Trust in Distributed Information Systems

Weiliang Zhao

A thesis submitted for the degree of Doctor of Philosophy

School of Computing and Mathematics

University of Western Sydney

February 2008
© Weiliang Zhao

Typeset in Computer Modern by \TeX{} and \LaTeX{} 2ɛ.
This is to certify that the dissertation is my own original work except where otherwise indicated. No part of this dissertation has been submitted as part of any other degree.

Weiliang Zhao

13 February 2008
To my family.
Acknowledgements

Many people have provided assistance throughout the completion of this dissertation.

I would like to acknowledge Professor Vijay Varadharajan for his support in every aspect of my doctorate, particularly his guidance to produce this research work and detailed suggestions to make this dissertation become a reality.

I would like to acknowledge Dr. Chun Ruan and Associate Professor George Bryan for their supervision, support, and encouragement in my PhD study.

I would like to thank Ms. Barbara Pinning, Ms Tracy Mills, Ms. Claudia Steinhoff, Dr. Carolyn McGregor, Dr. Leanne Rylands, and Professor Yan Zhang for their help in multiple administrative issues.

I would like to thank Luke Domanski, Abra Santilli, Angelos Molfetas, and other fellow students at the University of Western Sydney for their friendships and help.

I would like to thank other members of Information and Networked System Security Research under Professor Vijay Varadharajan for their help, discussions, and valuable comments on my works.

I would like to thank all my family for their support to complete the PhD study through all the good and bad times.
Finally, I would like to generally thank all the people who have provided assistance and help in the last several years.

Weiliang Zhao

February, 2008
Abstract

Trust management is an important issue in the analysis and design of secure information systems. This is especially the case where centrally managed security is not possible. Trust issues arise not only in business functions, but also in technologies used to support these functions. There are a vast number of services and applications that must accommodate appropriate notions of trust. Trust and trust management have become a hot research area. The motivation of this dissertation is to build up a comprehensive trust management approach that covers the analysis/modelling of trust relationships and the development of trust management systems in a consistent manner.

A formal model of trust relationship is proposed with a strict mathematical structure that can not only reflect many of the commonly used notions of trust, but also provide a solid basis for a unified taxonomy framework of trust where a range of useful properties of trust relationships can be expressed and compared. A classification of trust relationships is presented. A set of definitions, propositions, and operations are proposed for the properties about scope and diversity of trust relationships, direction and symmetry of trust relationships, and relations of trust relationships.

A general methodology for analysis and modelling of trust relationships in
distributed information system is presented. The general methodology includes a range of major concerns in the whole lifecycle of trust relationships, and provides practical guidelines for analysis and modelling of trust relationships in the real world.

A unified framework for trust management is proposed. Trust request, trust evaluation, and trust consuming are handled in a comprehensive and consistent manner. A variety of trust mechanisms including reputation, credentials, local data, and environment parameters are covered under the same framework. A trust management architecture is devised for facilitating the development of trust management systems.

A trust management system for federated medical services is developed as an implementation example of the proposed trust management architecture. An online booking system is developed to show how a trust management system is employed by applications.

A trust management architecture for web services is devised. It can be viewed as an extension of WS-Trust with the ability to integrate the message building blocks supported by web services protocol stack and other trust mechanisms. It provides high level architecture and guidelines for the development and deployment of a trust management layer in web services.

Trust management extension of CardSpace identity system is introduced. Major concerns are listed for the analysis and modelling of trust relationships, and development of trust management systems for digital identities.
# Contents

Acknowledgements

Abstract

1 Introduction

1.1 Distributed Information Systems

1.2 Trust and Trust Management

1.3 Research Motivations

1.4 Summary of Contributions

1.5 Outline of Dissertation

2 Background

2.1 Conceptions of Trust

2.1.1 Trust in Social Sciences

2.1.2 Trusted Computing

2.1.3 Marsh’s Formalism of Trust

2.1.4 Trust vs Reputation

2.1.5 WS-Trust

2.1.6 Trust Management

2.1.7 Domain Trust
### Contents

2.2 Trust Management Systems .................................. 37
   2.2.1 Pretty Good Privacy .................................. 39
   2.2.2 Public Key Infrastructure ............................... 41
   2.2.3 PolicyMaker and KeyNote ................................. 46
   2.2.4 REFEREE .................................................. 50
   2.2.5 IBM Trust Establishment ................................. 52
   2.2.6 Trust Negotiation Approach .............................. 54
2.3 Summary ....................................................... 60

3 Unified Taxonomy Framework of Trust .......................... 61
   3.1 Formal Model ............................................... 62
   3.2 Trust Classification ........................................ 71
   3.3 Scope and Diversity ........................................ 77
   3.4 Direction and Symmetry .................................... 80
   3.5 Relations and Operations .................................. 86
   3.6 Summary ....................................................... 97

4 Methodology for Analysis & Modelling of Trust Relationships 99
   4.1 Analysis and Modelling Guidelines .......................... 100
   4.2 Instances of Trust Relationships ............................ 103
   4.3 Individual Trust Relationships ............................... 104
   4.4 Whole Set of Trust Relationships ............................ 107
   4.5 Trust Relationships in View of Trust Management ............. 110
# Contents

4.6 Summary ........................................... 111

5 Unified Framework for Trust Management ........................................ 113

5.1 Introduction ......................................... 114

5.2 Trust Management Architecture ........................................... 117

5.2.1 TrustDatabase ...................................... 119

5.2.2 TrustControl ........................................ 120

5.2.3 LocatingTrust ....................................... 120

5.2.4 EvaluatingTrust ..................................... 121

5.2.5 ConsumingTrust ..................................... 123

5.3 System Components of TrustEngine ....................................... 126

5.4 System Setting Up and Operations ....................................... 133

5.5 Trust Management Algorithm ......................................... 137

5.6 Discussion ............................................. 145

5.7 Summary ............................................. 148

6 Trust Management System for Federated Medical Services ............ 151

6.1 Trust Management in Federated Medical Services ................... 152

6.2 Trust Management System as Implementation of TrustEngine ........ 155

6.3 Web Services for Trust Management .................................... 159

6.4 Online Booking System for Federated Medical Services ............. 169

6.5 Summary ............................................. 180
7 Trust Management and WS-Trust/Role Models/CardSpace 183

7.1 Trust Management in Web Services as Extension of WS-Trust 184

7.1.1 Introduction 184

7.1.2 Trust Relationships 187

7.1.3 Trust Management Architecture 190

7.1.4 Web Services Architecture Layers 193

7.1.5 Discussion 195

7.2 Trust Management as Extension of Role Models 197

7.2.1 Role Based Access Control 197

7.2.2 Extensions of Role Based Access Control 199

7.3 Trust Management Extension of CardSpace 202

7.3.1 Identity Metasystem and CardSpace 202

7.3.2 Trust Management for Digital Identities 204

7.4 Summary 208

8 Conclusions 209

8.1 Contributions 210

8.2 Future Work 214

A Publication List from This Dissertation 217

B Source Codes 219

Bibliography 221
Chapter 1

Introduction

Information processing, using computer systems on a global scale, has become the foundation of twenty-first-century life. The Internet and the eXtensible Markup Language (XML) based Web Services technologies have promoted and speeded the growth of distributed information systems that allow many users to interact and connect over open networks. There are a vast amount of distributed information systems that run organizations and enterprises such as online shops, libraries, transportation systems, hospitals, online casinos, and travel agents. These distributed information systems have become the foundation for global business and economics. Distributed information systems provide the foundation for trading partnerships and the automation of business collaborations within organizations, across enterprises, and across the Internet. Information could be shared and exchanged between different enterprises and processed cooperatively across enterprise boundaries.

Trustworthy computing and trust management are becoming increasingly significant. Trust management is an important issue in the analysis and design of secure information system, especially when centrally managed security is not
possible. A clear and comprehensive understanding of trust, from both sides of business and technology, plays an important role in the design of secure distributed information systems.

1.1 Distributed Information Systems

In this dissertation, distributed information systems refer to information systems where the resources are spatially distributed and connected by networks. This is in contrast to a centralized system where the resources are gathered in a single location. The definition includes both hardware and software resources. Information technologies have been going through a revolution in the past several decades. More and more complex applications are being developed to support business functions in an increasingly competitive business environment, and more flexible IT infrastructure is needed to provide greater freedom of design to build these information systems. The integration of a variety of discrete applications, and potentially diverse data sources, is the main concern in the development of these distributed information systems. Enhanced system interoperability protocols and networking capabilities are making the creation of wide-area-based, multi-platform systems possible. Major advances in communication technology and in World Wide Web have promoted and speeded the growth of distributed information systems.

A distributed information system consists of a dispersed set of application
programs communicating with one another by means of messages transmitted over various networks. An important purpose of distributed information systems is to automate the business operations of commercial enterprises or organizations. In a distributed information system, multiple autonomous nodes are interconnected by communication networks. Information is shared among multiple entities and information processing is beyond a single enterprise or organization. Distributed information systems provide the foundation for operational partnerships and business collaborations. These systems can be distributed over various networks and across various business boundaries. Both Intranet and Internet could be employed in these systems. The business operations are supported by distributed, message-based computer systems.

The Web Services approach is an important step in the evolution of distributed information systems. The eXtensible Markup Language (XML) based Web Services technologies have been rapidly evolving since 1999. Web Services provide functions for information processing among trading partners over networks, often the Internet, to support related business operations. Web Services technologies provide facilities for the Business-to-Business integration. There are a vast number of service-oriented applications on the Internet and they are often loosely coupled. Web Services technologies target loosely-coupled, language-neutral, and platform-independent, ways of linking applications for business process automation within organizations, across enterprises, and over the Internet. Web Services are based on open industry standards to integrate with partners and clients. The
important standards include eXtensible Markup Language, Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description Discovery and Integration (UDDI).

Security is one of the biggest challenges in computing technology. Today’s distributed information systems are more vulnerable to security threats. The number and different types of threats and security attacks are growing every day. There are a broad variety of security risks and information could be stolen, lost, or modified. The confidentiality, authentication, integrity, and non-reputation, are security requirements that focus on how to protect data and programs in open environments. These requirements are normally achieved through cryptographic methods. Authorization is another security requirement which guarantees the protection of resources such as data and applications by only allowing appropriate entities to access these resources. Authorization plays a crucial role in information processing, particularly when multiple applications and services are involved and they belong to different security domains. Authorization solutions can employ some cryptographic techniques but not necessarily.

A broad range of mechanisms to address the above security requirements have been studied over many years and there are many mature security solutions. However, there are still many challenges when these mature security solutions come to deployment in distributed information systems and emerging technologies, such as web services. Actually, information security has been a hot research field for several decades and there is no sign for this trend to stop with the emergence of
pervasive distributed technology, services, and applications.

1.2 Trust and Trust Management

The migration from a centralized information system to a distributed information system means that some operations and transactions will span a range of domains, and multiple entities may be involved in these operations and transactions. The involved entities may have different levels of familiarity and information access. The entities may not be trusted to the same extent. The notion of trust must be introduced and it is beyond the traditional security requirements mentioned in last section. The trust between customers and the providers of the services is crucial in electronic commerce transactions on the Internet. There are multiple trust requirements when a service needs to be trusted as it claims, the privacy of customers needs to be protected, and the providers of the services need to be paid as expected. The trust decision must be made before a business transaction can be achieved, and it forms the basis for the customer’s decision to choose the provided services and process the business transaction. Trust has become an intrinsic part of e-Business.

The issue of trust is one of the major concerns in distributed information systems in a range of research areas such as web services, grid computing, cooperative computing, and forensic computing. Trust issues arise not only in business functions, but also in technologies used in the implementation of these
Introduction

functions. The business requirements and the technologies employed in target information systems are normally mingled with each other. The target distributed information systems must address all these trust issues.

Here we list some of the major characteristics of distributed information systems related with trust issues:

- **Multiple Trust Mechanisms**

  Closed information systems have centralized control for security and trust. Trust is normally predefined and the related data is stored in the information system. Some distributed information systems and technologies only accept credentials to establish and broker trust relationships. At the same time, there is also an alternative trend of using reputation based trust for collaborations to satisfy security requirements in distributed information systems. Instead of centrally managed data and/or credentials, involved entities may use specific knowledge (both local and acquired from remote nodes or resources) to make trust decisions. In more complex cases, multiple trust mechanisms, such as credentials and reputation, can be required to work together for a single trust decision.

- **Open Nature**

  Business functions are normally open in modern distributed information systems. For example, everyone has the access to an online hotel booking service. The system is open to everyone and it can cover both known fre-
quent customers and some previously unknown customers. Different trust relationships must be figured out in the system for various business operations and transactions. These distributed information systems have intrinsic requirements for appropriate trust management. Modern distributed information systems are normally running over open networks, particularly over the Internet. The open nature of a distributed information system makes trust management a crucial part of the whole information system.

- **Multiple Domains**

Modern distributed information systems often span several networks, and there are multiple administrative or organizational boundaries. A typical distributed information system is composed of many interconnected heterogeneous resources that belong to multiple domains, and the relationship between these domains can be peer to peer or hierarchical, or a combination thereof. For example, a multi-enterprise financial trading system [76] is distributed over various networks worldwide including the Internet. Various enterprises and organizations are simply the components of the system, and each of them has its own internal information processing system. There are multiple sub systems such as the stock market information system, brokerage houses, and online customers (or their workstations), the Federal Reserve, investment banks, and the networks through which all these components communicate with each other. There are multiple boundaries and
domains in such a complicated information system. The trust relationships can be quite complex in such a system. There are many challenging trust issues in cross-boundary operations, management, and administration.

• **Real Time Trust**

In many distributed information systems, trust relationships must be evaluated and established in real time. Trust relationships are not static and they are continuously changing. The dynamic properties of trust must be included in many distributed information systems. Multiple evidences must be collected in real time for trust evaluation. The valid period of the result of trust evaluation is also time relevant (for example, it can only be used in a fixed time period). The concept of time is an important concern in most trust issues in distributed information systems.

• **Scalability**

Every distributed information system has its specific scale. A distributed information system may have a large number of resources and a large number of users, or potential users. Some of these distributed information systems are required to scale up to the scope of the Internet. The scale of a distributed information system is crucial in trust management.

• **Complexity**

Distributed information systems can be very complicated. Modern distributed information systems can have complicated business functions and
employ multiple advanced technologies. The trust management tasks can be very complex and challenging.

The above items describe the important challenges for trust management in modern distributed information systems.

1.3 Research Motivations

This section describes the research motivations. They arise from the major characteristics of distributed information systems and the challenges they give rise to in terms of trust described in last section. Research motivations include:

- The notion of trust has been around for many decades in different disciplines, in different disguises, and yet there is no consensus for the meaning of trust. A clear and comprehensive definition that can be used to capture a range of commonly understood notions of trust is still lacking. It is necessary to build up a formal model of trust relationship as the starting point of the research about trust in distributed information systems.

- There can be very complicated trust relationships in some distributed information systems, and there is very limited research about trust relationships and their properties. It is highly desirable to have a standard, unified taxonomy framework of trust that can cover important properties of trust relationships, and provide appropriate terms and enable tools for the analysis and modelling of trust relationships.
• In order to deal with trust issues in analysis, design, implementation, and maintenance of distributed information systems, it is desirable to have a general methodology that can provide general principles and guidelines for the analysis and modelling of trust relationships.

• Identity based trust and credential based trust are the two approaches for establishing trust between involved entities. In modern distributed information systems often there is no centralized control; trust is becoming more dynamic; and the evidences for trust are prone to change over time. Hence, a pure identity based, or credential based approach, cannot satisfy complex trust requirements in modern distributed information systems. In many cases, different kinds of evidences are required for the establishment of a trust relationship. The context based trust mechanisms must be introduced for trust management, as multiple aspects need to be considered in making the trust decisions such as past experience, recommendation from authorities, reputation in the community, and the real time environment. It is necessary to have a unified trust management framework to cover different trust mechanisms.

• Developing a trust management system can be a complex and costly task. It is desirable that the development of a trust management system can be automated to a substantially high level by simply implementing some business logic. It will be very helpful in the development of trust management
systems if there is a well-defined trust management architecture that can provide high-level designs of computing components for trust management tasks and their inter-relationships.

- The trust management architecture must be implementable. It will be highly appreciated to have a real world example with the details about the implementation of the trust management architecture. The example should show the translation between the high level architecture and programming level implementation.

- A set of specifications of web services based on SOAP have been published and adopted to provide a rich messaging environment. WS-Trust is the specification used to establish the presence of, and broker trust relationships based on security tokens. However, many trust management tasks are beyond the message level of web services. Therefore, it is desirable to have a trust management architecture for web services as an extension of WS-Trust. The architecture should have the ability to cover a broad range of trust mechanisms other than just the security tokens in SOAP messages.

The motivations discussed above effectively set out the goals of the research in this dissertation. The objectives of the dissertation are summarized as:

This dissertation is intended to develop a trust management approach in distributed information systems. The approach tackles the analysis/modelling trust relationships and the development of trust man-
management systems in a consistent manner. A formal model of trust relationship and a unified taxonomy framework of trust have been developed. The methodology for analysis/modelling of trust relationships and a unified framework for trust management are presented.

1.4 Summary of Contributions

This dissertation presents the research on the topic of trust in distributed information systems. The contributions of this dissertation include a formal model of trust relationship, a unified taxonomy framework of trust, a general methodology for analysis and modelling of trust relationships, a novel unified framework of trust management with TrustEngine as the proposed trust management architecture, an implementation example of federated medical services using the trust management architecture, trust management layer for web services, and trust management for digital identities as an extension of CardSpace [83, 26, 82, 6]. These contributions address the research motivations described in the previous section.

Firstly, this dissertation proposes a formal model of trust relationship with a clear and comprehensive definition that can be used to capture a range of commonly understood notions of trust in distributed information systems. The proposed model has a strict mathematical structure and expressive power for different computing purposes. The formal model of trust relationship is the
cornerstone of the approach for trust issues in distributed information systems.

Secondly, this dissertation proposes a new unified taxonomy framework of trust based on the formal model of trust relationship. The taxonomy framework is composed of a set of definitions about the properties of trust relationships in distributed information systems. The taxonomy framework can provide accurate terms and useful tools for enabling the analysis, design, and implementation of trust. The framework includes the classification of trust, a new hierarchy of trust relationships, a definition of trust scope label to describe the scope and diversity of trust relationship, a set of properties of trust direction and trust symmetry, and a set of definitions, propositions, and operations, based on the relations of trust relationships.

Thirdly, this dissertation proposes a general methodology for analysis and modelling of trust relationships. The general methodology is based on the formal model of trust relationship and the unified taxonomy framework of trust. The formal model of trust relationship and the taxonomy framework of trust provide basic terms, scenario examples, and enabling tools, for the incremental and iterative life cycle of the analysis and modelling of trust relationships. The general methodology provides practical guidelines from multiple aspects, and includes a range of major concerns for the analysis and modelling of the whole set of trust relationships in distributed information systems.

Fourthly, this dissertation proposes a unified framework for trust management that can cover a variety of trust mechanisms including reputation, credentials,
local data, and environment parameters. In the proposed unified framework for trust management, trust request, trust evaluation, and trust consuming are handled in a comprehensive and consistent manner. To express the unified framework for trust management, TrustEngine is devised as the target trust management system. It contains the trust related computing components that could be separated from applications in distributed information systems. The components of TrustEngine can be viewed as enabling tools for trust management. If necessary, multiple trust mechanisms can be easily assembled together. The unified framework provides high level details and guidelines for trust management systems, and it can employed for more efficient development and maintenance of trust management systems that can support a wide range of different context-based trust policies.

Fifthly, this dissertation provides details of a trust management system for federated medical services as an implementation example of proposed trust management architecture. The implementation example shows how the trust management approach facilitates the development of trust management systems in the real world.

Sixthly, this dissertation introduces a web service trust management layer in web services architecture layers. The trust management layer can facilitate the integration of the building blocks, defined by web services protocol stack, and other computing components, for trust management tasks. The trust management layer for web services covers the specific additional trust requirements of
web services. A range of trust mechanisms in distributed information systems are embraced in the web services paradigm.

