used in different contexts and applications (refer to *Figure 2-1*). This technique presupposes some computation scheme that represents trust as a quantified value based on information about an entity. In the trust establishment process, trust negotiation is deployed to obtain information about an entity.

As stated in Chapter 4, the concept of trust negotiation is not novel. It is used widely as an access control mechanism in different systems. However, in the trust negotiation process any trust context can be considered. This makes it possible to view trust negotiation from a general perspective without necessarily binding it to a specific application area.

Although the existing trust negotiation systems are utilised for a specific application, they have various aspects that can be generalised to create a generic trust negotiation system. From the analyses of the existing trust negotiation systems, the essential elements of trust negotiation are presented as digital credentials, policies, negotiation strategy and negotiation protocol. The extent of support for these elements varies in the existing systems. The negotiation procedure utilised in existing systems mainly focuses on the negotiation between two entities, which is not adequate when applied in the composite services created by using the Web Services framework. Trust-X

provides an abstract negotiation procedure for negotiating with composite services, but gives no details about how to carry out such a negotiation.

In order to support generic trust negotiations in Web services, the framework should support certain elements of negotiation such as digital credentials, negotiation protocol and aspects of the policies that will be communicated in a negotiation (see the discussion in Chapter 5). At this stage, digital credentials are least supported in the Web Services framework, whereas policies and negotiation protocol are partially supported in the form of WS-Policy and WS-Trust specifications.

The goals of WS-Policy specification closely match the specification requirements of trust resources and requirements in trust negotiation. However, WS-Policy can be extended to add various aspects to give a complete description of trust requirements and resources. Several of these aspects are pointed out in Chapter 5, along with the possible extensions to WS-Policy.

WS-Trust provides various message structures that can be included in SOAP messages, to support credential exchanges between a service provider, requestor and credential issuer. These message structures can be largely reused for the generic trust negotiation, which primarily occurs between the service provider and the requestor. Additional features have to be included in the WS-Trust in order to use it as a comprehensive negotiation protocol. Several of these aspects are highlighted in Chapter 5.

A trust negotiation procedure suitable for composite services is subsequently presented in Chapter 6. This procedure helps to provide a unified approach towards distinct trust negotiations with constituents, if required. This procedure is one possible way of negotiating with composite services and the details of carrying out such a negotiation are also presented.

The suggested negotiation procedure presents one way of handling issues of anonymity, complexity, dynamicity and trust contexts in trust negotiations with composite services. Suggestions are put forward in handling the issue of dynamicity although it requires additional extensions to WS-Policy and WS-Trust specifications.

In brief, this study has met the objectives of analysing the Web Services framework to support generic trust negotiations and providing a negotiation procedure for carrying out negotiations in composite services.

Further Work

A major phase of work to be done in the future includes providing necessary implementations for the specified negotiation procedure in the Web Services framework, along with an analysis of the complexity and communication effort for this procedure, based on the complexity of the composite service. In order to achieve this, the following aspects should receive attention:

- The possibility of incorporating a digital credential framework for Web Services, which handles issuance, distribution and maintenance of digital credentials. The credentials should be capable of conveying information in any trust context.
- A specification that is an extension of WS-Policy to include all the requirements of a complete trust negotiation language. This specification should also be able to support the complete specification of trust requirements and resources documents.
- A specification that is an extension of WS-Trust, which is a comprehensive trust negotiation protocol for generic trust negotiation.
- Building a trust negotiation system that can be utilised by Web services and that provides all the necessary support for trust negotiations. The feasibility of deploying this system as an independent Web service should also be examined.
- The additional features required for deploying generic trust negotiations in the Web Services framework (e.g. describing `tModel`s for describing negotiations that can be included in the UDDI).
- The specification and presentation of service structures as discussed in Section 6.7 of Chapter 6.
- Vocabularies for various trust contexts and information sources.
- The complete specification of TRtDs and TReDs.
- The implementation of a trust computation scheme based on the concepts given in Section 2.5 of Chapter 2.