Seventhly, this dissertation provides discussion about trust management for digital identities. The taxonomy framework of trust and trust management architecture are employed as tools for the analysis and modelling of trust relationships and the development of trust management extension of CardSpace.

The above contributions can be categorized into two groups. The first group focuses on trust relationships. The formal model of trust relationship and unified taxonomy framework of trust provide a solid theoretical basis to understand trust issues in distributed information systems. They also provide terminologies, scenario examples, and tools, for the analysis and modelling of trust relationships. Based on the formal model of trust relationship and unified taxonomy framework, the general methodology provides practical guidelines and detailed concerns for the analysis and modelling of trust relationships in distributed information systems. The second group focuses on the unified trust management framework. It addresses the efficient development and maintenance of trust management systems. With the help of the trust management framework, it is possible for developers to create a trust management system by simply implementing some business logic. The development of trust management systems can be automated to a substantially high level.

Beyond the existing research about trust, it is the belief that this dissertation is the first one to provide a comprehensive investigation of trust in distributed
information systems that covers both the analysis/modelling of trust relationships, as well as the development of trust management systems in a consistent manner. The dissertation provides a comprehensive solution that, when implemented, will allow developers to build trust solutions that leverage and expand upon existing trust standards and computing components, while allowing them to be integrated in the unified trust framework and take the advantage of the overall trust solution.

1.5 Outline of Dissertation

This dissertation has eight chapters as follows:

Chapter 1 provides an introduction to trust and trust management in distributed information systems. Research issues and the motivations are described. The contributions of this dissertation are summarized and the outline of dissertation is listed.

Chapter 2 reviews major research works in the area of trust and trust management in distributed information systems. Several major conceptions of trust in the computing world, and several popular trust management systems, are discussed. It provides background and foundation for the research on trust in distributed information systems.

Chapter 3 describes a unified taxonomy framework of trust. A formal model
of trust relationship is defined and the classification of trust is provided. A range of properties of trust relationships related to scope and diversity, direction, and symmetry of trust, are defined. A set of operations on trust relationships and relative types of trust relationships are also defined. Some scenario examples are provided. Parts of this work have been published in [1], [2], [3], and [6] of Appendix A.

**Chapter 4** describes the methodology of analysis and modelling of trust relationships that is based on the formal model of trust relationship and taxonomy framework of trust. Some general guidelines are provided. The instance of a trust relationship is defined. The analysis and modelling of individual trust relationships and the whole set of trust relationships are discussed. Parts of this work have been published in [2] and [4] of Appendix A.

**Chapter 5** provides the unified framework for trust management. Trust management architecture *TrustEngine* is proposed. The design details of generic system components are provided. The setting up of a generic system and operations of the target trust management system are discussed. Parts of this work have been published in [4] and [5] of Appendix A.

**Chapter 6** provides details of a trust management system for federated medical services, which serves as an example of the implementation of *TrustEngine* architecture. The details of the web services for a trust management system
Introduction

for federated medical services are provided. An online booking system for federated medical services is developed which consumes the web services for trust management tasks.

Chapter 7 discusses trust management in web services, the relation of the trust management approach to role based access control models, and trust management extension of CardSpace. Trust relationships in web services paradigm are discussed. A trust management architecture for web services is proposed and it is embraced in the web services architecture layers. Some research results about trust management for web services have been published in [7] of Appendix A. The role based access control is reviewed and the extensions of role based access control by our trust management approach are discussed. The CardSpace is reviewed and the possible extension of CardSpace for trust management of digital identities is discussed.

Chapter 8 concludes this dissertation. It summarizes the main contributions of the research and provides a brief discussion of potential future research and further work.
Chapter 2

Background

There has been a lot of research that studied and modelled trust across different areas. The notion of trust has been around for many decades in different disciplines, in different disguises. Trust management and trustworthy computing are becoming increasingly significant at present. In this chapter, we will discuss some popular conceptions of trust in the computing world, and review several well-known trust management systems. This chapter serves as background information for the research.

In Section 2.1, the conception of trust in social sciences, and several conceptions of trust in the computing world, are discussed. In Section 2.2, several trust management systems are reviewed. In Section 2.3, a short summary for this chapter is provided.

2.1 Conceptions of Trust

The concept of trust has been used in a broad variety of contexts. There are different conceptions of trust. Trust is a general term broadly used in day to
day life and its original concept is rooted in social sciences such as sociology, social psychology, law, and economics. In the computing world, trust has been initially used in trusted systems [121] and trusted computing [70]. S. Marsh gave a formalization of trust [80]. The term trust has been used in reputation systems and some researchers view trust as reputation. Trust has also been a key concept in Microsoft’s domain trust, web service trust language (WS-Trust), and trust management systems.

2.1.1 Trust in Social Sciences

Trust is a common phenomenon in everyday life relationships and in laws and regulations in society. Trust is the foundation for social order spanning many intellectual disciplines. The concept of trust has multiple origins and it may come from the perspectives of personality theorists, sociologists, economists, and social psychologists. In the Webster dictionary, trust is defined as:

- An assumed reliance on some person or thing. A confident dependence on the character, ability, strength, or truth of someone or something.

- A charge or duty imposed in faith or confidence or as a condition of a relationship.

- To place confidence (in an entity).

The fundamental meaning of trust is normally related to the existence of some kind of relationship between two entities, and confident positive expectations re-
garding the other’s conduct or behavior. J. D. Lewis and A. Weigert [72] point out that trust is indispensable in social relationships. J. K. Rempel and R. Souster [102] claim that trust is one of the most important and necessary aspects of any close relationship and trust has three fundamental elements, namely predictability, dependability, and faith. Many researchers have given different definitions of trust. M. Deutsch [35] provided one of the earliest definitions of trust as follows: “an individual may be said to have trust in the occurrence of an event if he expects its occurrence and his expectations lead to behavior which he perceives to have greater negative consequences if the expectation is not confirmed than positive motivational consequences if it is confirmed”. In M. Deutsch’s definition, trust involves the notion of motivational relevance as well as the notion of predictability. There have been several different research streams on trust between humans. P. Worchel [129] classifies them into three main categories, namely individual trust, societal trust, and relationship trust. The individual trust [101, 104] is the approach of personality theorists that focuses on the characteristics of individual personality. Trust as a belief, expectancy, or feeling, is rooted in the personality based on early psychological development and past experiences. Societal trust [72, 37] is the approach of sociologists and economists that focuses on the development of trust between individuals and institutions. As a general societal view of trust, an individual has to trust an institution (such as an organization) or a societal structure (such as a judicial system). Based on T. C. Earle and G. T. Cvetkovich [37], social trust is the process by which individuals assign to other
Background

persons, groups, agencies, or institutions, the responsibility to work on certain
tasks. Relationship trust [22, 112] is the approach of social psychologists that
focuses on the factors that create or destroy trust in individuals involved in a
personal or work relationship. Relationship trust is viewed as an expectation of
the other party in a relationship. Butler [22] states “trust in a specific person is
more relevant in terms of predicting outcomes than is the global attitude of trust
in generalized others”. B. R. Schlenker et al [112] define trust as “the reliance
upon information received from another person about uncertain environmental
states and their accompanying outcomes in a risky situation”.

Some researchers have tried to provide a general definition of trust that can
cover all of the aspects of individual trust, societal trust, and relationship trust.
D. Gambetta [44] defines trust as: “trust (or, symmetrically, distrust) is a partic-
ular level of the subjective probability with which an agent assesses that another
agent or group of agents will perform a particular action, before he can monitor
such action (or independently of his capacity ever to be able to monitor it) and
in a context in which it affects his own action”. D. Gambetta’s definition gathers
together the thoughts from a broad variety of research areas. The subjective
nature of the probability means that the individual’s personality characteristics
hold an important role in trust. Both individuals and institutions can be viewed
as agents, and a relationship between agents is implied in the definition.

Trust is closely related with risks. N. Luhmann sees trust as an attitude which
allows for risk-taking decisions. N. Luhmann [78] states that “trust is a solution
for specific problems of risk”. N. Luhmann [77] argues that the acceptance of risk and the means, through trust, to cope with and assimilate risk into the decisions, enables us to exist in the complex society which is around us. The knowledge of risk, and its implications, allows people to make plans for the future which take the risks into account.

### 2.1.2 Trusted Computing

In the computing world, the trust is initially used in the context of Trusted Computing Base (TCB) that is the totality of protection mechanisms within a computer system, including both the trusted hardware and trusted software [121, 70]. The “trusted” refers to the status that the system, hardware, or software will behave in specific ways. In this context, security and consistency are the attributes of trust. Trusted computing (a terminology from 1999) is related with the original concept of TCB but it is more specific. S. Pearson [97] defines a key notion in trusted computing to be a trusted platform, as “a Trusted Platform is a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes”. Trusted computing systems integrate security into their core operations, rather than implementing it via add-on applications [122]. Trusted computing is achieved through a combination of hardware and software. In trusted computing, hardware is added to a computer for supporting trusted computing functions, and there are corresponding software changes to adapt the new hardware with
specific security purposes. Trusted Computing Group defines “trust is the expectation that a device will behave in a particular manner for a specific purpose” and “Trusted Computing Platform is a computing platform that can be trusted to report its properties”.

In the history of trusted computing, there are three major trusted computing initiatives: Trusted Computing Group (TCG)/Trusted Computing Platform Alliance (TCPA) [50], Microsoft’s Palladium/Next-Generation Secure Computing Base (NGSCB) [30], and Intel’s LaGrande/Trusted Execution Technology [29].

TCPA was formed in 1999 by Compaq Computer (now part of Hewlett-Packard), HP, IBM, Intel, and Microsoft for creation of a trusted platform module, a motherboard mounted cryptographic processor with a unique digital signature. TCG is the successor of TCPA. The purpose of TCG is to develop, define, and promote open specifications for trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices [50]. The most important device in trusted computing is the Trusted Platform Module (TPM) that is basically a secure micro-controller with added cryptographic functionalities. The most important software specification of TCG is the Trusted Computing Module Software Stack (TSS) that includes software services to facilitate the use of the TPM.

Microsoft’s NGSCB [30], originally called Palladium, is a security technology for the Microsoft Windows platform. NGSCB is part of Microsoft’s Trustworthy Computing initiative. Microsoft’s vision of NGSCB is to create new security tech-
nology for the Microsoft Windows platform with a unique hardware and software
design to improve security. NGSCB can provide unprecedented capabilities for
enabling secure processing on the Microsoft Windows PC platform, with a new
secure computing base designed to be running parallel to the regular Windows
environment.

Intel’s Trusted Execution Technology [29], formerly called LaGrande Tech-
nology, uses hardware extensions to Intel’s processors and chipsets to enhance
security capabilities against software-based attacks. Intel’s Trusted Execution
Technology capabilities include: protected execution and memory spaces, sealed
storage for encryption keys and other sensitive data, attestation for the assurance
of Trusted Execution Technology environment, measured launch capability, and
memory protection. Trusted Execution Technology provides hardware-rooted
trust for virtual applications.

2.1.3 Marsh’s Formalism of Trust

S. Marsh [80] presents a formalism for trust as a computational concept. The
formalism targets many aspects of trust in sociology, social psychology, and dis-
tributed artificial intelligence. Marsh’s work provides a further step in the direc-
tion of a proper understanding and definition of human trust. Marsh’s formalism
provides the social sciences with a valuable tool for a precise discussion of trust.
It provides a basis for multi-agent systems to embed trust within agents. In
Marsh’s formalism, trust is a subjective measure that can be used as a reasoning
tool in embodied agents. The formalism allows a precise reasoning about trust while being relatively simple. The formalism provides agents the capability of using trust as a decision making tool for the evaluation of interactions. The formalism is extensible in its implementations.

Marsh separated trust into three aspects: basic, general, and situational trust. Particular agents are represented using the letters $a$ to $z$. An agent can be considered to be an independent entity which is extant with other similar entities. Situations are represented by Greek letters $\alpha$ to $\omega$. A situation is a specific point in time relative to a specific agent.

- **Basic Trust** is the trusting disposition of the agent that is derived from the past experience in all situations, through the agent’s entire life of experiences. It is represented by $T_x$ with value range $-1 \leq T_x < +1$.

- **General Trust** is the trust in two agents. The notation of general trust is $T_x(y)$ that stands for ‘$x$ trusts $y$ with the value’. The value range is $-1 \leq T_x(y) < +1$. The value represents the amount of trust that $x$ has in $y$. It is not relative to any specific situation.

- **Situational Trust** is a representation for the amount of trust that an agent has in another in a given situation. The notation of situation trust is $T_x(y, \alpha)$ that stands for ‘$x$ trusts $y$ in situation $\alpha$ with a value’. The value range is $-1 \leq T_x(y, \alpha) < +1$. A situation means something different to each agent experiencing it. Different situations require different considerations
Situational trust is the key concept in Marsh’s formalism. Actually, Marsh’s approach focuses on situational trust by using subjective variables to calculate a trust value. In Marsh’s formalism of estimating situation trust, utility and importance are independent factors. The notation for utility is similar to that for knowledge. $U_x(\alpha)$ stands for the amount of utility $x$ gains from situation $\alpha$ with a value range $-1 \leq U_x(\alpha) \leq +1$. Utility is taken to be based on the expected utility theory [134]. Utility is generally measurable, or relatively straightforward to find an estimate for. Utility is an accepted measure of the actual outcome of a situation. On the other hand, importance is a subjective judgment of a situation. Importance is a subjective measure of the expected benefits to be gained from a situation under consideration. $I_x(\alpha)$ stands for the importance of a situation $\alpha$ for agent $x$ with a value range $0 \leq I_x(\alpha) \leq +1$. The basic formula for estimating situational trust is as follows:

$$T_x(y, \alpha) = U_x(\alpha) \times I_x(\alpha) \times T_x(y)$$

In estimating situational trust in an agent, the knowledge of that agent in other settings, and in other situations, are embedded in the general trust values the trustor has of the trustee. The estimate of general trust is denoted by $T_x(y)$ for the amount $x$ trusts $y$. It is $x$’s estimate after taking into account all possible relevant data with respect to $T_x(y, \alpha)$ values in the past.

Actually, situational trust is defined to provide a means by which one agent
Background

judges whether or not to cooperate with another agent. Indeed, the basic assumption is that, if the situational trust is above a cooperation threshold, cooperation will occur; if not, cooperation will not occur.

\[ T_x(y, \alpha) > CooperationThreshold_x(\alpha) \Rightarrow Will_Cooperate(x, y, \alpha) \]

The cooperation threshold is a subjective measure, tempered by objective beliefs. There are different ways to calculate the cooperation threshold based on the states and assumptions of the multi-agent system. Marsh has proposed three versions of the formula for calculating the cooperation threshold with input variables as risk, competence, importance, and general trust. The determination of risk and competence involves differing methods for different situations. In S. Marsh’s approach, the calculation of cooperation threshold is as important as that of situational trust.

S. Marsh provides the modification of the formalism after considering reciprocation of agents and memory. The formalism provides the agents the capability of using trust as a decision making tool for the evaluation of interactions. S. Marsh’s PhD dissertation [80] provides the discussion of implementation and some experiments that have been performed on a testbed, populated with simple trusting agents, that substantiates the utility of the formalism. The formalism is extensible and it provides a starting point for work concerned with trust in multi-agent systems.
2.1.4 Trust vs Reputation

Trust and reputation are two related terms and both of them have strong foundations in social sciences. In several contexts, trust and reputation are even taken to be the same. However, we believe that trust and reputation are two different concepts. Trust talks about the relationship between two entities and it is a subjective expectation. Please see Section 2.1.1 for more details about trust as a concept in social sciences. In contrast, reputation talks about public opinion toward an entity (an entity can be a person, group of people, or an organization). Reputation is some socially transmitted belief about some properties of an entity.

In the computing world, L. Mui et al [85] have made explicit the difference between trust and reputation. L. Mui et al defined “reputation as a quantity relative to the particular embedded social network of the evaluating agent and encounter history” and “trust as a dyadic quantity between the trustor and the trustee which can be inferred from reputation data about the trustee”. In an agent system, the reputation of an agent is relative to the particular embedded social network, community, or society, in which the agent is being evaluated, and an agent tends to trust other agents with a good reputation for being trustworthy. Reputation is used as input information to evaluate trust.

Trust and reputation have become important topics of research in distributed computing. Multiple reputation systems have been proposed and implemented in e-commerce systems such as eBay and Amazon. In these systems, trust will
be inferred from the related reputation.

A reputation system collects, distributes, and aggregates, feedback about participants’ past behavior. In a reputation system, there are long-lived entities and they form a community. The feedback on the behavior of acting entities can be captured and distributed in the community. For example, eBay employs a simple reputation system to record a rating of the peer party (either positive, negative, or neutral) after a transaction; a party’s reputation is expressed by the count of positive and negative transactions in that party’s history; this reputation can be used by other parties to decide whether the reputation holder can be trusted.

Reputation-based systems such as XREP [33], NICE [71], P-Grid [2], provide a facility to compute the reputation of an involved entity by aggregating the perception of other entities in the system. There are many proposals with ad-hoc schemes to formulate reputation [46, 131, 38, 105, 106]. Some reputation systems like TrustNet [111] and NodeRanking [99] utilize existing social relationships to compute reputations based on various parameters.

2.1.5 WS-Trust

Several versions of WS-Trust have been published since 2002, the latest one is WS-Trust 1.3 [91]. WS-Trust targets at a framework for requesting and issuing security tokens, and to broker trust relationships. It defines extensions that build on Web Service Security (WS-Security) to enable security token interoperability [28].
WS-Trust enables applications to construct trusted SOAP message exchanges [124]. In WS-Trust, trust is represented through the exchange and brokering of security tokens. WS-Trust supports a wide variety of security models. WS-Trust has two key driving requirements as (i) requesting and obtaining security tokens; and (ii) establishing, managing, and assessing, trust relationships. WS-Trust includes web service trust model, security token service framework, extensions for security token exchange, and key and token parameter extensions. WS-Trust provides a protocol agnostic way to issue, renew, and validate, security tokens. The set of mechanisms and security protocols are flexible.

In WS-Trust, trust [91] is defined as follows: “Trust is the characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes”. WS-Trust defines three types of trust as direct trust, direct brokered trust, and indirect brokered trust. In WS-Trust 1.3 [91], they are defined as:

- **Direct Trust** is when a relying party accepts as true all (or some subset of) the claims in the token sent by the requestor.

- **Direct Brokered Trust** is when one party trusts a second party who, in turn, trusts or vouches for, a third party.

- **Indirect Brokered Trust** is a variation on direct brokered trust where the second party negotiates with the third party, or additional parties, to assess the trust of the third party.
In the web service security model defined in WS-Trust, a web service can require an incoming message to prove a set of claims. The required claims, and related information, are expressed in terms of policies. These policies follow the grammar for policies defined in WS-Policy. The mechanisms for associating such policies with the subjects to which they apply are defined in WS-PolicyAttachment. The service will ignore or reject a message if the message cannot provide the required proof of claims. A requester sends messages and security tokens for the proof of the required claims. The requester, or someone on its behalf, may not have the required claims on hand, and it is necessary to obtain the necessary claims by contacting other web services that are referred to as security token services. These security token services broker trust between different trust domains by issuing security tokens. A security token service may be a web service and the requester may be a service. The security token service model is illustrated in the Figure 2.1. A web service has its trust engine as a conceptual

![Figure 2.1: Security Token Service Model](image)
component that evaluates the security-related aspects of messages. A web service with a policy receives a message from a requester that may include security tokens. The message may have some protection using WS-Security mechanisms. The trust engine will perform the following key steps:

- Verify that the claims comply with the policy and the message conforms to the policy.

- Verify the signatures on the attributes of the claimant. In brokered trust models, the signature may verify the identity of the intermediary who asserts the identity of the claimant.

- Verify that the issuers of the security tokens can be trusted for the claims they have made. The trust engine may need to externally verify or broker tokens (trust engine sends security tokens to a security token service for obtaining other security tokens to be used in the evaluation).

When the above conditions are satisfied, the service will process the service request. WS-Trust defines how security tokens are requested and obtained from security token services, and how these services may broker trust and trust policies. There are different models for obtaining tokens and brokering trust. There are three situations for the security token issuance: (1) Token Acquisition: a security token is requested explicitly as part of a message flow; (2) Out-of-Band Token Acquisition: a security token is not received in response to a direct SOAP
request; and (3) Trust Bootstrap: an administrator or other trusted authority may designate that all tokens of a certain type are trusted.

WS-Trust provides a security token service framework for token issuance. The framework provides the basic structure of token request and response of XML elements identifying the general mechanisms and most common sub-element. WS-Trust provides details about a set of bindings using the model framework. For a simple request and response for security tokens, WS-trust defines issuance binding, renewal binding, cancel binding, and validation binding. In order to deal with complicated situations where a set of exchanges between the parties is required prior to returning (e.g., issuing) a security token, WS-Trust defines the extensions to the base WS-Trust mechanisms to enable exchanges for negotiation and challenges. WS-Trust also defines key and token parameter extensions and key exchange token binding. In WS-Trust, SOAP fault mechanism is employed to handle errors. A series of XML elements are introduced for various purposes in the security token service framework. The syntax is flexible and the framework is extensible. More details can be found in the original documentation of WS-Trust [91].

2.1.6 Trust Management

Trust has been broadly used in the context of trust management [65, 14]. Trust management has emerged as a new philosophy to provide standard, general-purpose mechanisms to address trust issues in distributed applications. Multiple
trust management solutions [56, 5, 7, 14, 20, 27] have been broadly used in distributed information systems. Trust management is an important issue in security analysis and design, particularly when centrally managed security is not possible. Trust management is becoming increasingly significant and it is one of key concerns of this dissertation. Trust management will be discussed in details in Section 2.2 where a review on several well-known trust management systems will be provided.

2.1.7 Domain Trust

Microsoft has defined domain trust as a server technology in its Windows NT Server, Windows 2000 Server, and Windows 2003 Server. A domain trust allows users from a trusted domain to access resources in a trusting domain. Users/resources are managed in different multiple domains. The trusting domain is willing to accept login information and authentications from the trusted domain. The domain administrator is responsible for the users and resources within the domain. Domains provide the basis to set the scope of security policies.

Microsoft defines trust relationship as “a logical relationship established between domains to allow pass-through authentication, in which a trusting domain honors the logon authentications of a trusted domain”. The trust relationships between two domains can be one-way trust, two-way trust, transitive trust, and nontransitive trust [31].

A one-way trust is a trust relationship with a unique direction as one domain
trusts another domain. One-way trust is defined as “a trust relationship between two domains in which only one of the two domains trusts the other domain”. A two-way trust is a trust relationship with dual directions and it can be viewed as the combination of two one-way trusts with opposite directions. The two domains trust each other. Two-way trust is defined as “a trust relationship between two domains in which both domains trust each other”. A transitive trust extends trust relationships to other domains; Transitive trust is defined as “a trust relationship that flows throughout a set of domains, such as a domain tree, and forms a relationship between a domain and all domains that trust that domain”. A nontransitive trust does not extend trust relationships to other domains. Nontransitive trust is defined as “a trust relationship in a multiple-domain environment that is restricted to just two domains”. Both transitive trust and nontransitive trust can be one-way or two-way.

Realm trust is defined as “a trust between non-Windows Kerberos V5 realms, such as a UNIX realm, and Active Directory domains”. Trust relationships can be established between Windows 2000 Server or Windows Server 2003 domain and non-Windows-based operating system with Kerberos version 5 realm. Realm trust allows cross-platform interoperability based on Kerberos version 5 implementations. Realm trusts can be transitive, nontransitive, one-way, or two-way.

Forest trust is defined in a Windows Server 2003 forest. It is a one-way or two-way, transitive trust relationship by linking two disjoined Windows Server 2003 forests together. Forest trust is not transitive. It cannot be implicitly extended
to a third forest. A forest trust implies a one-way or two-way transitive trust relationship between every domain in first forest and every domain in the second forest.

In Windows Server 2003, Active Directory Domains and Trusts (Domain.msc) and Windows Domain Manager (Netdom.exe) are the two principal tools to create and manage trusts. In addition, Nltest (Nltest.exe), Network Connectivity Tester (Netdiag.exe), and the Domain Controller Diagnostic tool (Dcdiag.exe) are available to help in troubleshooting trust related issues.

Establishing trusts between two domains just means that a system in one domain can recognize a user in another domain. Domain trust does not remove all security. Domain trust allows users to be authenticated at another domain. Domain trust can be used as a tool in the administration and management of resources sharing in different Windows servers.

2.2 Trust Management Systems

The trust management problem was first identified as a distinct and important component of security in distributed information systems by M. Blaze, J. Feigenbaum, and J. Lacy in their proposed PolicyMaker [14]. M. Blaze et al claimed that “trust management, introduced in the PolicyMaker, is a unified approach to specifying and interpreting security policies, credentials, and relationships, that allows direct authorization of security-critical actions”[20]. Before the term
Background

trust management was introduced, Pretty Good Privacy (PGP) [1] and X.509 public-key certificates [56, 5] had already included the implicit notion of trust management. Trust management focuses on building a trust management layer with a new philosophy for codifying, analyzing, and managing, trust decisions in distributed information systems. The trust management covers both “why” trust is granted and “how” trust is enforced. Trust management is an important issue in security analysis and design, particularly when centrally managed security is not possible. Multiple trust management systems [14, 20, 27] have been developed to address the issue of trust management. These trust management systems help applications answer the question whether an operation can conform to the required security policies or not. These trust management systems separate generic mechanisms of trust management from application-specific policies which are defined by each application. Normally, security credentials are employed in these trust management systems to describe a specific delegation of trust among public keys. These credentials provide evidence for authorization of required actions. In most trust management systems, trust is established in a particular context. Trust management layer makes software designers and application developers consider trust management explicitly and put the design of security policies, credentials, and trust relationships, in a unified framework.

In the following section, a brief overview of several popular trust management solutions will be provided.
2.2.1 Pretty Good Privacy

PGP [7, 23] was created by P. Zimmermann as the software for secure e-mail and file encryption on the Internet. PGP targets private personal communications, and empowers people to take their privacy into their own hands [136]. PGP uses a public-key cryptography system [7, 23, 24] to enable people who have never met earlier to transmit messages securely over the Internet, to guard against unauthorized reading, and to add digital signatures on messages to guarantee their authenticity.

In PGP systems, there is no central authority that everyone trusts and no fixed or formal certification paths. PGP adopts web of trust approach [1] to establish the authenticity of the binding between a public key and a user. The concept of web of trust was first put forth by Phil Zimmermann in the manual for PGP version 2.0. Validity is the confidence about the authenticity of the binding between a public key and its owner. Validity is essential in a public key environment. A public key may be trusted directly, or trusted in some chain going back to a directly trusted root as an introducer, or by a group of trusted introducers. In PGP, individual users sign each other’s keys. The digital signature is used as the introduction of another’s key. The introducer of the key is the person who signs it. As this process proceeds, individual public keys are linked together to build a web of trust. It is more reasonable to trust a public key that has more trusted introducers. PGP products employ web of trust as
their internal certificate ‘vetting scheme’. Any PGP user can validate another
PGP user’s public key certificate by evaluating its status on the web of trust. Naturally, web of trust employs a cumulative trust model and it is a reputation
system. In the web of trust, an introducer’s reputation should be judged when a
public key certificate is evaluated. Some people are normally good as introducers
and some people are not so good.

As a reputation system, web of trust encompasses other trust models. Other
trust models can be employed to provide input information in the validating pro-
cess of a public key certificate using the web of trust. Web of trust adds the
notion that links the trustworthiness of public-key certificate and the trustwor-
thiness of its introducers. In the web of trust, a user can assign introducers at
different trust levels. In most versions of PGP, there are three trust levels to
be assigned. They are fully trusted, partially trusted, and untrusted. The vali-
dating results are classified into three levels as well. They are valid, marginally
valid, and invalid. OpenPGP uses a point system in which fully trusted has two
points, partially trusted has one point, and untrusted has zero point. If there are
enough introducers with two points, the certificate is considered valid. In PGP,
trust cascade can be used. The chains of certificates provide the base of web of
trust. PGP web of trust can be extended to allow a continuous scale of trust
assignments [81]. In PGP, individual users have the flexibility to choose different
schemes to make the trust decision.

There are several problems for the web of trust in PGP products. When
chains of introducers are used, it is necessary to trust each person of the chain to be honest and competent about signing keys. It is a very strong constraint to judge whether these introducers are likely to honestly follow the guidelines about verifying the identity of people before signing certificates. It is likely that a user with a new certificate will not be readily trusted by other users. In practice, it may be difficult to readily find one person, or several people, to endorse a new certificate. PGP does not provide formal mechanisms for the creation, acquisition, and distribution of certificates. Hence, it is unreliable for PGP to be used in applications with high level security requirements, such as military applications and serious transactions in e-commerce. PGP’s web of trust model is based on the fact that PGP is designed for personal communications and primarily used for encrypting the contents of email messages and attachments.

2.2.2 Public Key Infrastructure

A public key infrastructure (PKI) [125] is composed of security and operation policies, security services, and communication protocols, that are needed for ongoing management of keys and associated certificates in a distributed system. A PKI provides a foundation on which other security components for applications, operating systems, or networks, are built.

A PKI enables principals to be authenticated to verifiers without having to exchange any secret information in advance. Certificate Authority (CA) is a trusted authority that issues, renews, and revokes certificates. A PKI employs
one or more CAs to achieve the secure generation, distribution, and management, of public keys and associated public key certificates.

A certificate must be validated before it can be used. The validation of a public key certificate to an established trust point is proof of the existence of a chain of certificates. The established trust point is referred to as a trust root, or a trust anchor, that must be unconditionally trusted. A certification path is the path in the chain of certificates. It starts with the validated certificate and proceeds through a number of intermediate certificates up to a trusted root certificate. In the validation of a public key certificate, every certificate within that certification path must be checked. The whole process to validate a certificate is referred to as certification path processing [75].

PKIs are facilities for representing and managing trust relationships in distributed information systems. It is desirable to have suitable trust models for a PKI to enforce [98, 74, 51]. The most typical implementation of a PKI employs a hierarchical CA trust model. In a hierarchical CA model, multiple CAs are organized into a hierarchy based on a predetermined set of rules and conventions. Parent-child relationships between CAs are clearly defined. A parent CA signs children’s certificates that bind the public key of a CA to its identity. The CA at the top of the CA hierarchy is a root CA that must be trusted unconditionally. If the root CA is compromised, every other CA and certificate in the hierarchy might have been compromised. The CAs, rather than the root CA in the CA hierarchy, are subordinate CAs. If a CA uses its key to issue certificates of other
CAs, these CAs are referred to as intermediate CAs. An intermediate CA serves as a certifying authority to CAs in the hierarchy branch under it. A hierarchical CA trust model allows for an extensible, efficient, and scalable PKI.

Several trust models have been proposed that extend the basic hierarchical CA trust model. These models include rooted trust model, network trust model, hybrid trust model (based on rooted trust model and network trust model), and bridge CA trust model.

**Rooted Trust Model:** In rooted trust model, the root CA is the trust anchor. It issues and signs a certificate for itself. The Figure 2.2 shows an example of an implementations of a rooted trust model.

![Figure 2.2: An Example of Implementations of Rooted Trust Model](image)
**Network Trust Model:** In a network trust model, all CAs are trust anchors. They issue and sign certificates for themselves. Each CA can issue a cross-certificate to another CA, and a one-way trust between them can be established (naming constraints may be applied). Cross-certification is not necessary to be bidirectional. With the help of these cross-certificates, a trust network is built up. A network trust model can be viewed as a hierarchy because a cross-certificate is essentially the same as a subordinate CA certificate in a rooted trust model. The Figure 2.3 shows an example of an implementation of a network trust model.

![Figure 2.3: An Example of Implementations of Network Trust Model](image)

**Hybrid Trust Model:** A hybrid trust model can be achieved by using some form of cross-certification in a network trust model to link subordinated trust hierarchies in a rooted trust model. The hybrid trust model inherits the benefits of both rooted trust model and network trust model. In the hybrid trust model, there are multiple root CAs and all non-root CAs are in a trust hierarchy. Cross-
certification is normally defined at the root level, but selective cross-certification between non-root CAs is permitted. The hybrid trust model provides the ability to construct arbitrary trust paths in a simple manner. It becomes possible to introduce direct “short-cut” for frequently used paths by joining lower points within the hierarchies using cross-certifications.

**Bridge CA Trust Model:** In a bridge CA trust model, there are multiple CA hierarchies and one or more bridge CAs. A bridge CA behaves as a peer of trust roots, but itself is not a trust root. It only provides cross-certificates for the roots of trust hierarchies or other bridge CAs to link them with each other. The cross-certification is controlled by bridge CAs and name constraints can be applied to it. Bridge CA Trust Model can be viewed as a special case of the hybrid model discussed above.

There are some alternatives of hierarchy based trust models. The most popular ones among them are the web of trust and the trust list model. We have already discussed web of trust model earlier in Section 2.2.1. In the trust list model, it is not necessary to have trust hierarchy. An application has its CA trust list that is composed of all CAs that the application trusts. The listed CAs are accepted a set of trusted roots. All certificate chains terminate at one of these trusted roots. The specification of an application must define that it uses a CA trust list, and trust list must be defined before the application performs the trust validation.
ITU-T X.509 is the broadly accepted standard of public key certificates [96, 92, 93, 94]. Public Key Infrastructure X.509 (known as PKIX) Working Group, which is under the Internet Engineering Task Force (IETF), has developed a series of standards to support X.509-based PKI on the Internet [54, 56, 135, 109, 5, 86, 110, 4, 87, 55].

2.2.3 PolicyMaker and KeyNote

PolicyMaker and KeyNote are trust management systems developed by AT&T Research [14, 20, 13]. The PolicyMaker is the first one to be explicitly claimed as a trust management system. Being independent of any particular application or service, PolicyMaker [14] is designed as a general tool for the development of services with features of privacy and authenticity. KeyNote [12, 20] is the successor of PolicyMaker and is more extensible and expressive. Both PolicyMaker and KeyNote can be embedded into applications as relatively an independent module, or run as a “daemon” service. PolicyMaker serves applications as a query engine.

There are important theoretical contributions in the trust management approach proposed by developers of PolicyMaker and KeyNote. PolicyMaker is actually designed as an illustration system of the trust management approach. The trust management problem is identified explicitly as a general problem and studied in its own right. A general trust management layer, with an appropriate level of abstraction, is introduced with the idea to separate generic trust mechanisms from application specific policies. The concept of trust management is
introduced as a unified approach to deal with security policies, security credentials, and trust relationships. The security policies are specified by the application and interpreted by the trust management system. A trust management system provides a standard interface for applications to request compliance checking for actions against local security policies. The authorization of security-critical actions is assumed to be controlled by one or multiple credentials. The access rights can be bound to a public key directly. The previously defined certificate frameworks such as X.509 and PGP only provide mechanisms for identity authentication that bind a public key to its owner, and the binding of access rights to the key’s owner occurs outside the certificate framework. Trust management engines employ a programming language to express privileges and restrictions. It is not necessary to resolve identities in an authorization decision.

The basic function of PolicyMaker is to evaluate whether a particular query, or proposed action, is compliant with local policies or not. In PolicyMaker, the action under consideration is represented by an action string. An action string is an application-specific message that describes the trusted action requested by one or more principals. The semantics of action strings are determined by the applications. Policies represent trust requirements. Policies are application-specific and under local control. Credentials represent statements from others (normally from the entities who request for the action to be authorized). Policies and credentials are collectively referred to as assertions. There is a compliance checker at the heart of PolicyMaker. The compliance checker [17] performs compliance
checking based on the input which are composed of policies, credentials, and action string. A Yes/No answer, or additional restrictions, will be returned depending on where the credentials constitute a proof that the request complies with the policies.

Medical information systems [15] and an PICS information-labelling system [16] have been chosen to demonstrate the power and adaptability of PolicyMaker as a trust management system. PICS stands for Platform for Internet Content Selection which is a scheme for rating and labelling resources that is machine-readable [103].

KeyNote [20, 12] is the successor of PolicyMaker and it has the same design principles as that of PolicyMaker. Both PolicyMaker and KeyNote employ trust management credentials for the direct authorization of security-critical actions. These credentials bind keys to the authorization of specific actions. The KeyNote architecture and language are more friendly to Internet protocols/services and public key infrastructure. Compared with PolicyMaker, KeyNote is simpler and takes into account about standardization and ease of integration into applications.

In KeyNote, an action is described by a set of attribute/value pairs which is referred to as an action attribute set. An action attribute set is passed to the KeyNote compliance checker with each query. The semantics of attributes and values of these attributes are defined in applications and are not interpreted by KeyNote. KeyNote’s action attribute sets have more expressive power and are
more standard than the ActionString in PolicyMaker.

KeyNote assertions always return a boolean answer (authorized or not). Such an arrangement is simpler and removes some possible issues of the returned additional restrictions in PolicyMaker. A standard signature verification belongs to general purpose mechanisms. It is more suitable to put a standard signature verification in a trust management engine. Credential signature verification is built into the KeyNote engine.

In Policymaker, the filter is not well-defined. To remove the possible confusion by filter, KeyNote predicates are written in a simple notation based on C-like expressions. Any assertion language can be used in PolicyMaker and this causes a negative effect for PolicyMaker’s compliance checker to integrate with applications. KeyNote uses its predefined assertion language. The assertion syntax is based on a “RFC-822” [32] style syntax which is human-readable (it is originally designed for Internet text message). The integration of applications and compliance checker is simpler.

In KeyNote, policies and credentials have the same format but policies are locally trusted. The principals are explicitly specified in an assertion in the Licensees field. The Conditions field includes the programs for the tests of action environment variables. In compliance proofs, Licensees field and Conditions field contain the required information for satisfying the assertion. KeyNote uses a depth-first search algorithm that attempts recursively to satisfy at least one policy assertion. KeyNote’s evaluation model is a subset of PolicyMaker’s model. The
Background

compliance checking algorithm is more efficient than the one in PolicyMaker.

The assertion syntax and compliance-checking algorithm in KeyNote are sub-
sets of the corresponding ones in PolicyMaker. KeyNote is more specific and it is
possible to use PolicyMaker as its extension. Demonstration examples of KeyNote
system have been published for dealing with trust management in network layer
protocols [18] and IPsec [21]. Both KeyNote and PolicyMaker do not enforce
policies. They are only query engines to provide advice for the applications that
call them.

2.2.4 REFEREE

REFEREE [27] is the acronym of Rule-controlled Environment for Evaluation
of Rules and Everything Else. REFEREE is designed as a trust management
system for web applications. In the web environment, both web clients and
web servers have critical trust issues. On the web, there are some sensitive and
high value web transactions that require a strict proof of security; meanwhile
there are some applications or web resources which can be accepted based on
weaker forms of evidence. For example, a recommendation from a close friend
may convince someone to trust that a piece of software is virus-free. As a trust
management system, REFEREE follows the design principles of PolicyMaker [14]
and employs PICS label [103] credential to state some properties of an Internet
resource. Trust decisions are recommended by the compliance checker based on
the actions requested, credentials provided, and the policies satisfied.
REFEREE uses a rule based policy language, profile-0.92, for writing policies. REFEREE provides a general policy-evaluation mechanism for both web clients and web servers. In REFEREE, every action is controlled by some policy, particularly the evaluation of compliance with policy is controlled by some policy as well. REFEREE is a system for writing policies about policies, cryptographic keys, PICS label bureaus, certification authorities, trust delegation, or anything else. REFEREE is an environment for evaluating compliance of the requested actions with the specified policies. REFEREE differs from PolicyMaker in placing everything under policy control. Here the “everything” includes both the evaluation of compliance with policies and the evaluation of credentials. REFEREE allows the trust management engine to fetch additional credentials and perform signature verification in the process of request evaluation. REFEREE allows non-monotonic policies and credentials, whereas PolicyMaker and KeyNote only support monotonic policies. The compliance checker of REFEREE supports inter-assertion communication and allows assertion programs to call each other. REFEREE is a module-based system which uses a collection of modules as basic building blocks. Each module is designed for a particular policy evaluation. A module can call other modules to delegate subtasks.