BIBLIOGRAPHY

[1] Abdul-Rahman A. and Hailes S. (1997). A Distributed Trust Model. *Proceedings of the 1997 Workshop on new Security Paradigms*. ACM. pp. 48-60.

[2] Abdul-Rahman A. and Hailes S. (2000). Supporting Trust in Virtual Communities. *Proceedings of the 33rd Hawaii International Conference on System Sciences*. Volume 6. IEEE.

[3] Aberer K. and Despotovic Z. (2001). Managing Trust in a Peer-2-Peer Information System. *Proceedings of the Tenth International Conference on Information and Knowledge Management(CIKM'01)*. ACM. pp. 310-317

[4] Adams C. and Boeyen S. (2002). UDDI and WSDL Extensions for Web Services: A Security Framework. *Proceedings of 2002 ACM Workshop on XML Security.* ACM. pp. 30-35.

[5] Alonso G., et al. (2003). Web Services Concepts, Architectures and Applications. *Springer-Verlag*.

[6] Anderson S., et al. (2005). Web Services Trust Language (WS-Trust). Online. 68 pages. http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-trust.pdf. Accessed on 6th May 2005.

[7] Andrews T., et al. (2003). Business Process Execution Language for Web Services Version 1.1. Online. 136 pages. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/. Accessed on19th September 2005.

[8] Au R., Looi M. and Ashley P. (2001). Automated Cross-organizational Trust Establishment on Extranet. *Proceedings of the IEEE Workshop on Information Technology for Virtual Enterprises*. IEEE.

[9] Bajaj S., et al. (2004). WS-Policy. Online. 17 pages. ftp://www6.software.ibm.com/software/developer/library/ws-policy.pdf. Accessed on18th May 2005.

[10] Bajaj S., et al. (2004). WS-PolicyAttachment. Online. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policyattachment.asp. Accessed on 10th October 2005.

[11] Barton J.J., et al. (2000) SOAP Messages With Attachments. Online. http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211. Accessed on 6th May 2005.

[12] Bellwood T., et al. (2004). UDDI Version 3.0.2. Online. http://uddi.org/pubs/uddi_v3.htm. Accessed on19th September 2005.

[13] Benatallah B., et al. (2002). Overview of Some Patterns for Architecting and Managing Composite Web Services. *ACM SIGecom Exchanges*. Volume 3(3) pp. 9-16.

[14] Benatallah B., et al. (2002). Declarative Composition and Peer-to-Peer provisioning of Dynamic Web Services. *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*. IEEE.

[15] Bentallah B., et al. (2004). Abstracting and enforcing web service protocols. *International Journal of Cooperative Information Systems*. Volume 13(4). pp. 413-440.

[16] Bertino E., Ferrari E. and Squicciarini A. (2004). Trust Negotiations: Concepts, Systems, and Languages. *Web Engineering*. IEEE. Volume 6(4) pp.27-34.

[17] Bertino E., Ferrari E. and Squicciarini A. (2003). Trust-X: A Peer-to-Peer Framework for Trust Establishment. *Transactions on Knowledge and Data Engineering. IEEE*. Volume 16 (7). pp. 827-842.

[18] Bertino E., Ferrari E. and Squicciarini A. (2003) X-TNL: An XML based Language for Trust Negotiations. *Proceedings of the. 4th International Workshop on Policies for Distributed Systems and Networks*. IEEE. pp. 81-84.

[19] Bertino E., Ferrari E. and Squicciarini A. (2003) Trust-X: An XML Framework for Trust Negotiations. *Proceedings of the Communications and Multimedia Security (CMS 2003)*. Lecture Notes in Computer Science 2828. Springer-Verlag. pp.146-157.

[20] Beth T., Borcherding M. & Klein B. (1994). Valuation of trust in open networks. *Proceedings of the European Symposium on Research in Computer Security* (*ESORICS*). Lecture Notes in Computer Science 875. Springer-Verlag. pp.3-18.