REFEREE uses statement lists and tri-values for its programs. Each program takes an initial statement list as input (may take additional arguments as well). The program uses a tri-value and a statement list as the output. A statement list is a collection of assertions expressed in a particular format. A REFEREE
Background

statement has a two-element structure consisting of content and context. The context determines how the content is to be interpreted. The context must be understandable by both the calling application and REFEREE. The tri-value includes true, false and unknown. A program may call other programs during its execution.

The policy language profiles-0.92 is designed to work with W3C PICS. It has a language construct invoke for calling another REFEREE program. The name of the invoked program is added to the context portion of each returned statement and these returned statements are then appended to the statement list. The information about invoked programs is recorded in the statement list. Profiles-0.92 supports the invocation of load-labels program to look for PICS labels and write REFEREE statements after parsing the found labels. Profiles-0.92 has operations AND, OR, and NOT for two tri-values. It has a pattern-matcher that can be used to search a statement list for statements with a particular form.

2.2.5 IBM Trust Establishment

IBM Trust Establishment (TE) [60, 53] is a trust management system for e-business where the involved parties are not known in advance, and some trust can be established based on public key certificates that provide references obtained from third parties. TE system supports the establishment of dynamic ad-hoc relationships based on ‘web of trust’, that is, accepting recommendations from community/networks rather than requiring a predefined hierarchy with one or
more trust roots (such as PKI). As an extension of the role-based access control systems, TE system provides a mechanism that allows a business to define a policy for mapping accessed users to roles based on certificates received from the user and/or collected automatically by the system.

In a classical role-based access control system, a static table is used to map a subject to a given role. Everything is predefined. In the TE system, business roles are predefined but the role assignment for a user is dynamic. The role assignment is under the control of trust policies. TE system collects public key certificates according to the requirements of trust policies. A role is assigned to an unknown user based on the subject’s certificates, trust policies, and public key certificates collected. Then the assigned role is fed as input to the classic role based access control systems. TE system supports privacy because it is not necessary to know the user identity in order to map it to a role.

The trust policies are written in Trust Policy Language (TPL). The TPL syntax is written in XML which programmers and administrators are familiar with. A trust policy defines a group by a set of rules. Groups can represent roles in a role based access control system. A rule defines a set of certificates necessary to join a group and further conditions on the attributes of these certificates. The policy DTD (Document Type Definition) can be found at [60].

IBM has implemented the TE system using Java and X.509v3 certificate format. The implemented TE system has an API toolkit that can be used to extend the access control abilities of existing applications or web servers. TE_1.1.0 is the
current version of IBM TE system. It is claimed to be a lightweight policy evaluator module based on the JDK standard X.509 certificate API.

### 2.2.6 Trust Negotiation Approach

Trust negotiation, sometimes referred to as automated trust negotiation as well, is a promising approach that enables the establishment of trust between entities, without enough prior knowledge of each other, through an iterative exchange of digital credentials. Trust negotiation normally occurs in open systems, such as the Internet, for the purpose of sensitive interactions across different security domains.

There has been much research addressing the underlying theory and required policy languages for trust negotiations [127, 114, 133, 132, 9, 118, 126]. Trust negotiations have also been studied in a broad range of contexts such as web services [116], semantic web services [95], digital library web services [117], peer-to-peer systems [130], and healthcare information systems [123]. Researchers have developed a prototype system, called TrustBuilder [128, 113, 119], for negotiating trust across organizational boundaries. The architecture of TrustBuilder incorporates trust negotiation into standard network technologies such as HTTP, SSL/TLS, and IPSec.

In this section, a review is provided of the general characteristics of trust negotiation, trust negotiation policy languages, trust negotiation policy compliance checkers, and other system level requirements for trust negotiation systems.
The trust negotiation problem originates from the access control challenges when strangers are involved in sharing resources, or conducting business transactions, in open environments. A client can attach appropriate credentials to service requests for the purpose of authorization, and the client may require some credentials from the server to have some level of trust on the server. Possibly, these credentials include sensitive information and they cannot be delivered unconditionally. These credentials are released based on conditions the counterpart can satisfy. These conditions are defined in the disclosure policies of these credentials. Disclosure of sensitive credentials is regulated by these policies. The disclosure of a credential may require a set of credentials from the counterpart and which can, in turn, require other credentials. Less sensitive credentials are disclosed earlier for higher levels of mutual trust to be established. More sensitive credentials are disclosed based on the satisfaction of their disclosure policies that may require a set of credentials from the counterpart. Credentials are exchanged iteratively and trust is incrementally established. In successful trust negotiations, trust requirements on both sides are satisfied for the target business transaction or resource sharing. Trust negotiation approach enables resource requesters and access mediators to establish trust with each other through iterative disclosures of sensitive credentials.

The general methodology for developing trust negotiation solutions is composed of two important technical aspects, namely the policy languages and the policy compliance checkers.
The language requirements for policies in trust negotiation solutions have been discussed by researchers in [114, 9]. A policy language for trust negotiations should have the following features:

- **Well Defined Semantics:** The meaning of a policy written in the language should be independent of the language’s implementation.

- **Monotonicity:** If a set of credentials can satisfy the disclosure policy of a resource, then the disclosure of additional credentials can satisfy the disclosure policy of the resource.

- **Credential Combinations:** The policy language must have expressive ability to require submission of combinations of credentials with conjunction and disjunction operators.

- **Constraints on Attribute Values:** The policy language must have the ability to constrain submitted credentials as a certain type and restrict their attribute values.

- **Inter-Credential Constraints:** The policy language must have the ability to express constraints on values in multiple credentials of the same subject even when they use different keys.

- **Credential Chains:** The policy language should have the expressive power to describe and constrain chains of credentials. Particularly, the constraints
on the number or type of intermediate links in credential chains can be expressed.

- **Authentication**: The policy language must support explicit specification of authentication requirements either as part of the language itself or as external function calls.

- **Sensitive Policy Protection**: The protection of sensitive policies can be handled at policy language level or a system level.

- **Unified Formalism and Interoperability**: The policy language should support interoperability among negotiation participants that may run in different environments.

The major requirements of policy compliance checkers for trust negotiations have been identified by K. E. Seamons et al [114]. There are two operation modes of a compliance checker in trust negotiations. In order to illustrate these two operation modes, let us assume that Alice and Bob are the two parties involved in a trust negotiation.

**Server Mode of Compliance Checker**: In the first operation mode of a compliance checker, Alice’s negotiation manager invokes Alice’s compliance checker to evaluate Bob’s request for the access of a sensitive resource of Alice. Alice’s local policy, Bob’s disclosed credentials, and Bob’s request are the input to Alice’s compliance checker. Alice’s compliance checker produces the result to indicate
whether or not Bob’s credentials can satisfy the policy. When access is denied, some compliance checkers may provide a justification. For Bob’s request, Alice’s compliance checker behaves as a server. This is why we refer to this operation mode as the server mode. In a trust negotiation, Alice and Bob are peers and Alice can also request Bob for the access of Bob’s sensitive credentials, policies, or resources. In the server mode of a compliance checker, the request party and the owner of the compliance checker are always different. See Figure 2.4 (from Ref. [114]) for the server mode of compliance checker.

**Figure 2.4: Server Mode of Compliance Checker**

**Client Mode of Compliance Checker:** When Alice can not grant Bob’s access request to a sensitive credential or resource because Bob has not provided sufficient credentials, Alice discloses her access control policy of the sensitive resource to Bob. Bob’s negotiation manager invokes Bob’s compliance checker to evaluate Bob’s request against the disclosed policy of Alice. Alices disclosed pol-
icy, Bob’s local credentials, and Bobs request are the input to Bob’s compliance checker. Bob’s compliance checker produces the result to indicate whether or not Bob’s credentials can satisfy the policy. For a positive result, Bobs compliance checker also returns a set of local credentials that satisfies Alice’s policy. Bob discloses these credentials to Alice to continue the trust negotiation. In contrast to the server mode, this operation mode is referred to as the client mode. Bob’s request is evaluated by Bob’s compliance checker. In the client mode of a compliance checker, the requester and the owner of the compliance checker is the same party. See Figure 2.5 (from Ref. [114]) for the client mode of compliance checker. In order to support the functions of the policy compliance checker and trust nego-

![Figure 2.5: Client Mode of Compliance Checker](image)

...tiation protocols, a trust negotiation system normally has a series of system level computing components to prove credential ownership, to check credential valid-
ity, to support credential chain discovery, to support trust negotiation strategies, and to maintain credential sequences for reuse.

A trust negotiation system could be very complicated. It is necessary to have advanced functions to deal with credential chains, authentication of multiple identities, and complex compliance checker modes. Trust negotiation systems running on the Internet must address the issues of interoperability and scalability. A complicated trust negotiation system can have conflictive requirements and it is normally a challenging task to trade off them.

2.3 Summary

This chapter has presented an overview of several major conceptions of trust and trust management systems. The concept of trust has a deep root in its everyday usage and in social sciences. The concept of trust in social sciences is introduced at first. The notions of trust in the computing world in some sense provide extensions to this basic concept. This chapter provides an overview on trusted system and trusted computing, S. Marsh’s formalization of trust, trust vs reputation, Web Services Trust, trust management, and Microsoft’s domain trust. Then, there is a brief review of some of the popular trust management systems proposed so far. These trust management systems include PGP, PKI, PolicyMaker, KeyNote, REFEREE, and IBM trust establishment. A discussion of trust negotiation approach has also been provided.
Chapter 3

Unified Taxonomy Framework of Trust

This chapter introduces a formal model of trust relationship and a new unified taxonomy framework of trust. The formal definition of trust relationship captures a range of commonly understood notions of trust and it is the cornerstone of the research in this dissertation. In order to have a clear understanding of trust relationships, a taxonomy framework of trust for categorizing and describing trust relationships is provided. The new taxonomy framework is built on a formal model of trust where a range of useful trust relationships can be expressed and compared. The taxonomy framework reflects the different forms of trust relationships based on their specific characteristics.

The taxonomy framework of trust includes the classification of trust, the properties of trust relationships, and relations of trust relationships. For the properties of trust relationships, the taxonomy framework specifies the properties about trust scope and diversity, trust direction, and trust symmetry. The taxonomy framework provides terminologies and scenarios to describe a broad
range of situations of trust relationships in distributed information systems. It will provide a solid basis for the analysis and modelling of trust relationships.

The formal definition of trust relationship is outlined in Section 3.1. The classification of trust is provided in Section 3.2. Trust scope and diversity is described in Section 3.3. Trust direction and symmetry is described in Section 3.4. The relations of trust relationships are presented in Section 3.5. A summary of this chapter is provided in Section 3.6.

3.1 Formal Model

In the computing world, the concept of trust arises in many branches of computing systems. There is not a clear consensus about the meaning of trust. When the term of trust is used, its meaning must be judged based on the particular domain where it is used. In the last chapter, several important conceptions of trust were reviewed. Multiple notions of trust in computing make trust complex, multifaceted, and context-dependent. The broad generality of the term trust makes the concept of trust abstract and somewhat elusive. Trust is a popular term and plays an important role in information systems, particularly in the security paradigm. It is believed to be necessary to build up a solid taxonomy framework which can be used to describe the various characteristics of trust and clarify the difference between them.

In order to capture the essence of trust, the starting point is to provide a
formal definition of trust relationship. The concerns about trust and trust relationship are within the scope of information technology, particularly in the context of information security.

Though trust has been a foundational stone for security, it has been a difficult concept to define clearly. Actually, several researchers have studied the concept of trust in the secure computing world. For example, S. Marsh [80] has described situational trust as a representation for the amount of trust that an agent has in another in a given situation (see section 2.1.3). T. Grandison and M. Sloman [49] have defined trust as “the quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context”. F. Hussain, E. Chang, and T.S. Dillon [10, 59] have defined trust as “the trusting peers belief in the trusted peers willingness and capability to behave as expected by the trusting peer in a given context at a given time slot in a particular association relationship”. These definitions are definitely helpful to make the concept of trust more clear. However, these definitions follow the traditional trend of using general terms and normally their implementations only cover narrow topics such as agent systems. In particular, they are only used to analyse/model some trust phenomena, but there is no trust management framework based on these definitions.

In this dissertation, a formal definition of trust relationship is outlined that can be used as a formal model of trust. This formal definition provides the kernel concept and the foundation of the unified framework of trust management
which will be described in Chapter 5. The definition of trust relationship is based on specific requirements for the concept of trust in the information security paradigm. The definition of trust relationship should satisfy the following requirements:

- The definition should provide a solid base to analyze both commonly used as well as certain unique trust notions that arise in distributed computing. The majority of existing conceptions of trust can be covered.

- The definition should have strong expressive power while, at the same time, being simple.

- The definition should have a strict mathematical structure which can be implemented for the purpose of computing.

- The definition should provide a solid foundation to define properties of trust in a broad variety of situations.

- The definition should capture concrete (“hard”) security mechanisms but have the capability to embrace “soft” security mechanisms (such as reputation).

“Hard” security is used to imply concrete security with complete certainty. For instance, mechanisms of hard security in access control have the general property of allowing complete access or no access at all. Common technologies for hard security include passwords, firewalls, cryptographic algorithms for authen-
tication and/or authorisation. In contrast, the notion of soft security is based on social control such as reputation [100]. In soft security, trust is inferred from the related reputation. Many reputation-based systems have been proposed and implemented (see Section 2.1.4). These reputation systems compute the reputation of an involved entity by aggregating the perception of other entities in the system. Normally, these systems are limited in the sense that they do not link the reputation to the required control inferred from the reputation. The definition of trust relationship should follow hard security mechanisms but have the ability to embrace soft security mechanisms. For example, reputation can be used as input information in the evaluation of conditions of the trust relationship.

It is clear that trust can not be understood as a simple bilateral relation between trustors and trustees. The syntax of a trust relationship needs to include the context information as specified conditions, as well as specified target properties. A trust relationship should be under a set of specified conditions, a set of trustors trust that a set of trustees have a set of specified properties (that is, the set of trustees will/can perform a set of actions or have a set of attributes). Our formal definition of trust relationship is expressed as follows:

**Definition 1** A trust relationship is a four-tuple $T = < R, E, C, P >$ where:

- $R$ is the set of trustors. It cannot be empty.
- $E$ is the set of trustees. It cannot be empty.
- $C$ is the set of conditions. It contains all conditions (requirements) for the
current trust relationship. Normally, a trust relationship has some specified conditions. If there is no condition, the condition set is empty.

- $P$ is the set of properties. The property set describes the actions or attributes of the trustees. It can not be empty. The property set can be divided into two sub sets:

  - Action set: the set of actions which trustors trust that trustees will/can perform.
  
  - Attribute set: the set of attributes which trustors trust that trustees have.

When trust relationships are used, the full syntax (four-tuple $< R, E, C, P >$) must be followed. Trust relationship $T$ means that under the condition set $C$, trustor set $R$ trust that trustee set $E$ have property set $P$.

There are some specific cases of the trust relationship when some involved sets include nothing (empty set) or anything (whole set of possible entities). These specific cases have special meanings and are crucial in the understanding of the definition of trust relationship. These specific cases play important roles in the real world. Here are five specific cases of trust relationships:

1. $R$ is ANY. Trustor set includes all possible entities. All possible entities trust that the set of trustees $E$ have the set of properties $P$ under the set of conditions $C$. 
2. \( E \) is \textit{ANY}. Trustee set includes all possible entities. All possible entities can be trusted to have the set of properties \( P \) by the set of trustors \( R \) under the set of conditions \( C \).

3. \( C \) is \textit{EMPTY}. There is no condition in the trust relationship. The set of trustors \( R \) trusts that the set of trustees \( E \) has the set of properties \( P \) without any condition.

4. \( P \) is \textit{ANY}. The property of the trustee can be anything. The set of trustors \( R \) trusts that the set of trustees \( E \) has all possible properties under the set of conditions \( C \).

5. \( C \) is \textit{EMPTY} and \( P \) is \textit{ANY}. The set of trustors \( R \) trusts that the set of trustees \( E \) has all possible properties without any condition. This case happens when the set of trustors \( R \) trusts the set of trustees \( E \) by default.

The proposed formal definition of trust relationship has a strict mathematical structure and a broad expressive power. The frequently used existing conceptions of trust can be expressed as specific cases of the formal definition of trust relationship, where some parts of full syntax have been omitted. Informal notions of trust commonly deal with conditions and properties as implicit information. For example, when it is stated that “Alice trusts Bob”, the set of properties are implied in Bob’s personality and behavior; the set of conditions is assumed to be the normal set of circumstances. The statement such as “Alice trusts Bob” is acceptable in everyday life because the context of the statement can provide
complementary information and often the accuracy is not very important. If the full syntax of the representation is not used, then it can lead to lack of clarity and misunderstanding. In the computing world, it is necessary to remove context-based implications and assumptions. The formal definition of trust relationship helps to avoid potential confusions caused by these context-based implications and assumptions.

The formal definition of trust relationship can cover many of the popular conceptions of trust that were introduced in Section 2.1. In Marsh’s situational trust, the concept of situation is too general, the formal definition of trust relationship uses the condition set and property set to replace the situation, and the meaning of trust becomes more clear. WS-Trust focuses on requesting and issuing security tokens, and brokering trust relationships. The formal definition of trust relationship is able to represent WS-Trust and its further extensions. This aspect will be addressed and more details will be provided in Chapter 7. Following the whole syntax of trust relationship, a one way trust can be viewed as: users in the trust domain, trust users in trusted domain, without any condition, that users in trusted domain have the right to access the set of resources in the trust domain (see more details in Section 3.4).

In the formal definition of trust relationship, condition set and property set are used to define the specified context. The condition set defines the associated requirements and the property set defines target properties in a trust relationship. The property set includes two subsets, namely action set and attribute set.
Condition set covers a variety of situations and mechanisms. The conditions can be classified as pre-conditions and post-conditions. Pre-conditions can include existing facts such as credentials from trustees, stored data of trustors, environment parameters, and community based reputation. Post-conditions can include commitments or evidences of the current actions such as committing statements with trustees’ signatures. Both hard security mechanisms and soft security mechanisms are embraced by the condition set. Pre-condition set provides the filter function to accept evidences and avoid risks. In order to capture information as evidences or risks, the condition set may have a broad variety of interfaces for obtaining data from data storage, applications, and systems.

The format of trust relationship enables to define security policies based on one or multiple trust relationships. However, trust relationships are different from security policies. Security policies are normally defined for local resources. Trust relationship is a more general concept which focuses the statement of facts rather than that of control. The property set in a trust relationship covers not only actions, but also attributes. Some trust relationships can be converted to security policies. Some other trust relationships do not have any counterparts in security policies. For example, a trust relationship with attributes as its property set is a pure statement of some fact and it can not be used as a control policy. Security policies are normally closely bound to the resources that they protect. Trust relationships can be more independent. The trust solution is more flexible than existing solutions based on security policies. In this solution, the
analysis/modeling, evaluating, and consuming of trust, are regarded as separated processes in distributed information systems (see Chapter 4 and Chapter 5).

The formal definition can be used to capture trust relationships in different security scenarios related with security domain, role based access control, and mandatory access control. For example, Microsoft’s domain trust can be viewed as trust relationships when trustor set is mapped to trustor domain, trustee set is mapped to trustee domain, the condition set is the required authentication, the property set is the access of the resources in the trustor domain. With role-based access control, permissions are assigned to roles and then users are assigned to the specified roles [40, 108]. Similar ideas have led to the use of trustor set and trustee set, rather than single trustor and trustee, in the formal definition of trust relationship. Section 7.2 provides more discussion about the relationship between our trust management approach and the role based access control models. The security labels enable putting the resources or actions into groups that provide a suitable level of abstraction to be used in access rules. In the formal definition of trust relationship, the property set can be viewed as a kind of security label that can provide a similar abstraction level.

When a trust relationship is used in the real world, trustors, trustees, and properties, are normally involved individually. The trust relationship will be evaluated based on one trustor, one trustee, and one property. It is a runtime task to map individual trustor, trustee, and property, to corresponding sets in a trust relationship. Section 4.2 defines the instance of a trust relationship to
make the above scenario more clear. However, from the view point of analysis and design, it is convenient to define trust relationship with trustor set, trustee set, condition set, and property set as four tuples.

The formal definition of trust relationship provides the foundation for the unified taxonomy framework of trust which will be introduced in the following sections of this chapter.

### 3.2 Trust Classification

This section describes different types of trust. The research about trust focuses on trust relationships in distributed information systems from the perspective of information security, and the trust classification described in this section will reflect this concern. Grandison and Sloman [48] have given a bottom-up classification of trust. They used terms such as resources access trust, service provision trust, certification trust, delegation trust, and infrastructure trust. Obviously, there are some trust relationships that are beyond these types. However, these types are important ones and they can cover most situations of trust in distributed information systems. Under the taxonomy framework of trust, they are expressed as follows:

- **Resources Access Trust**: Resources access trust relationship is a kind of trust relationship for the purpose of accessing resources. The access control has been the central concern of information security for many decades. The
trust relationship can be refined into authorization policies that specify actions the trustee can perform on the trustor’s resources and constraints that apply, such as the time periods, for which the access is permitted. With the syntax of formal definition of trust relationship, resource access trust will be like “the trustors trust trustees under some conditions that trustees have the right to get access to some of trustors’ resources”. Resource access trust is the most important trust type in distributed information systems which is normally referred to as authorisation. Authorisation guarantees the protection of resources such as data and applications by only allowing appropriate entities to access these resources. Multiple tasks of authorisation include identifying the required resource access trust, setting up security policies based on required resource access trust, assessing the satisfaction of resource access trust, and permitting or refusing requests of resources.

- **Service Provision Trust**: Service provision trust describes trustors’ trust in the provided services or resources of trustees. It is related to protection from maliciously or unreliably provided services or resources. With the syntax of formal definition of the trust relationship, service provision trust will be like “the trustors trust trustees under some conditions that trustees will provide the claimed services”. Service provision trust is particularly important in the paradigm of web services.
• **Certification Trust:** Certification trust is based on certification of the trustworthiness of the trustee by a third party. Certification trust is related to a special form of service provision trust. Certification authority is, in fact, providing a trust certification service. With the syntax of formal definition of trust relationship, certification trust will be like “trustors trust trustees if trustees can provide certificates that trustees have a set of attributes or can do a set of actions according the certificates.” The related service provision trust of certification trust will be like “trustors trust certification authority under some conditions that the certification authority will only give certificates to suitable entities”. Certification trust is the foundation of PKI (see Section 2.2.2) which plays an important role in secure distributed information systems.

• **Delegation Trust:** Delegation trust is a special form of service provision trust [36]. With the syntax of formal definition of trust relationship, delegation trust will be like “trustors trust trustees under some conditions that trustees can make decisions on trustors’ behalf, with respect to resources or services that the trustors own or control”. Compared with other trust types, delegation trust is more complicated because there are more than two parties involved. A broad range of delegation mechanisms has been developed and implemented in distributed information systems.

• **Infrastructure Trust:** Infrastructure trust is a kind of trust that trustors
trust some base infrastructure under some conditions [3]. With the syntax of formal definition of trust relationship, infrastructure trust will be like “trustors trust base infrastructure under some conditions for a set of properties of the infrastructure (some actions and attributes)”. For example, a trusted computing system (see section 2.1.2) targets at infrastructure trust that covers all behaviors of hardware and software in the system.

All these trust types must build on a more basic trust relationship, which is, the authentication trust or identity trust. Authentication trust is “trustors trust trustees under some condition that trustees are who they claim to be”. Under the taxonomy framework, trust relationships are categorized into two layers and provide a new hierarchy of trust relationships. Authentication trust belongs to a separate layer and all other trust types belong to another layer above it. This is illustrated in Figure 3.1.

<table>
<thead>
<tr>
<th>Second Layer</th>
<th>First Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Access Trust</td>
<td>Authentication Trust</td>
</tr>
<tr>
<td>Service Provision Trust</td>
<td></td>
</tr>
<tr>
<td>Certification Trust</td>
<td></td>
</tr>
<tr>
<td>Delegation Trust</td>
<td></td>
</tr>
<tr>
<td>Infrastructure Trust</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.1**: Trust Layers

Note that trust types of layer two may not be necessarily specified in terms of an identity. Anonymous authorization belongs to access trust, and it is an example that there is no specified identity. Anonymous authorization can be im-
implemented using certificates with capabilities. The real identity of the involved
trustee will not be revealed. For example, a customer has a certificate for ac-

cessing some resources on the Internet. The customer’s behavior of accessing the
resources can be recorded. For some resource access trust and service provision
trust, the anonymity of customers is required. In a situation of anonymous au-
thentication, the layer of authentication still needs to provide a mechanism to
deal with the same entity as the trustee in the whole scope of the trust process
(such as a session or a transaction). Normally, there is a temporary and dynamic
identification that will be uniquely connected with the involved trustee in the
scope of the trust process.

The above two layer trust can be found in systems such as Microsoft Windows
Servers that control their resources based on authentication and authorization.
The authentication and authorization are related with each other and the resource
access trusts have a typical two-layer structure.

Trust relationships at layer two can be classified in different ways. In the
following part of this section, another kind of classification is given that is different
from the bottom-up classification by Grandison and Sloman [48]. Based on strict
definition of trust relationship, trust relationships at layer two can be classified
according to the nature of the trustees in a trust relationship \(< R, E, C, P >\). If
\(E\) is an infrastructure, the trust relationship belongs to infrastructure trust. If \(E\)
is not an infrastructure, the trust relationship belongs to noninfrastructure trust.
Noninfrastructure trust relationships can be classified based on the ownership of
the property set. If the trustors have the ownership of the property set, then the trust relationship belongs to access trust. If the trustees have the ownership of the property set, then the trust relationship belongs to provision trust. If some properties are owned by trustees and some other properties are owned by trustors, then the trust relationship belongs to a mixture of access and provision trust. The hierarchy of trust relationships at layer two is illustrated in Figure 3.2. In such a classification, delegation trust and certification trust are not independent types. As discussed, the delegation trust is a special form of provision trust; trustees are the providers of delegated decisions on behalf of trustors. A certification trust can be any subtype of noninfrastructure trust based on the nature of its property set.

\[
\begin{align*}
\text{Trust Relationship} & : \\
\text{Infrastructure Trust} & : \\
\text{Non-infrastructure Trust} & : \\
\end{align*}
\]

\[
\begin{align*}
\text{Infrastructure Trust} & : \\
\text{Access Trust} & : \\
\text{Provision Trust} & : \\
\text{Mixture(A&P) Trust} & : \\
\end{align*}
\]

\textbf{Figure 3.2:} Trust Hierarchy
3.3 Scope and Diversity

Scope and diversity are two other related aspects of a trust relationship. The diversity of trust has been discussed by Audun Jøsang [62], who expresses trust in three dimensions of diversity. The first dimension represents trustors or trust originators; the second dimension represents the trust purpose; and the third dimension represents trustees. Jøsang [62] uses the term trust purpose based on the observation that trust is relative to a domain of actions. In our formal definition of trust relationship, trustors and trustees are two tuples and they are similar to the terms used by Jøsang. The origin diversity about trustors and target diversity about trustees are straightforward. In this section, trust scope label will be defined to take the place of the trust purpose. The benefits of trust scope label will be discussed later in this section. The trust scope label is based on the four tuples of a trust relationship and it is the binding of the condition set and property set. The trust scope label is a new element of the taxonomy framework and it is defined as follows:

**Definition 2** A trust scope label is a two-tuple \( SL = \langle C, P \rangle \) where \( C \) is a set of conditions and \( P \) is a set of properties.

The details of condition set \( C \) and property set \( P \) can be found in the formal definition of trust relationship in Section 3.1. Actually, trust scope label provides a new layer of abstraction under the trust relationship and it defines the properties of the trust and its associated conditions. To compare two trust scope labels
Unified Taxonomy Framework of Trust

\( TSL_1 = \langle C_1, P_1 \rangle \) and \( TSL_2 = \langle C_2, P_2 \rangle \), the following rules apply:

1. \( C_1 \subseteq C_2 \) and \( P_1 \supseteq P_2 \) \( \iff \) \( TSL_1 \geq TSL_2 \);

2. \( C_1 = C_2 \) and \( P_1 = P_2 \) \( \iff \) \( TSL_1 = TSL_2 \);

3. \( C_1 \supseteq C_2 \) and \( P_1 \subseteq P_2 \) \( \iff \) \( TSL_1 \leq TSL_2 \).

4. In other cases, \( TSL_1 \) and \( TSL_2 \) can not be compared with each other.

The trust scope label is beyond the trust purpose in several aspects. Trust scope label is composed of a subspace of trust relationships (two tuples out of four tuples) and describes the characteristics of the combination of condition set \( C \) and property set \( P \). Trust scope labels could be treated as an independent subspace of trust relationships in the analysis and design of overall information systems. The property set in trust scope label covers not only actions but also attributes of trustees. Trust scope labels can be embedded in all the trust types described in Section 3.2, and two trust scope labels could be compared with each other based on these rules.

**Scenario Example I**: Consider an online software shop. It is assumed that anybody who wants to enter the online shop must register as a member of the online shop first. For describing the condition set and property set in possible trust relationships between the shop and possible customers, the following notations are used:
• p1 stands for that customers can read the documentation of the software.

• p2 stands for that customers can download the software.

• c1 stands for certificate of membership.

• c2 stands for the commitment of the payment for the software.

• c3 stands for the payment for the software.

The following trust scope labels are applicable:

1. \( TSL_1 = \langle \{c1\}, \{p1\} \rangle \)

2. \( TSL_2 = \langle \{c1, c2\}, \{p1, p2\} \rangle \)

3. \( TSL_3 = \langle \{c1, c2, c3\}, \{p1, p2\} \rangle \)

Based on the rules to compare two trust scope labels:

• \( TSL_1 \) cannot be compared with \( TSL_2 \) (or \( TSL_3 \)). There is no obvious relationship between \( TSL_1 \) and \( TSL_2 \) (or \( TSL_3 \)).

• \( TSL_2 > TSL_3 \). It means that the trust scope of \( TSL_2 \) is less strict than that of \( TSL_3 \).

In the analysis and modelling of trust relationships, some trust scope labels may be quite complicated and the above comparison rules can be used as tools in making judgements.
3.4 Direction and Symmetry

The properties of trust direction and trust symmetry play an important role in information systems. In this section, a set of definitions is provided for the properties of trust direction and trust symmetry and scenario examples are given. These definitions about trust direction and symmetry are new elements of the unified taxonomy framework of trust. It is believed that these definitions can cover most situations about trust direction and symmetry in the real world and they can be used to analyze and model the properties of direction and symmetry of trust in distributed information systems.

Definition are given of one-way trust relationship, two-way trust relationship, and reflexive trust relationship for the properties of trust direction. For the properties of symmetry of trust relationships, definitions are provided of symmetric trust relationships, symmetric two-way trust relationship, and the whole set of trust relationships.

The details of these definitions are as follows:

**Definition 3** One-way trust relationship is the trust relationship with a unique trust direction from the trustors to trustees.

One-way is the default feature of a trust relationship if there is no further description.

Two-way trust relationship can be defined and used in information systems such as Microsoft’s domain trust. Actually, two-way trust relationship is the
result of binding two one-way trust relationships together. Many business transactions require two-way trust relationships. For example, when a customer does online shopping, the online shop needs to trust the customer under the condition that the customer provides suitable credit card information for the customer to pay appropriately for the shopping items. Meanwhile, the customer needs to trust the online shop under the condition of receiving a commitment from the online shop to guarantee that the shop will deliver the shopping items to the customer properly. These two one-way trust relationships are pre-conditions of the target transaction between them. These two one-way trust relationships should be bound together and be viewed as a two-way trust relationship.

A two-way trust relationship is defined as follows:

**Definition 4** Two-way trust relationship $TT'$ is the binding of two one-way trust relationships $T = \langle R, E, C, P \rangle$ and $T' = \langle R', E', C', P \rangle$, with $R' = E$ and $E' = R$. $T$ and $T'$ are the reflective trust relationships with each other in the two-way trust relationship.

In this definition, “binding” is the key word. If there are two one-way trust relationships between $R$ and $E$ but they are not bound with each other, then they are only two one-way trust relationships and there is no two-way trust relationship. When two one-way trust relationships are bound together, then there is a two-way trust relationship and these two one-way trust relationships can be called reflective trust relationships with each other. If the trustors and
the trustees are the same, then the trust relationship is reflexive. The reflexive trust relationship is defined as follows:

**Definition 5** Trust relationships \( T = < R, E, C, P > \) is a reflexive trust relationship when \( R = E \).

The symmetry of two trust relationships could be an important concern in the analysis or modelling of trust relationships in distributed information systems. The symmetry of two trust relationships is defined as the follows:

**Definition 6** If there is trust relationship \( T' = < R', E, C', P' > \), which is the result of swapping trustors and trustees in another trust relationship \( T = < R, E, C, P > \) (the swapping includes all possible ownerships in condition set and property set), then there is symmetry between \( T \) and \( T' \); \( T \) and \( T' \) are symmetric trust relationships with each other.

In the above definition, the swapping of trustors and trustees includes all possible ownerships in the condition set and the property set. The two trust relationships have the same condition set and property set except the possible ownerships in them. The symmetric/asymmetric two-way trust relationship is defined as follows:

**Definition 7** A two-way trust relationship \( TT' \) is a symmetric two-way trust relationship if there is symmetry between \( T \) and \( T' \); otherwise \( TT' \) is an asymmetric two-way trust relationship.
To discuss the symmetry of all trust relationships between a trustor set and a trustee set, the following is a definition:

**Definition 8** $WTR(R, E)$ is the whole set of trust relationships with the same trustor set $R$ and trustee set $E$.

**Definition 9** If every trust relationship in $WTR(R, E)$ has a symmetric trust relationship in $WTR(E, R)$, and every trust relationship in $WTR(E, R)$ has a symmetric trust relationship in $WTR(R, E)$, the trust between $R$ and $E$ are symmetric.

**Scenario Example II**: Here Microsoft’s domain trust is used as a regressive scenario example to discuss the properties of trust direction and trust symmetry defined in this section. Domain trust allows users to authenticate to resources in another domain. Also, an administrator is able to administer user rights for users in the other domain. The general definitions for the properties of direction and symmetry of trust relationships have general expressive power and can cover a broad range of commonly used notations. The related concepts in domain trust can be viewed as specific cases of these general definitions. In the following, the terms defined in this paper are used to review some concepts in domain trust.

- Based on Definition 1 in Section 3.1, the domain trust can be expressed as “entities in domain A trust entities in domain B without any condition that entities in domain B have the right to access the set of resources in domain A”.

• Microsoft’s domain trust includes both one-way trust and two-way trust. In Microsoft’s domain trust, one-way trust is defined as a unidirectional authentication path created between two domains. This means that in a one-way trust between domain A and domain B, users in domain A can access resources in domain B. However, users in domain B cannot access resources in domain A. Microsoft’s one-way trust is an example of one-way trust relationship in Definition 3. In a two-way domain trust, authentication requests can be passed between the two domains in both directions. Two-way trust is an example of two-way trust relationship in Definition 4.

• The entities in same domain trust each other without any condition that entities have the right to access the set of resources in the same domain. This is an example of reflexive trust relationship in Definition 5.

• There is symmetry in the two-way domain trust. The two one-way trust relationships bound in the two-way trust relationship are “entities in domain A trust entities in domain B without any condition that entities in domain B have the right to access the set of resources in domain A” and “entities in domain B trust entities in domain A without any condition that entities in domain A have the right to access the set of resources in domain B”. These two one-way trust relationships are symmetric trust relationships with each other in Definition 6. Microsoft’s two-way trust is a symmetric two-way trust relationship in Definition 7. Most of two-way trust rela-
tionships are asymmetric two-way trust relationships because trustors and trustees normally have different trust properties and trust conditions.

- In domain trust, the $WTR(A,B)$ based on Definition 8 has only one trust relationship from trustor domain $A$ to trustee domain $B$. For two-way domain trust, the trust between domain $A$ and domain $B$ is symmetric based on Definition 9.

The above scenario is used to illustrate the definitions about trust direction and symmetry. Actually, Microsoft’s domain trust includes some other features and most of these other features are based on the hierarchy of domains. The current considerations about trust direction and trust symmetry only focus on the simple situations with a flat structure for all the four-tuples of trust relationships (the concept of hierarchy is not involved). Beyond the Microsoft’s domain trust, definitions about trust direction and symmetry in this section are general concepts of trust and they can be used in applications and networks.

The properties of direction and symmetry of trust relationships play an important role in the modelling of the trust relationships in collaborative interactions in distributed environments. These definitions are new elements of the taxonomy framework of trust. It is believed that they can cover most situations related with direction and symmetry of trust relationship in the real world. In real implementations, these properties can be customized and configured based on the specific requirements.
3.5 Relations and Operations

The relations of trust relationships are considered in this section. Normally, there are multiple trust relationships in an information system and there are relations among these trust relationships. In particular, before a new trust relationship is introduced into the information system, the relations between the candidate and existing trust relationships must be carefully considered. These relations are important in the analysis and modelling of trust relationships in information systems. The understanding of business logics may be crucial in the analysis of these relationships. Specific business logics is beyond the scope of this section. The discussion about relations and operations of trust relationships is only at the expressive level and addresses the mathematical properties of four-tuple structure of trust relationships.

The discussion about relations of trust relationship will be based on the formal definition of trust relationship provided in last section. The trust relationship has a full syntax with trustor set, trustee set, condition set, and property set, and the full syntax must be strictly followed when trust relationships are expressed. A series of definitions, propositions, and operations for the relations of trust relationships will be provided. All definitions about these relations are elements of the unified taxonomy framework of trust. These relations of trust relationships will be used as tools in the analysis and design of trust relationships in real information systems.
According to the four-tuple mathematical structure of trust relationships, new trust relationships can be derived based on some existing trust relationships. A set of operations is defined to generate new trust relationships from existing trust relationships. These operations include operations using two existing trust relationships to generate a new trust relationship under specific constraints and operations of decomposing one existing trust relationship into two new trust relationships under specific constraints. Definitions are provided of equivalent, primitive, derived, direct redundant, and alternate trust relationships. Direct redundant trust relationships are classified into several subtypes. At the end of this section, two scenario examples are provided to show the usage and power of these operations and definitions in real information systems.

**OPERATION 1** Let $T_1 = (R_1, E_1, C_1, P_1)$ and $T_2 = (R_2, E_2, C_2, P_2)$. There is a set $T = (R_1 \cap R_2, E_1 \cap E_2, C_1 \cup C_2, P_1 \cup P_2)$. If $R_1 \cap R_2 = \emptyset$ or $E_1 \cap E_2 = \emptyset$, $T = \emptyset$.

If $R_1 = R_2$ and $E_1 = E_2$, the operation becomes:

**OPERATION 1A** Let $T_1 = (R, E, C_1, P_1)$ and $T_2 = (R, E, C_2, P_2)$. There is a set $T = (R, E, C_1 \cup C_2, P_1 \cup P_2)$.

If $R_1 = R_2$, $E_1 = E_2$ and $C_1 = C_2$, the operation becomes:
OPERATION 1B Let \( T_1 = (R, E, C, P_1) \) and \( T_2 = (R, E, C, P_2) \). Then there is a set \( T = (R, E, C, P_1 \cup P_2) \).

OPERATION 2 Let \( T_1 = (R_1, E_1, C, P) \) and \( T_2 = (R_2, E_2, C, P) \). There is a set \( T = (R_1 \cup R_2, E_1 \cap E_2, C, P) \).