[21] Blaze M., Feigenbaum J. and Lacy J. (1996). Decentralized Trust Management. *Proceedings of the 1996 IEEE Symposium on Security and Privacy (SP'96)*. IEEE. pp. 164-173.

[22] Blaze M., Feigenbaum J., Ioannidis J. and Keromytis A.D.(1999). The Role of Trust Management in Distributed Systems Security. *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*. Lecture Notes in Computer Science State-of-the-Art series. Springer-Verlag pp. 185 – 210.

[23] Blaze M., et al. (1999). The Keynote Trust Management Systems, Version 2, Internet Engineering Task Force RFC 2704. Online. www.ietf.org/rfc/rfc2693.txt. Accessed on 12th October 2005

[24] Bonatti P. and Samarati P. (2000). Regulating Access and Information Release on the Web. *Proceedings of the 7th ACM conference on Computer and communications security (CCS '00)*. ACM. pp. 134-143.

[25] Bosworth A. (2001). Developing Web Services. *Proceedings of the 17th International Conference on Data Engineering (ICDE '01)*. IEEE.

[26] Christensen E., et al. (2001). Web Services Description Language (WSDL) 1.1. Online. http://www.w3.org/TR/wsdl. Accessed on19th September 2005.

[27] Curbera F., et al. (2002). Unraveling the Web Services Web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet computing*. Volume 6(2). pp. 86-93.

[28] Damiani E., Vimercati S.D.C.D. and Samarati P. (2002). Towards Securing XML Web Services. *Proceedings of the 2002 ACM workshop on XML security*. ACM. pp 90-96.

[29] Della-Libera G., et al. (2002). WS-SecurityPolicy. Online. http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/. Accessed on18th of May 2005.

[30] Dr Holland C.P. and Prof. Geoff Lockett A. (1998). Business Trust and the Formation of Virtual Organizations. *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*. Volume 6. IEEE.

[31] Elfatatry A. and Layzell P. (2002). Software As a Service: A Negotiation Perspective. *Proceedings of the 26th Annual International Computer Software and Applications Conference (COMPSAC'02)*. IEEE.

[32] Ellison C., et al. (1999). SPKI Certificate theory, IETF RFC 2693. Online www.ietf.org/rfc/rfc2693.txt. Accessed on 12th October 2005.

[33] Essin D.J. (1997). Patterns of Trust and Policy. *Proceedings of the 1997 workshop on New security paradigms*. ACM. pp.38-47.

[34] Galli D.L. (2000). Distributed Operating Systems. Prentice Hall.

[35] Gambetta D. (2000). Can we Trust Trust? *Trust: Making and Breaking Cooperative Relations,* University of Oxford.

[36] Gardner T.(2001). An introduction to web services. *Ariadne Issue 29*. Online. 4 pages. http://www.ariadne.ac.uk/issue29/gardner/intro.html. Accessed on 27th February 2003.

[37] Gergic J., et al. (2002). An approach to lightweight deployment of web services. *Proceedings of the 14th International conference on Software engineering and knowledge engineering* (*SEKE '02).*ACM. pp. 635-640.

[38] Grandison T. and Sloman M. (2003). Trust Management Tools for Internet Applications. *Trust Management 2003*. Lecture Notes in Computer Science 2692. Springer-Verlag. pp. 91-107.

[39] Grandison T. and Sloman M. (2000). A survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*. Volume 4(4*)*. Online. 15 pages.

Available at http://www.comsoc.org/pubs/surveys. Accessed on 12th October 2005.

[40] Grandison T. and Sloman M. (2001). SULTAN-A Language for Trust Specification and Analysis. *Proceedings of the 8th Annual Workshop HP OpenView University Association (HP-OVUA)*. pp.1-9.

[41] Gudgin M. (2004). Secure, Reliable, Transacted; Innovation in Web Services Architecture. *Proceedings of the 2004 ACM SIGMOD International conference on Management of data*. ACM. pp. 879 - 880.

[42] Gudgin M., et al. (2003). SOAP Version 1.2 Part 1: Messaging Framework. Online. http://www.w3.org/TR/soap12-part1/. Accessed on 19th September 2005.