If \( E_1 = E_2 \), the operation becomes:

OPERATION 2A Let \( T_1 = (R_1, E, C, P) \) and \( T_2 = (R_2, E, C, P) \). There is a set \( T = (R_1 \cup R_2, E, C, P) \).

OPERATION 3 Let \( T_1 = (R_1, E_1, C, P) \) and \( T_2 = (R_2, E_2, C, P) \). There is a set \( T = (R_1 \cap R_2, E_1 \cup E_2, C, P) \). If \( R_1 \cap R_2 = \emptyset \), \( T = \emptyset \).

If \( R_1 = R_2 \), the operation becomes:

OPERATION 3A Let \( T_1 = (R, E_1, C, P) \) and \( T_2 = (R, E_2, C, P) \). There is a set \( T = (R, E_1 \cup E_2, C, P) \).

OPERATION 4 Let \( T = \langle R, E, C, P \rangle \). If there are \( R_1, R_2 \) and \( R = R_1 \cup R_2 \), then there are trust relationships \( T_1 = \langle R_1, E, C, P \rangle \) and \( T_2 = \langle R_2, E, C, P \rangle \).

OPERATION 5 Let \( T = \langle R, E, C, P \rangle \). If there are \( E_1, E_2 \) and \( E = E_1 \cup E_2 \), then there are trust relationships \( T_1 = \langle R, E_1, C, P \rangle \) and \( T_2 = \langle R, E_2, C, P \rangle \).
§3.5 Relations and Operations 89

OPERATION 6 Let $T =< R, E, C, P >$. If there are $P_1$, $P_2$ and $P = P_1 \cup P_2$, then there are trust relationships $T_1 =< R, E, C, P_1 >$ and $T_2 =< R, E, C, P_2 >$.

This operation has the following special case:

OPERATION 6A Let $T =< R, E, C, P >$ and there are $P_1$, $P_2$, $C_1$, $C_2$ and $P = P_1 \cup P_2$, $C = C_1 \cup C_2$. If $C_1$ is the condition set for $P_1$ and $C_2$ is the condition set for $P_2$, then there are trust relationships $T_1 =< R, E, C_1, P_1 >$ and $T_2 =< R, E, C_2, P_2 >$.

All operations can be used to generate new trust relationships from the existing trust relationships under some specific constraints. The Operation 1 deals with any two trust relationships and a new trust relationship is generated, if the result is not $\emptyset$. The Operation 1A, 1B, 2A, 3A deal with how to use two trust relationships to generate a new trust relationship under some specific constraints. The Operation 4, 5, 6 and 6A deal with how to decompose one trust relationship into two trust relationships under some specific constraints. Operation 1A and Operation 6A are inverse operations. Operation 1B and Operation 6 are inverse operations. Operation 2A and Operation 4 are inverse operations. Operation 3A and Operation 5 are inverse operations.

In the following part of this section, the equivalent, primitive, derived, direct
redundant, and alternate trust relationships are discussed and defined. The direct redundant trust relationships will be classified into different types as well.

**Definition 10** Let $T_1 = (R_1, E_1, C_1, P_1)$ and $T_2 = (R_2, E_2, C_2, P_2)$. If and only if $R_1 = R_2$ and $E_1 = E_2$ and $C_1 = C_2$ and $P_1 = P_2$, then $T_1$ and $T_2$ are equivalent, in symbols:

$$T_1 = T_2 \iff R_1 = R_2 \text{ and } E_1 = E_2 \text{ and } C_1 = C_2 \text{ and } P_1 = P_2$$

**Definition 11** If a trust relationship cannot be derived from other existing trust relationships, the trust relationship is a primitive trust relationship.

**Definition 12** If a trust relationship can be derived from other existing trust relationships, the trust relationship is a derived trust relationship.

Note: Trust relationships are predefined in information systems. A derived trust relationship is always related to one or more other trust relationships. For an independent trust relationship, it is meaningless to judge it as a derived trust relationship or not.

**Proposition 1** If a derived trust relationship exists, there is information redundancy.

**Proof.** When the derived trust relationship is moved out of the system, the information of the derived trust relationship has not been lost. The derived trust relationship can be built when it is required. Hence, from the viewpoint of information, there is redundancy.
Definition 13 Let $T = \langle R, E, C, P \rangle$. If there is trust relationship $T' = \langle R', E', C', P' \rangle$ and $T \neq T'$, $R \subseteq R'$, $E \subseteq E'$, $C \supseteq C'$, $P \subseteq P'$. $T$ is a direct redundant trust relationship.

In the following part of this section, several special cases of direct redundant trust relationships based on one special tuple of a trust relationship are discussed. It is believed that these special cases play important roles in the analysis and design of trust relationships in information systems.

**Direct Redundancy Type 1 :** DLR-Redundant Trust Relationship

Let $T = \langle R, E, C, P \rangle$. If and only if there is a trust relationship $T' = \langle R', E, C, P' \rangle$ and $R' \supset R$, $T$ is a DLR-Redundant trust relationship.

Trust relationship $T$ is DLR-redundant trust relationship means that there is another trust relationship with super set of trustors and all other tuples are same as peers in $T$.

**Direct Redundancy Type 2 :** DLE-Redundant Trust Relationship

Let $T = \langle R, E, C, P \rangle$. If and only if there is a trust relationship $T' = \langle R, E', C, P' \rangle$ and $E' \supset E$, $T$ is a DLE-Redundant trust relationship.

Trust relationship $T$ is DLE-redundant trust relationship means that there is another trust relationship with super set of trustees and all other tuples are same as peers in $T$. 
Direct Redundancy Type 3: **DMC-Redundant Trust Relationship**

Let $T = \langle R, E, C, P \rangle$. If and only if there is an alternate trust relationship $T' = \langle R, E, C', P \rangle$ and $C' \subseteq C$, $T$ is a DMC-Redundant trust relationship.

Trust relationship $T$ is DMC-redundant trust relationship means that there is another trust relationship with subset of conditions and all other tuples are same as peers in $T$.

Direct Redundancy Type 4: **DLP-Redundant Trust Relationship**

Let $T = \langle R, E, C, P \rangle$. If and only if there is a trust relationship $T' = \langle R, E, C, P' \rangle$ and $P' \supseteq P$, $T$ is a DLP-Redundant Trust Relationship.

Trust relationship $T$ is DLP-redundant trust relationship means that there is another trust relationship with super set of properties and all other tuples are same as peers in $T$.

**Definition 14** Let $T = \langle R, E, C, P \rangle$, $T' = \langle R, E, C', P \rangle$ and $C \neq C'$. $T$ and $T'$ are alternate trust relationships of each other.

An alternate trust relationship means that there is an alternate condition set for the same trustor set, trustee set and property set. Perhaps, there are multiple alternate trust relationships. In distributed computing, multiple mechanisms and multiple choices are necessary in many situations and it is the main reason why we define and discuss alternate trust relationships here.

**Proposition 2** If $T$ is a DMC-Redundant trust relationship, there is one or
more than one alternate trust relationships which are not DMC-Redundant trust relationship.

**Proof.** If $T$ is a DMC-Redundant trust relationship, there is $T' = < R, E, C', P >$ and $C' \subset C$. $T'$ is an alternate trust relationship of $T$. If $T'$ is not DMC-redundant trust relationship, the proposition is proved. If $T'$ is a DMC-redundant trust relationship, the next $T''$ can be found, $T'' = < R, E, C'', P >$ with $C'' \subset C'$. Such a process will continue until the set of conditions includes minimum number of conditions. In every turn of the process, one or more conditions are removed from the condition set. Because $C$ contains limited conditions, the process can finish when no condition can be removed from the condition set. The final set of conditions is $C_f$. $T_f = < R, E, C_f, P >$ is an alternate trust relationship with non-redundant conditions.

A DMC-Redundant trust relationship may have multiple alternate trust relationships with different sets of non-redundant conditions.

In the following, two scenario examples are provided to show how the operations and relative types work in the real world.

**Scenario Example III:** Consider an online e-commerce service called FlightServ, which can provide flight booking and travel deals. FlightServ is designed using web services. FlightServ connects with customers, airlines, hotels and credit card services (some of these may also be web services). The whole system could
be very complicated, but in this example, only some basic trust relationships in the system are considered. In the system, customers are classified into normal flyers and frequent flyers. Originally, some trust relationships are modelled as follows:

- **TS3-1**: Airlines trust normal flyers can make their airline bookings, if they have address details and confirmed credit card information.

- **TS3-2**: Airlines trust frequent flyers with no condition that frequent flyers can make their airline bookings.

- **TS3-3**: Hotels trust normal flyers can make their hotels booking, if they have address details and confirmed credit card information.

- **TS3-4**: Hotels trust frequent flyers can make their hotels booking, if they have address details and confirmed credit card information.

- **TS3-5**: Credit card services are trusted by all possible entities without any condition that the credit card services will give the correct evaluation of credit card information.

- **TS3-6**: Credit card services are trusted by all possible entities without any condition that the credit card services will keep the privacy of credit card information.

For the above trust relationships in the system, based on definitions and operations in this section, the following analysis is derived:
• All the above trust relationships are primitive.

• Using the Operation 3A, trust relationships TS3-3 and TS3-4 can be merged to a new trust relationship TS3-(3)(4): “Hotels trust customers if they have address details and confirmed credit card information that customers can make their hotels booking”. If TS3-(3)(4) has been defined in the system, TS3-3 and TS3-4 becomes DLE-Redundant trust relationships and will be removed out of the system.

• Using the Operation 1B, trust relationships TS3-5 and TS3-6 can be merged to a new trust relationship TS3-(5)(6): ”Credit card services are trusted by all possible entities without any condition that the credit card services will give the correct evaluation of credit card information and the credit card services will keep the privacy of credit card information”. If TS3-(5)(6) has been defined in the system, TS3-5 and TS3-6 becomes DLP-Redundant trust relationships and will be removed out of the system.

Scenario Example IV: When people want to change their names, they need to apply to a specific organization (for instance, in Australia, the organization is the Registry of Birth Deaths & Marriages). The officers in the organization and the requesters are involved in this scenario. Using the full syntax of our definition of trust relationship, some trust relationships may be modelled as follows:

• TS4-1: Officers trust requesters if requesters have their Birth Certificate and Driver’s Licence that requesters have the right for the change.
- **TS4-2**: Officers trust requesters if requesters have their Citizenship Certificate and Driver’s Licence that requesters have the right for the change.

- **TS4-3**: Officers trust requesters if requesters have their Birth Certificate and Citizenship Certificate and Driver’s Licence that requesters have the right for the change.

If **TS4-1**, **TS4-2** and **TS4-3** are all the trust relationships in this information system, based on the definitions and operations in this section, we can have the following analysis:

- **TS4-1** and **TS4-2** are primitive trust relationships.

- **TS4-1** and **TS4-2** are alternate trust relationships of each other.

- **TS4-3** is a derived trust relationship which can be derived by **Operation 1A** with **TS4-1** and **TS4-2**.

- **TS4-3** is a DMC-Redundant trust relationship and it should be removed out of the system.

These scenario examples show how these operations and definitions provide terminologies and how they are used as tools in the analysis of trust relationships in information systems. In the analysis of these two scenarios, only some operations and definitions about relations of trust relationships are employed. It is hoped that these examples can provide a general picture for the usage of these operations and definitions.
3.6 Summary

In this chapter, a formal definition has been provided of trust relationship with a strict mathematical structure that can reflect many of the commonly used notions of trust. Based on this formal definition, a unified taxonomy framework of trust is provided. Under the taxonomy framework, different classifications of trust are discussed. In particular, the base level authentication trust at the lower layer and a hierarchy of trust relationships at a higher level is addressed. The trust scope label in order to describe the scope and diversity of trust relationship is defined. The properties of trust direction and trust symmetry are defined and discussed. A set of definitions, propositions, and operations based on the relations of trust relationships is provided. All the definitions about the properties of trust become elements of the unified taxonomy framework of trust. The taxonomy framework of trust provides accurate terms and useful tools for enabling the analysis, design, and implementation of trust. The taxonomy framework of trust is the first part.
of the research in the development of the overall methodology of analysis and
modelling of trust relationships and trust management solution in distributed
information systems.
In distributed information systems, the analysis and modelling of trust relationships could be a complex challenging task. Currently, it is lacking methodology and tools to facilitate the analysis and modelling of trust relationships. In this chapter, a general methodology is provided for analysis and modelling of trust relationships in distributed information systems, based on the formal model of trust relationship and taxonomy framework of trust described in last chapter.

The structure of this chapter is as follows. Section 4.1 describes the life cycle of trust relationships and provides general guidelines for the analysis and modelling of trust relationships. Section 4.2 provides the definition of an instance of a trust relationship. Section 4.3 discusses the analysis and modelling of individual trust relationships. Section 4.4 discusses the analysis and modelling the whole set of trust relationships. Section 4.5 discusses the linkage between analysis/modelling of trust relationships and trust management. Section 4.6 provides a summary for this chapter.
4.1 Analysis and Modelling Guidelines

The analysis and modelling of trust relationships are sub tasks in the system development life cycle (SDLC). SDLC includes typical phases as system requirements, system analysis, system design, system implementation, and system maintenance. Trust requirements and trust assumptions belong to the system requirements of an information system. Correct and thorough requirement specifications are essential in system development. Based on the trust requirements, the analysis and modelling of trust relationships form part of the system analysis and design in SDLC. A set of trust relationships should have been defined after the phases of system analysis and system design. In the implementation phase of SDLC, the set of trust relationships must be implemented according to the implementation circumstances of their related computing components. The set of trust relationships should be maintained according to business changes and system modifications.

The target of the methodology for analysis and modelling of trust relationships is to define and maintain a set of trust relationships with high quality. Analysis and modelling of trust relationships requires identification of subtle trust assumptions. The analysis and modelling of trust relationship must be put in the background of whole SDLC. With the life cycle of the development of a distributed information system, the analysis, modelling, implementing, and maintaining of trust relationships is an incremental, iterative process. A trust
relationship has its life cycle in the SLDC. The life cycles of trust relationships share the following basic steps, although the details and the orders of the steps may be changed in some cases.

1. Extract trust requirements of the system.

2. Identify possible trust relationships from trust requirements.

3. Choose the whole set of trust relationships from potential trust relationships.

4. Refine the whole set of trust relationships.

5. Implement the whole set of trust relationships.

6. Maintain trust relationships in system.

The research results about trust relationships in the last chapter provide the basis of the methodology. The formal model of trust relationship and the taxonomy framework of trust provide basic terms, scenario examples, and enabling tools in extracting trust requirements, analysis/modelling trust relationships, implementing trust relationships, and maintaining trust relationships.

The following provides some high level guidelines for the modelling and design of quality trust relationships in distributed information systems.

- **Completeness**: The completeness means that the whole set of trust relationships must cover all trust requirements for possible relationships in an information system.
Methodology for Analysis & Modelling of Trust Relationships

- **Efficiency**: The trust relationships defined in an information system should provide suitable level of abstraction to permit their efficient usage. A trust relationship had better have more coverage and then this can lead to the total number of trust relationships in the system being reduced.

- **High Cohesion**: A trust relationship should have a narrow scope. It had better be meaningful and understood easily. A trust relationship itself has some inherent maintainability. Efficiency and high cohesion are contradictory requirements. Trust relationships are always modelled after the trade-off between them.

- **Primitiveness**: In an ideal condition, all the trust relationships defined and used in an information system should be primitive trust relationships (see Section 3.5). That is, there is no information redundancy.

The analysis and modelling of the whole set of trust relationships in a distributed information system is one part of the analysis and design of the whole system and they are dependent on, or strongly coupled with, the design of other parts of the system. The initial set of trust relationships will be smoothed and refined in multiple life cycles in terms of completeness, efficiency, high cohesion and primitiveness.
4.2 Instances of Trust Relationships

Section 3.1 provided the formal definition of trust relationship with four tuples as trustor set, trustee set, condition set, and property set. These sets are defined from the viewpoint of information management and for the convenience of system analysis and design. At runtime, trust is always evaluated and established based on one trustor, one trustee, a set of conditions, and a set of properties. The instance of a trust relationship is defined as follows:

Definition 15 When trust is evaluated based on trust relationship $T = \langle R, E, C, P \rangle$ at runtime, only one trustor $r$, one trustee $e$ and requested properties $p$ are involved. There are $r \in R$, $e \in E$, $c \equiv C$, $p \subseteq P$. The $t = \langle r, e, c, p \rangle$ is called an instance of trust relationship $T$.

In Section 3.2, there is a description of the two layers of trust. For an instance of a trust relationship, the authentication trust on layer one is always evaluated at first. Then the conditions of the trust relationship are evaluated.

The instances of a trust relationship are normally used in an information system after the trust relationship has been modelled. However, instances of a trust relationship have deep roots in real business processes and they exist before the trust relationship has been modelled. Actually, the initial use cases of trust requirements provide the original states of instances of trust relationships. These use cases provide first-hand information for the analysis and modelling of trust relationships. Trust relationships are identified based on the abstraction of these
use cases.

The instances of trust relationships express runtime situations more directly than trust relationships. It is a basic design requirement that any instance of trust relationship must be assessable at runtime. This is a compulsory requirement for any trust relationship.

4.3 Individual Trust Relationships

The formal model of a trust relationship and the properties of individual trust relationships, described in the last chapter, are employed as the theoretical base and enabling tools for the analysis and modelling of individual trust relationships, discussed in this section. During the process of the analysis and modelling of an individual trust relationship, multiple aspects of the trust relationship should be examined and judged carefully. In the following, some of these major aspects are discussed:

- The four tuple syntax of trust relationships must be strictly followed. All the four tuples must be explicitly expressed in order to avoid any possible confusion, particularly due to one or more of the tuples being any one, or empty. An individual trust relationship must be well-defined. Any implication or indication that can be inferred from the context is not acceptable. Section 3.1 provides the theoretical base for this aspect.

- The type of the trust relationship should be identified based on the trust
classification described in Section 3.2. Normally, trust relationships under the same type share some common characteristics. For example, in Section 7.1, the concerns about the analysis and modelling of trust relationships for the access control of web services are based on the characteristics of web services and the type of resources access trust.

- The trust scope labels and comparing rules can be used as tools in the judgement of the scope and diversity of a trust relationship. A trust scope label expresses the target properties of trust and associated conditions for the specified context of the trust relationship. Trust scope labels could be treated as an independent subspace of trust relationships in the analysis and design. With the help of trust scope labels, it is possible to reduce the complexity of analysis and modelling of complicated individual trust relationships. Section 3.3 provides more details on this aspect.

- For trustor set, there are similar concerns when a role is defined in the role-based access control model. Trustor sets can be predefined based on associated business requirements. Then, individual trustors are assigned to trustor sets according to their status in the information system. All the above arguments for trustor sets are also suitable for trustee sets. The trustor set, trustee set, condition set, and property set, are combined together to define a trust relationship. The combination of trustor sets, trustee sets, condition set, and property set, could be more stable than
the particular combination of individual trustors, individual trustees, a set of conditions, and a set of properties.

- Condition set includes constraints for the trust relationship. The modelling of these constraints is based on the analysis of potential risks and threats in the information system. From the view point of execution, condition set provides filter functions to accept evidences and avoid risks by dynamically evaluating the trust relationship. Condition set has multiple interfaces to other computing functions/utilities/systems for capturing required information. Multiple evaluating mechanisms such as certificates, reputation, stored data, and environment status, are covered by the condition set. In next chapter, a trust management architecture is proposed with computing components to support checking of the constrains (referred to as trust evaluation in next chapter). The knowledge about the available constraint evaluating mechanisms in the information system can provide another perspective for the modelling of condition set.

- The definitions in Section 3.4 provide terms and scenario examples in the analysis and modelling of the properties of trust direction and trust symmetry. For an individual trust relationship, it will be judged based on the terms of one-way trust relationship, two-way trust relationship, reflexive trust relationship, and asymmetric/symmetric two-way trust relationship.
The above concerns about analysis and modelling of individual trust relationships are not exhaustive. It is hoped that the above list can provide some helpful judgement guidelines for the analysis and modelling of individual trust relationships in information systems.