[43] Hull R. and Su J. (2004). Tools for Design of Composite Web Services. *Proceedings of the 2004 ACM SIGMOD International conference on Management of data*. ACM. pp. 958 - 961.

[44] IBM Trust Establishment. Online. http://www.alphaworks.ibm.com/tech/trustestablishment. Accessed on 12th October 2005.

[45] Jepsen T. (2001). SOAP Cleans up Interoperability Problems on the Web. *IT Pro*. Volume 3(1). IEEE. pp. 52-55.

[46] Jones S., Wilikens M., Morris P. and Masera M. (2000). Trust Requirements in E-Business. *Communications of the ACM*. Volume 43(12). pp.81-87

[47] Josang A. (1996). The right type of trust for distributed systems. *Proceedings of the 1996 workshop on New security paradigms*. ACM. pp. 119-131.

[48] Kalcklosch R. and Herrmann K. (2003). Statistical Trustability (Conceptual Work). *Trust Management 2003*. Lecture Notes in Computer Science 2692. Springer-Verlag. pp. 271-274.

[49] Kinateder M. and Rothermel K. (2003). Architecture and Algorithms for a Distributed Reputation System. *Trust Management 2003*. Lecture Notes Computer Science 2692. Springer-Verlag. pp.1-6.

[50] Li N., Grosof B.N. and Feigenbaum J. (2000). A practically implementable and Tractable Delegation Logic. *Proceedings of 2000 IEEE symposium on Security and Privacy*. IEEE.

[51] Li N. and Mitchell J.C. (2003). RT: A Role-based Trust-Management Framework. *Proceedings of the 3rd DARPA Information survivability Conference and Exposition (DISCEX III)*, IEEE.

[52] Li N., Mitchell J.C. and Winsborough W.H. (2002). Design of a role-based Trust Management Framework. *Proceedings of 2002 IEEE symposium on Security and Privacy*. IEEE.

[53] Lehti I. and Nikander P. (1998). Certifying Trust. *Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*. Lecture Notes Computer Science 1431. Springer- Verlag. pp. 83-98.

[54] Manchala D. W. (1998). Trust Metrics, Models and Protocols for Electronic Commerce Transactions. *Proceedings of the18th International Conference on Distributed Computing Systems*. IEEE.

[55] Mes A. D. and Rongen E. (2003). Web service Credentials. *IBM Systems Journal*. Volume 42(3).

[56] McGregor C. and Kumaran S. (2002). Business Process Monitoring using Web Services in B2B e-Commerce. *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02).*IEEE.

[57] Mihic M. (2004). Building Dynamic Application Networks with Web Services. *Proceedings of the 2004 ACM SIGMOD International conference on Management of data*. ACM. pp. 878-878.

[58] MSDN. Web Services Specifications Index Page. Online. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/wsspecsover.asp. Accessed on 19th September 2005.

[59] Mui L., Mohtashemi M. & Halberstadt A. (2002). A computational Model of Trust and Reputation. *Proceedings of the 35th Hawaii International Conference on System Sciences*. IEEE.

[60] Nadalin A., et al. (2004). Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). Online. 56 pages. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf. Accessed on 6th May 2005.

[61] NewComer E. (2002). Understanding Web Services XML, WSDL, SOAP and UDDI. Addison Wesley.

[62] Papazoglou P.M. and Yang L. (2002). Design Methodology for Web Services and Business Processes. *Proceedings of the Third International Workshop on Technologies for E-Services*. Lecture Notes in Computer Science 2444. Springer-Verlag. pp. 54-64.

[63] Patrick A.S. (2002). Building Trustworthy Software Agents. *IEEE Internet Computing*. Volume 6(6). pp. 46-53.

[64] Peltz C. (2003). Web Services Orchestration and Choreography. *Computer*. IEEE. Volume 36(10). pp. 46-52.

[65] Pirzada A.A. and McDonald C. (2004). Establishing Trust In Pure Ad-hoc Networks. *Proceedings of the 27th conference on Australasian computer science*. ACM. pp. 47-54.

[66] Roy J. and Ramanujan A.(2001). Understanding Web services. *IT Professional.* Volume 3(6). IEEE. pp.69-73.

[67] Seamons K.E., et al. (2003). Trust Negotiation in Dynamic coalitions. *Proceedings of the DARPA Information survivability Conference and Exposition (DISCEX '03)*. IEEE.

[68] Seamons K.E. et al. (2002). Requirements for Policy Languages for Trust Negotiation. *Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*. IEEE.

[69] Schimkat R.D., Nusser G. and Buhler D. (2000). Scalability and Interoperability in Service- Centric Architectures for the Web. *Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA'00)*. IEEE.

[70] Shirky C. (2002). Web Services and Context Horizons. *Computer.* Volume 35(9). IEEE. pp. 98-100.

[71] Skogsurd H., Bentallah B and Casati F. (2004) Trust-Serv: Model-Driven Lifecycle Management of Trust Negotiation Policies for Web Services. *Proceedings of the 13th International conference on World Wide Web*. ACM. pp. 53- 62.

[72] Skogsurd H., Bentallah B and Casati F. (2003) Model-Driven Trust Negotiation for Web Services. *IEEE Internet Computing.* Volume 7(6). pp. 45- 52.

[73] Stal M. (2002). Web Services: BEYOND COMPONENT-BASED COMPUTING. *Communications of the ACM*. Volume 45(10). pp. 71-76.

[74] Smith B., Seamons K.E. & Jones M. D. (2004). Responding to policies at Runtime in TrustBuilder. *Proceedings of the fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*. IEEE.

[75] Swarup V. & Fabrega J.T. (1999). Trust: Benefits, Models, and Mechanisms. *Secure Internet Programming, Security Issues for Mobile and Distributed Objects*. Lecture Notes in Computer Science 1603, Springer-Verlag. pp. 3-18.

[76] Tai S., Khalaf R. and Mikalsen T. (2004). Composition of Coordinated Web Services. *Middleware 2004*. Lecture Notes in Computer Science 3231. Springer-Verlag. pp. 294-310.

[77] Tan H.K. and Moreau L. (2001). Trust Relationships in a Mobile Agent System. *Proceedings of the 5th International Conference on Mobile Agents*. Lecture Notes in Computer Science 2240. Springer-Verlag. pp.15-30.

[78] Tan Y. and Thoen W. (2000). Formal Aspects of a Generic Model of Trust for Electronic Commerce. *Proceedings of the 33rd Hawaii International Conference on System Sciences*. Volume 6. IEEE.

[79] Varadharajan V. and Lin C. (2003). Modelling and Evaluating Trust Relationships in Mobile Agents Based Systems. *Proceedings of the International Conference on Applied Cryptography and Network Security ACNS (2003)*. Lecture Notes in Computer Science 2846. Springer-Verlag. pp. 176-190.

[80] Wang Y and Vassileva J. (2003). Trust and reputation Model in Peer-to-Peer Networks. *Proceedings of the Third International conference on Peer-to-peer Computing* . IEEE.

[81] Winsborough W.H., Seamons K.E. and Jones V.E. Automated Trust Negotiation. (2000). *Technical Report: TR-2000-05*. North Carolina State University at Raleigh.

[82] Winslett M., et al. (2002). Negotiating Trust on the Web. *IEEE Internet Computing*. Volume 6(6). pp. 30-37.

[83] Witkowski M. and Pitt J. (2000). Objective Trust-based Agents: Trust and Trustworthiness in a Multiagent Trading society. *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS)*. IEEE.

[84] W3C. Web Services Architecture Requirements. (2004). Online. http://www.w3.org/TR/wsa-reqs/. Accessed on 19th September 2005.

[85] Yu T., Ma X. and Winslett M. (2000). PRUNES: An efficient and complete Strategy for Automated Trust Negotiation over the Internet. *Proceedings of the 7th ACM conference on Computer and communications security.* ACM. pp.210-219.