4.4 Whole Set of Trust Relationships

The analysis and modelling of individual trust relationships discussed in the last section only focus on the properties of an individual trust relationship. For the whole set of trust relationships, relations among individual trust relationships must be considered. The relations among trust relationships bring in additional tasks that are beyond the analysis and modelling of individual trust relationships. In the last chapter, a set of operations and a series of definitions for the relations of trust relationships have already been defined. These operations and definitions provide terminologies, scenario examples, and tools in the analysis and modelling of the whole set of trust relationships when relations of trust relationships are considered. Some properties about trust direction and trust symmetry target certain kinds of relations of trust relationships as well. These properties about trust direction and trust symmetry should also be considered. In the following, a list of major concerns are provided for the analysis and modelling of the whole set of trust relationships.

- It is important to consider whether a trust relationship is a primitive trust
relationship or a derived trust relationship. The set of operations, defined in last chapter, provide tools to generate new trust relationships that are helpful to undertake this judgement. It has been proved that a trust relationship is not a primitive trust relationship if a derived trust relationship can be found. On the contrary, it is difficult, or there is no way, to prove the primitiveness of a trust relationship. The judgement for a primitive trust relationship is normally based on the fact that we have not found a way to prove it is a derived trust relationship.

- If a derived trust relationship is found, it is necessary to determine whether it is a direct redundant trust relationship or not. Unless there is a special need (such as design choice for improving performance), all the direct redundant trust relationships should be removed out of the whole set of trust relationships. The definitions of different direct redundancy types can help the checking of direct redundant trust relationships in the whole set of trust relationships. If a derived trust relationship is not a direct redundant trust relationship, it may be necessary to restructure the whole set of trust relationships and the process may be complicated. In order to remove the information redundancy, multiple trust relationships need to be modified, removed, or added. In some cases, the information redundancy will not be removed out of the system because of design choices.

- Next it needs to be addressed as to whether there are some alternate trust
relationships or not? If there are alternate trust relationships, then it needs to be determined whether there are some DMC-Redundant trust relationships in them. Alternate trust relationships are defined to support multiple mechanisms and multiple choices. It is necessary to judge the group of alternate trust relationships together, rather than only judge them one by one.

- Normally it is a good practice to review the group of trust relationships with same trustor set and same trustee set together, rather than only consider them one by one. Sometimes it is necessary to consider the symmetric property between a trustor set and a trustee set. Definition 8 and Definition 9 in Section 3.4 provide the basis to address these kind of concerns.

- Completeness has been listed in high level guidelines for the modelling and design of quality trust relationships (see Section 4.1). Actually, completeness is a requirement for the whole set of trust relationships. The whole set of trust relationships targets the coverage of all trust requirements in relationships in an information system. The completeness is judged based on the collection of individual trust relationships without considering relations among them.

- The whole set of trust relationships is normally not static. Corresponding with the changes of business requirements, it may be necessary to introduce some new trust relationships and remove some existing trust relationships.
The change management must be considered for the applications that are dependent on these new trust relationships or deleted trust relationships. When the whole set of trust relationships is modelled, it is better to separate the frequently changed factors from relatively stable factors and put them in different trust relationships. The convenience of future maintenance of the whole set of trust relationships should be considered at the early stage of the analysis and modelling of trust relationships.

Based on thorough understanding of trust requirements and available trust mechanisms in the target information system, the above concerns can be used as auxiliary tools in the analysis and modelling of the whole set of trust relationships.

4.5 Trust Relationships in View of Trust Management

The main objective of the trust research is to create a set of powerful tools to enable trust management in distributed information systems. The analysis and modelling of trust relationships is only the first step in trust management, and it is not the end of the whole story. The whole set of trust relationships will be implemented in the information system.

How to integrate the trust relationships with computing components in an information system is related to the analysis and modelling of trust relationships. The next chapter proposes a unified framework for trust management to enable
the development of trust management systems. A trust management architecture is proposed that includes data storage mechanism for trust related data and component packages for trust locating, trust evaluating, and trust consuming. Trust management systems focus on the processing of trust relationships. The understanding of the proposed unified framework for trust management is definitely helpful for the analysis and modelling of trust relationships. However, the methodology for analysis and modelling of trust relationships discussed in this chapter is not dependent on the unified framework. The unified framework for trust management and its implementation details can provide another angle for the analysis and modelling of trust relationships in an information system. The analysis and modelling of trust relationships is a basic task, and we present the methodology for the analysis and modelling of trust relationships before the introduction of unified framework for trust management in next chapter.

4.6 Summary

This chapter provides a general methodology for analysis and modelling of trust relationships in information systems. The life cycle of trust relationships has been discussed and some high level guidelines for the analysis and modelling of trust relationships have been given. The definition of instance of a trust relationship has been provided and its role in the the analysis and modelling of trust relationships has been discussed. Some major concerns for the analysis
and modelling of individual trust relationships as well as the whole set of trust relationships have also been discussed. At last, trust relationships have been put in the view of trust management of information systems.
Chapter 5

Unified Framework for Trust Management

This chapter proposes a unified framework for trust management that can cover a range of trust mechanisms including reputation, credentials, local data, and environment parameters. This framework provides a high level architecture and guidelines for the development of trust management systems. It is a unified framework that deals with locating trust, evaluating trust, and consuming trust, in a consistent manner, and in which different trust mechanisms can be easily assembled together. It has the ability to embrace and leverage established standards related with trust and trust management. The development of trust management systems in real applications can be automated to a substantially high level based on the proposed framework.

A generic trust management system is devised that is referred to as TrustEngine. The TrustEngine follows the initial ideas of PolicyMaker to separate generic mechanisms of trust management from applications. TrustEngine has a generic set of functions, interfaces, and data storage for trust management tasks in distributed
information systems. TrustEngine has a modular-based open architecture. It is extensible and embraces new computing components.

This chapter is structured as follows. Section 5.1 provides an introduction for the necessity of a unified framework for trust management. Section 5.2 describes the trust management architecture. Section 5.3 provides more design details of generic system components in the proposed trust management architecture. Section 5.4 discusses the generic system setting up and operations of the target trust management system based on the proposed trust management architecture. Section 5.5 provides an abstract algorithm for trust management tasks. Section 5.6 discusses the characteristics and advantages of the proposed unified framework for trust management. Section 5.7 provides a summary for this chapter.

5.1 Introduction

There are many trust management systems that are exclusively based on credentials. In these systems, credentials are the only type of trust evidence accepted. Before the clear concept of trust management, earlier systems such as PKI and PGP have already used credentials to deal with trust management problems. PolicyMaker [14], KeyNote [20], and REFEREE [27] belong to these kinds of trust management systems. They have been reviewed in Chapter 2. Normally, these trust management systems include credential verification and security policies to restrict access to resources and services. G. Suryanarayana et al [120] pointed
out that these systems are limited in the sense that they do not enable an entity to aggregate the perception of other entities in the system in order to choose a suitable reputable service.

The reputation of an entity can be used as a criterion to determine the restriction of access to resources and services. Some information systems such as e-Bay employ reputation as the exclusive evidence for trust. Reputation-based systems such as XREP [33], NICE [71], P-Grid [2] provide the facilities to compute the reputation of an involved entity by aggregating the perception of other entities in the system. Some reputation systems like TrustNet [111] and NodeRanking [99] utilize existing social relationships to compute reputations based on various parameters. M. Kinateder et al [66] proposed a generic model for trust based on reputation. Normally, these reputation systems are limited in the sense that they do not link the purpose of reputation to its evaluation.

All existing trust management systems focus on building up a new trust management layer and the concept of trust is always assumed in a specific context. These systems normally support certain types of trust mechanisms exclusively. Most of them support credentials and only focus on credential processing. It is believed that it is necessary to have a unified framework for trust management with the ability to put different trust mechanisms under the same umbrella.

This chapter proposes a unified framework for trust management that can address the above mentioned limitations of current trust management systems. The unified framework uses a consistent way to cover a broad variety of trust
mechanisms including credentials, reputation, local data storage, and environment parameters. Different trust mechanisms can be assembled together easily when they are needed. The framework will embrace established standards and existing computing utilities/functions/systems in distributed information systems.

A trust management architecture is proposed and the generic computing components in the architecture are described that can be used as enabling tools for the development of sub systems (or a separated layer) of trust management in distributed information systems.

The unified framework for trust management is based on the formal model of trust relationship and unified taxonomy framework of trust proposed in Chapter 3. The formal model of trust relationship can cover multiple and/or complex trust mechanisms in distributed information systems. The taxonomy framework can reflect the different forms of trust relationships based on their specific characteristics, and a range of useful trust relationships can be expressed and compared. The general methodology has been developed for the analysis and modelling of trust relationships in Chapter 4. These research results form a solid basis of the unified framework for trust management.

For a real trust management system, trust relationships must have been modelled and loaded into the trust management system before these trust relationships are requested by related applications. When a trust relationship is defined, any condition in a condition set must be assessable, that means the condition can always be evaluated in the trust management system. The supporting trust mech-
anisms and condition constraints must be consistently considered in the analysis and modelling of trust relationships and the development of trust management systems.

The concerns in this chapter focus on the general characteristics of trust management systems. The computing components and processes in real systems will be abstracted to generic computing components and processes. This chapter devises TrustEngine as a generic trust management system to express the unified framework. The unified framework for trust management is expressed by TrustEngine with its architecture, generic system components, generic system setting up, and typical operation sequences.

5.2 Trust Management Architecture

Trust management architecture targets a standard, high level design for the development of trust management systems. Trust management architecture can be used as an auxiliary tool in the whole life cycle of the development of trust management systems, including specifications of requirements, preliminary design, active deployment, and maintenance. The architecture provides the basis for dependency and consistency analysis for trust management. As a general architecture, it can be reused in different systems.

The trust management architecture should have the ability to embrace frequently used mechanisms of evaluation of trust relationships and consumption of
Unified Framework for Trust Management

trust relationships. The architecture describes the high level design of the trust management system in terms of major computing components and their inter-relationships. The details of its generic computing components provide guidelines and constrains for its implementation.

TrustEngine holds a set of trust related computing components that could be separated from applications. These computing components are generic. TrustEngine expresses a separated layer of trust management in distributed information systems. The formal definition of a trust relationship provides the starting point for the trust management architecture. TrustEngine addresses applications’ trust requests like a database query engine. TrustEngine accepts a requested trust relationship, or a set of inputs, that could be used to determine the requested trust relationship. Depending on the form of the query, TrustEngine locates the requested trust relationship, evaluates the trust relationship, and manages the consumption of the evaluation result.

In TrustEngine, there is a data storage mechanism that is separated from other computing components. The finding of trust relationships based on the requests, the evaluation of trust relationships with the help of trust mechanisms, and the consuming management are separated and put into different computing packages. These computing packages have the flexibility to be extended for holding new trust components. Each component in TrustEngine performs some trust function or has some data storage to be used by other trust functions.

TrustEngine includes TrustDatabase for the storage of trust related data and
component packages as LocatingTrust, EvaluatingTrust and ConsumingTrust. Figure 5.1 shows the top level components of TrustEngine.

![Figure 5.1: TrustEngine Package Hierarchy](image)

5.2.1 TrustDatabase

TrustEngine includes a persistent storage mechanism for storing and retrieving information about trust. TrustDatabase is the data storage of TrustEngine that
maintains trust relationships, instances of trust relationships, and trust parameters. These trust related data will only be used by the computing components of TrustEngine. The storage mechanism can be a relational database or data profile.

For a real distributed information system, when an application is being developed, trust relationships required in the application must be analyzed and modelled. The trust relationships and trust related parameters must have been loaded into the TrustDatabase before they are involved at runtime. The storage and retrieving mechanisms for instances of trust relationships, and other runtime parameters, must have been set up as well.

5.2.2 TrustControl

TrustControl is the package for the overall management and control of TrustEngine at runtime. TrustEngine controller is the only computing component in this package. It is the general controller of TrustEngine that links applications and functional packages of TrustEngine (LocatingTrust, EvaluatingTrust and ConsumingTrust). See Figure 5.2.

5.2.3 LocatingTrust

LocatingTrust is the package for finding the trust relationship based on the request. There are three components in this package that are referred to as locating trust controller, trust relationship locator, and authentication controller. Locat-
Figure 5.2: Components of TrustControl

EvaluatingTrust contains computing components required for the evaluation of trust relationships. The evaluation of a trust relationship involves checking whether the conditions of a trust relationship can be satisfied or not. The conditions of trust relationships take into account the risks from the evil actions of trustees, evil actions from other parties, and from unstable environments.
Multiple trust mechanisms can be involved in the evaluation of a single trust relationship. The unified trust management framework provides an integration place for these trust mechanisms to cooperate with each other. The existing standards and systems can be employed to support required tasks in the evaluation processes. Any existing system or mechanism for checking or evaluating the evidence of trust could be included in the EvaluatingTrust package. For instance, the existing reputation-based systems and credential based systems can be employed to provide required information for trust evaluation.

In EvaluatingTrust, trust evaluation controller is the computing component that assigns the evaluation tasks to other functional components in this package. EvaluatingTrust has functional components for specific evaluating tasks, namely credential evaluation, reputation evaluation, stored data evaluation, and environment evaluation. In the implementation, the package of EvaluatingTrust will be
customized or extended based on the specific requirements. See Figure 5.4.

![Diagram](image)

**Figure 5.4**: Components of EvaluatingTrust

### 5.2.5 ConsumingTrust

ConsumingTrust contains the computing components for the control and management of trust consuming. Consuming trust deals with how to use the output of the evaluation of a trust relationship. ConsumingTrust contains the consuming controller and the two next level packages ApplicationConsuming and SystemConsuming. Consuming controller is the manager of trust consuming. It receives the result of trust evaluation and assigns consuming tasks to ApplicationConsuming and SystemConsuming. ApplicationConsuming deals with the consuming of
trust by applications. SystemConsuming deals with the consuming of trust by TrustEngine and auditing system. See Figure 5.5.

![Figure 5.5: Components of ConsumingTrust](image)

5.2.5.1 ApplicationConsuming

In application consuming, the evaluation of an instance of a trust relationship is not always to be consumed immediately. The result of the evaluation of the instance of the trust relationship can be stored and/or distributed in different ways. There are three ways to use the output of trust evaluation. The first way is that the result of trust evaluation is immediately used by requesting applications. The computing component for this way is direct trust consuming controller. The second way is to generate credentials with the result of trust evaluation as input. These credentials will be used in the future by the same or other applications. Credential generator consuming is the corresponding computing component. The third way is that the result of trust evaluation is stored
in the database and the data will be retrieved and used by applications in the future. Data Storage Consuming is the corresponding computing component.

The package of ApplicationConsuming has four computing components, namely application consuming controller, direct trust consuming controller, credential generator consuming, and data storage consuming. Application consuming controller plays the role of the manager for application consuming. Application consuming controller receives tasks from consuming controller and it assigns tasks to direct trust consuming controller, credential generator consuming, and data storage consuming. See Figure 5.6.

![Figure 5.6: Components of ApplicationConsuming](image)

### 5.2.5.2 SystemConsuming

The package of SystemConsuming has three components that are system consuming controller, TrustEngine consuming controller, and auditing consuming controller. System consuming controller plays the role of the manager for system consuming. System consuming controller receives tasks from the consuming
controller and it assigns tasks to the TrustEngine consuming controller and auditing consuming controller. TrustEngine consuming controller deals with the consuming of trust by TrustEngine. Auditing consuming controller deals with the consuming of trust by auditing system. See Figure 5.7.

![Figure 5.7: Components of SystemConsuming](image)

5.3 System Components of TrustEngine

This section provides further details on the system components of the TrustEngine. The description of these system components will focus on their generic functions, interfaces, and inter-relationships with other system components. The existing computing standards, utilities, and systems are viewed as generic building blocks in these generic components.

These generic components cover the majority of required trust functions that
can be separated from applications in a broad range of distributed information systems in the real world. These generic system components should have a high degree of comprehensibility and flexibility. They can provide guidelines or further information for the development of individual computing components of trust management systems in the real world.

The real trust management system must be developed based on specific business requirements, available technologies, and computing environments. The above generic description of system components provides a high level design for these system components. With the help of these generic components, the development of real system components in a trust management system becomes the implementation of business requirements with the considerations of available technologies and computing environments. These generic system components can bring benefits to reduce the cost and time of the system development.

The following lists the generic system components and provides a high level description for each of them.

**TrustEngine Controller:** TrustEngine controller is a runtime controller of TrustEngine. It has an interface to receive trust requests from applications and an interface to return necessary feedback information from TrustEngine to request applications. It assigns tasks to locating trust controller, trust evaluating controller, and trust consuming controller. It receives returned information from locating trust controller, trust evaluating controller, and trust consuming con-
troller. It performs management tasks for TrustEngine among the computing packages LocatingTrust, EvaluatingTrust, and ConsumingTrust.

**Locating Trust Controller:** Locating trust controller has an interface to receive requests from TrustEngine controller. It forwards the request to trust relationship locator for finding the related trust relationship. The locating trust controller has an interface to return the status of locating of the required trust relationship to TrustEngine controller. The locating trust controller assigns authentication controller to perform the task of authentication for the involved trustee. It has an interface to return information of trust locating to TrustEngine controller.

**Trust Relationship Locator:** Trust relationship locator performs the function to find the related trust relationship. It has an interface in connection with TrustDatabase where trust relationships are maintained. It has another interface to return the searching result to locating trust controller.

**Authentication Controller:** Authentication controller performs the function to authenticate the trustee in a required trust relationship. It has an interface to receive authentication task from locating trust controller and it has an interface to employ existing functions/utilities/systems for the authentication. It has an interface to return authentication information (authentication tokens or status)
§5.3 System Components of TrustEngine

To locating trust controller.

**Trust Evaluation Controller**: Trust evaluation controller performs the function of managing trust evaluation. It has an interface to receive tasks of trust evaluation from TrustEngine controller. It assigns evaluation tasks to the computing components of credential evaluation, reputation evaluation, stored data evaluation, and environment evaluation. It has an interface to return evaluating results to TrustEngine controller.

**Credential Evaluation**: Credential evaluation is the computing component for credential evaluation. It includes multiple evaluating mechanisms for different credentials. It has computing functions and/or provides interfaces linking with existing computing utility of credential evaluation. It has an interface to receive tasks from trust evaluation controller and an interface to return the result of evaluation to trust evaluation controller.

**Reputation Evaluation**: Reputation evaluation looks after computing tasks in reputation evaluation. It includes computing functions for reputation calculation and/or interfaces to the existing utilities of reputation evaluation. It has an interface to receive the task from trust evaluation controller, and an interface to return the result of evaluation to trust evaluation controller.
Stored Data Evaluation: Stored data evaluation looks after the evaluation of trust against stored data. It has an interface to receive tasks from trust evaluation controller, and an interface to return the result of evaluation to trust evaluation controller.

Environment Evaluation: Environment evaluation looks after the evaluation of trust against environment variables. It has an interface to receive tasks from trust evaluation controller, and an interface to return the result of evaluation to trust evaluation controller.

Trust Consuming Controller: Trust consuming controller performs the management of trust consuming of TrustEngine. It has an interface to receive the consuming tasks from TrustEngine controller, and an interface to return the consuming result/status to TrustEngine controller. It assigns consuming tasks to application consuming controller and system consuming controller.

Application Consuming Controller: Application consuming controller performs the management of consuming of trust by applications. It has an interface to receive tasks from trust consuming controller, and an interface to return the consuming result/status to trust consuming controller. It assigns tasks to direct trust consuming controller, credential generator consuming, and data storage consuming.
Direct Trust Consuming Controller: Direct trust consuming controller looks after the consuming of trust when the evaluation of trust relationship is consumed immediately by the request application. It has an interface to receive the consuming tasks from the application consuming controller, and an interface to return the consuming status to application consuming controller. It has an interface for the consuming of trust relationship by the request application.

Credential Generator Consuming: Credential generator consuming looks after the consuming of trust when the evaluation of trust relationship is consumed by the generation of credentials. It has an interface to receive the consuming tasks from the application consuming controller, and an interface to return consuming status to application consuming controller. It has functions to generate and manage credentials. Existing standards and computing utility can be employed in the generation and management of the credentials.

Data Storage Consuming: Data storage consuming looks after the consuming of trust when the evaluation of trust relationship is consumed by storing related information in the database. The information stored in the database will be retrieved by applications in the future. It has an interface to receive the consuming tasks from the application consuming controller, and an interface to return consuming status to application consuming controller. It has functions to format
data and has interfaces to save data with different data storage mechanisms such as local database, remote database, and profiles.

**System Consuming Controller**: System consuming controller performs the management of consuming of trust by system. It has an interface to receive tasks from trust consuming controller, and an interface to return the consuming result/status to trust consuming controller. It assigns tasks to the TrustEngine consuming controller and the auditing consuming controller.

**TrustEngine Consuming Controller**: TrustEngine consuming controller looks after the consuming of trust by TrustEngine itself. It has an interface to receive the consuming tasks from the TrustEngine consuming controller, and an interface to return consuming status to TrustEngine consuming controller. It has functions for trust consuming by TrustEngine, and interfaces to save data in TrustDatabase.

**Auditing Consuming Controller**: Auditing consuming controller manages the consuming of trust for the auditing purpose. It has an interface to receive the consuming tasks from the TrustEngine consuming controller, and an interface to return consuming status to TrustEngine consuming controller. It has interfaces to link to auditing functions or database in the system.
In the development of a trust management system, the system components described in the last section will be customized based on specific requirements of the target information system. The implementation result of TrustEngine normally runs as a relatively independent system on a local server, or a logical local server, to serve one or multiple applications for their trust management tasks. It is also possible to embed the implementation result of TrustEngine into applications as a relatively independent software package. After the required computing components are installed, at runtime, a set of operations of these components will be activated based on the specific trust request from applications.

This section provides a generic description of system setting up and system operations. The description in this section provides high level guidelines for the system setting up and operations in the real world.

The system setting up of TrustEngine includes the setting up of its database and system components. The TrustEngine uses TrustDatabase for its data storage. The suitable type of database should be chosen and installed at first. The data storage mechanisms for trust relationships, instances of trust relationships, and trust parameters must be defined. In the case of a relational database, all data are stored in a set of tables. The TrustDatabase must be set up before TrustEngine can perform its functions of trust management.

The important system components have been described in the last section.
Their customization and setting up are based on the real situations of business and system requirements. In the implementation of the unified framework of trust in the real world, a customized list of the above computing components will be developed and installed. It is also possible to need more computing components that are beyond the above mentioned system components. These additional components will belong to component packages of LocatingTrust, EvaluatingTrust, or ConsumingTrust.

TrustEngine looks after all the trust management tasks that could be separated from applications. At runtime, when there is a trust request from an application, a set of system operations of TrustEngine will be activated. A typical sequence of involved system operations is as follows:

**TC1:** TrustEngine controller is the first computing component to be activated and it will assign a task to locating trust controller.

**LT1:** Locating trust controller assigns a task to trust relationship locator.

**LT2:** Trust relationship locator finds the required trust relationship and returns it to locating trust controller.

**LT3:** Locating trust controller requires authentication controller to perform the task of authentication for the involved trustee.

**LT4:** Authentication controller performs the task of authentication.

**LT5:** Locating trust controller returns information of locating of trust to TrustEngine.
TC2: TrustEngine controller requires trust evaluation controller to evaluate the trust relationship.

TE1: Trust evaluation controller assigns evaluation tasks to the computing components of credential evaluation, reputation evaluation, stored data evaluation and environment evaluation.

TE2: Credential evaluation checks credentials.

TE3: Reputation evaluation performs the computing tasks of reputation evaluating.

TE4: Stored data evaluation performs the evaluation of trust against stored data.

TE5: Environment evaluation performs the evaluation of trust against environment variables.

TE6: Trust evaluation controller integrates the results of TE2, TE3, TE4, and TE5 and returns final evaluating result to TrustEngine controller.

TC3: TrustEngine controller assigns trust consuming controller to manage the consuming of the evaluated trust relationship.

TU1: Trust consuming controller assigns consuming tasks to application consuming controller, and system consuming controller.
**TUA1:** Application consuming controller assigns tasks to direct trust consuming controller, credential generator consuming and data storage consuming.

**TUA2:** Direct Trust Consuming Controller informs the application of the initiator of the trust request for the consuming of trust.

**TUA3:** Credential generator consuming generates credentials based on the result of trust evaluation. The credentials will be stored or delivered based on the specific requirements of a real system.

**TUA4:** Data storage consuming formats the data and saves it with different data storage mechanisms such as local database, remote database, and profiles.

**TUS1:** System consuming controller assigns tasks to TrustEngine consuming controller and auditing consuming controller.

**TUS2:** TrustEngine consuming controller performs functions for trust consuming by TrustEngine and it saves data in TrustDatabase.

**TUS3:** Auditing consuming controller performs functions of trust consuming for the auditing purpose.

The above typical sequence of system operations provides the general description of the behaviors of TrustEngine at runtime. It shows the flow of logic within TrustEngine and the interactions between TrustEngine and applications. The relationships and the interactions between system components are emphasized.
5.5 Trust Management Algorithm

In this section, we provide an abstract algorithm for trust management tasks in distributed information systems. We assume that a sub trust management system should be developed based on our proposed trust management architecture that can separate trust management tasks from a distributed information system. We use an abstract example scenario where a client uses a service provided by an entity using a set of trust requirements as preconditions. The algorithm will provide a rough sense about how to use the computing components of our proposed architecture in the target trust management system.

We assume that there is a trust relationship $T = < R, E, C, P >$ where $R$ is the trustor set; $E$ is the trustee set; $C$ is the condition set; and $P$ is the property set. Trust relationship $T$ captures the trust requirements when clients using the service provided by the entity. The following is the whole set of high level pseudo-codes for the trust management algorithm:

```
TrustManagement

Receive trust request message $(c, e, s)$

Perform TrustEngineController

Return status message

End-TrustManagement

TrustEngineController
```
Perform LocatingTrustController

IF locating trust is successful

Perform TrustEvaluatingController

IF conditions in trust relationship $T$ are satisfied

Perform TrustConsumingController

ELSE

Prepare message for unsatisfied trust evaluation

ENDIF

ELSE

Prepare status message for unsuccessful locating trust

ENDIF

Return status message

End-TrustEngineController

LocatingTrustController

Perform TrustRelationshipLocator

IF a unique trust relationship is located

Perform AuthenticationController

ENDIF

Return status message

End-LocatingTrustController
Trust Relationship Locator

Query TrustDatabase with $c, e, s$ as criteria

IF a unique trust relationship is found

Return a message including $(c, e, T, E, C, P)$

/* $c$: truster; $e$: trustee. */

/* $T$: trusterset; $E$: trusteeset, $C$: conditionset; $P$: propertyset. */

ELSE

Return a message for status of querying result

ENDIF

End-TrustRelationshipLocator

AuthenticationController

Authenticate trustee $e$

/* Existing functions/utilities/systems for authentication */

/* can be employed here. */

Return the status of authentication

End-AuthenticationController

Trust Evaluating Controller

IF credential evaluation is required

Perform Credential Evaluation

ENDIF
IF reputation evaluation is required

    Perform ReputationEvaluation

ENDIF

IF stored data evaluation is required

    Perform StoredDataEvaluation

ENDIF

IF environment evaluation is required

    Perform EnvironmentEvaluation

ENDIF

Return the status of trust evaluation

End-TrustEvaluatingController

CredentialEvaluation

    Fetch the credential

    /* The credential can be provided by a message from the client */

    /* or the credential is stored at an available place. */

    Find the type and algorithm associated with the credential

    Get the public key for checking the signature on the credential

    Check the expired date of the credential

    Check related details of the credential

    /* The details of the credential (such as it is for trustee e) */

    Check the signature of the credential
/ * Using the public key and associated algorithm */

Return the status of credential checking

End-CredentialEvaluation

ReputationEvaluation

Calculate the reputation of trustee $e$

/ * Public opinions, normally community-based. */

/ * Existing reputation functions/utilities/systems may be employed. */

Check calculated reputation has achieved the required value or not

Return the status of reputation checking

End-ReputationEvaluation

StoredDataEvaluation

Retrieve related information from stored data

Check the retrieved information against required criteria

Return the status of stored data checking

End-StoredDataEvaluation

EnvironmentEvaluation

Collect related environment parameters

Check environment parameters against required criteria

Return the status of environment checking
Unified Framework for Trust Management

End-EnvironmentEvaluation

TrustConsumingController

IF application consuming is required

Perform ApplicationConsumingController

ENDIF

IF system consuming is required

Perform SystemConsumingController

ENDIF

End-TrustConsumingController

ApplicationConsumingController

IF direct trust consuming is required

Return message for immediately consuming by client c’s application

ENDIF

IF credential generator consuming is required

Perform CredentialGeneratorConsuming

ENDIF

IF data storage consuming is required

Perform DataStorageConsuming

ENDIF

Return the status of application consuming
5.5 Trust Management Algorithm

End-ApplicationConsumingController

CredentialGeneratorConsuming

  Prepare detailed data in target credential
  Make the credential
  Get the private key for signature
  Sign the credential with the private key
  Store or deliver the credential to its target
  Return the status of credential generator consuming

End-CredentialGeneratorConsuming

DataStorageConsuming

  Prepare detailed data to be stored in database
  Update database to store required information
  Return the status of data storage consuming

End-DataStorageConsuming

SystemConsumingController

  IF TrustEngine consuming is required
    Perform TrustEngineConsumingController
  ENDIF

  IF auditing consuming is required
Perform AuditingConsumingController

ENDIF

Return the status of system consuming

End-SystemConsumingController

TrustEngineConsumingController

Collect data for target TrustEngine consuming

Activate functions of TrustEngine consuming

Update the related data records in TrustDatabase

Return the status of TrustEngine consuming

End-TrustEngineConsumingController

AuditingConsumingController

Collect data for auditing purpose

Send required data to auditing services

Return the status of auditing consuming

End-AuditingConsumingController

The above algorithm provides a general programming structure for target trust management systems in distributed information systems. In particular, the algorithm can be implemented in a distributed way allowing a range of computing components for trust management tasks to run at different nodes on networks or
5.6 Discussion

The trust management approach follows the basic idea of M. Blaze et al [14] to build a separated trust management layer. TrustEngine has been devised to express this trust management layer. The unified framework for trust management is composed of the architecture of TrustEngine, description of generic computing components, and system setting up and operations. The proposed unified framework for trust management covers a range of mechanisms for trust evaluation and trust consumption in distributed information systems. The goal of our trust management framework is to provide a solid foundation upon which may evolve the system’s functions for trust management tasks.

There are multiple aspects for TrustEngine that are different from existing trust management systems:

- **Unified Framework of Trust Mechanisms:** Most of existing trust management systems only focus on one kind of trust mechanisms. For example, PolicyMaker, KeyNote, REFEREE, and IBM Trust Establishment employ credentials as the only kind of mechanism for trust management. TrustEngine supports multiple trust mechanisms. The major trust mechanisms include credentials, reputation, local data, and environment parameters. A complicated trust mechanism can be built based on these single
trust mechanisms. The unified framework for trust management allows different trust mechanisms including credentials, reputation, local data, and environment parameters to work together under the same umbrella. The unified framework for trust management provides a base to handle multiple trust mechanisms in a comprehensive and consistent manner. In particular, credentials and reputation can cooperate with each other under the same unified framework.

- **Multiple Ways of Trust Consumption:** Most of the existing trust management systems assume a straightforward way of trust consumption, where the application requesting the trust evaluation will consume the trust evaluation results directly. For example, PolicyMaker accepts input that is composed of a set of local policy statements, a collection of credentials, and a string describing a proposed action; PolicyMaker evaluates the request; then PolicyMaker returns a simple yes/no answer or additional restrictions to the application that initiates the original trust request. Different to existing trust management systems, the TrustEngine provides not only computing components that support multiple trust mechanisms, but also computing components that support multiple ways of trust consumption. The results of trust evaluation could be consumed by both the trust management system and applications. The application consuming includes direct trust consuming, credential generator consuming, and data storage...
• **Embrace of Established Standards:** The proposed unified framework for trust management has the ability to embrace established standards or existing systems for trust management tasks. All the existing standards or systems for trust management can be employed by TrustEngine to perform some specific tasks. For example, in reputation evaluation, the reputation could be calculated by existing utility in the system. TrustEngine supports the calling of existing utility and uses the result in its reputation evaluation. Established standards and existing systems can be easily integrated in TrustEngine.

• **Trust Relationships Oriented:** Most of existing trust management systems separate mechanisms from application-specific policies directly. Normally trust policies are defined in applications and policy compliance checkers are used to evaluate trust policies at runtime. The evaluation results are always returned to requested applications directly. It is always necessary to define a policy language to express and interpret trust policies. TrustEngine is based on trust relationships rather than trust policies. A trust relationship itself has a strict structure and can be accurately interpreted both in TrustEngine and requested applications. The formal definition of trust relationship, and the taxonomy framework of trust, have provided a solid base for the analysis and modelling of trust relationships.
TrustEngine is designed to support the processing of trust relationships. A trust management system as the implementation of TrustEngine must have the computing ability to support the whole set of trust relationships of the target information system. Any condition of any trust relationship can be evaluated. The analysis and modelling of trust relationships and the development of a trust management system are related to each other and should be dealt in a consistent manner.

- **High Extensibility:** TrustEngine is module-based. Its implementation is highly extensible. When an application needs more trust mechanisms that a TrustEngine cannot provide, the corresponding computing components for these trust mechanisms can be easily added into the target trust management system. When a new computing component is introduced in the trust management system, there is minimal or no affect on the other computing components.

### 5.7 Summary

This chapter proposes a unified framework for trust management. A generic trust management system, referred to as TrustEngine, is devised to express the framework. The unified framework for trust management is based on the research results of the formal model of trust relationship, the unified taxonomy framework of trust, and the methodology for analysis and modelling of trust relationships.
It targets the goal that the developers of trust management systems can have a solid high level architecture to evolve system functions for trust management tasks by simply implementing some business logic. The development of a trust management system in the real world can be automated to a high level based on the proposed framework.

The framework puts multiple trust mechanisms including credentials, reputation, stored data, and environment parameters under the same umbrella, so that they can cooperate with each other to satisfy some complex trust requirements. The framework supports multiple ways of trust consumption. The framework has the ability to embrace existing trust standards and computing functions/utilities/systems for trust management tasks.

The proposed TrustEngine provides a standard high level architecture for trust management systems, with a series of generic computing components for trust management tasks. In the trust management architecture, trust request, trust evaluation, and trust consuming are handled in a comprehensive and consistent manner. The details of generic computing components in the trust management architecture are provided. A generic description of system setting up and system operations of TrustEngine is also provided.

Finally, the characteristics of the proposed unified framework for trust management have been discussed. These characteristics include unified framework of trust mechanisms, multiple ways of trust consumption, embracement of established standards, trust relationships based, and high extensibility.
This page is intentionally left blank.
Chapter 6

Trust Management System for

Federated Medical Services

This chapter serves as an example of the implementation of the proposed TrustEngine architecture, and demonstrates how the devised trust management system can support an online booking system for federated medical services. The computing components of TrustEngine architecture are customized based on trust requirements in federated medical services. The customized set of computing components of TrustEngine are implemented by a series of web services to perform related trust management tasks. A SQL Server is used for the required data storage.

An online booking system for federated medical services is developed. We assume that an online booking requires a trust relationship between the involved patient and physician as a precondition. The online booking system employs the developed trust management system to perform trust management tasks.

Section 6.1 provides a general introduction of trust management in federated medical services and defines three trust relationships in the target trust...
management system. Section 6.2 provides a high level description of the trust management solution in federated medical services using the implementation of TrustEngine architecture described in last chapter. Section 6.3 describes the specifications of web services and three most important XML messages in the target trust management system. Section 6.4 describes an online booking system for federated medical services that employs the developed web services in the last section for carrying out the trust management tasks. Section 6.5 provides a summary for this chapter.

6.1 Trust Management in Federated Medical Services

The role of trust management in federated medical services has recently gained increasing attention [15, 52, 11]. The Internet makes it possible for federated medical services to be pervasive and ubiquitous. In federated medical services, there are multiple trust relationships between entities such as patients, physicians, hospitals, insurance companies, and pharmacies. In particular, trust relationships between patients and physicians, or medical service providers, are critical in the usage of federated medical services in providing patients services and protecting sensitive information. The need for effective trust management in federated medical services is self-evident. The trust management is also central for the ongoing debates about the structure and regulation of medical service delivery.
Federated medical services can have complicated context based trust requirements, and a range of applications need to make complex trust decisions that are dependent on runtime situations. These context-based trust requirements must be captured by the trust relationships defined in federated medical services that can express contextual constraints. These trust relationships are evaluated based on real situations, at runtime, and there may be multiple ways for the evaluation results to be consumed in federated medical services. Multiple trust mechanisms including credentials, reputation, data storage, and environment parameters could be involved in federated medical services, and these trust mechanisms need to be separated from the applications.

In federated medical services, there are many trust relationships and they could be very complicated. Some of these trust relationships are considered as the purpose of this chapter is to show an example of the implementation of TrustEngine and provide a demonstration for the consuming of the developed trust management system.

The formal model of trust relationship and the trust framework described in Chapter 3 provides a theoretical basis for considerations of trust relationships in distributed information systems. Chapter 4 discussed the general methodology for analysis and modelling of trust relationships in distributed information systems. The analysis and modelling of trust relationships in federated medical services are based on the theoretical results in the above two chapters. This chapter models several typical trust relationships in the federated medical services and
uses them as examples to discuss the evaluation and consuming of trust. Let us consider the following trust relationships:

1. $T_1 = < R_1, E_1, C_1, P_1 >$. $R_1$ includes general patients; $E_1$ includes general practitioners; $C_1$ includes general practitioner licences; $P_1$ includes that general practitioners perform general medical practices.

2. $T_2 = < R_2, E_2, C_2, P_2 >$. $R_2$ includes cardiac patients; $E_2$ includes cardiologists; $C_2$ includes cardiologist licences and NonnegativeRep reputation; $P_2$ includes that cardiologists perform heart checks.

3. $T_3 = < R_3, E_3, C_3, P_3 >$. $R_3$ includes cardiac surgery patients; $E_3$ includes cardiac surgeons; $C_3$ includes cardiac surgery licences, GoodRep or ExtragoodRep reputation, experience of attending more than two successful heart surgeries in the specified hospital, and the specified hospital is on active list for performing cardiac surgeries; $P_3$ includes that cardiac surgeons perform heart surgeries.

It is assumed that reputation can be calculated by some reputation system in the federated medical services. The reputation will be labelled as NegativeRep, NonnegativeRep, GoodRep, and ExtragoodRep. The active list of hospitals for performing cardiac surgeries is maintained by an authority. Related to cardiac surgery patients and cardiac surgeons, whether or not the specified hospital is on the list is an environment condition.
The TrustEngine described in last chapter is chosen, as the architecture for the trust management in the federated medical services. The outcome of the implementation of TrustEngine is the target trust management system. The target trust management system provides the computing support for trust relationships, described above for trust evaluation, trust consuming, and related management tasks. The next section provides details of the implementation of TrustEngine based on these trust relationships.

6.2 Trust Management System as Implementation of TrustEngine

The information system for federated medical services is a typical distributed information system in an open environment, and crosses multiple administrative or organizational boundaries. It is possible to have established standards and computing utilities related with trust in the information system. The implementation of TrustEngine in federated medical services employs the web services technology. The required computing components in TrustEngine are implemented by corresponding web services. The trust management system is composed of a group of web services that support a range of trust management tasks. The implementation of TrustEngine focuses on the kernel utility web services for trust management tasks in federated medical services. The target trust management system is an integration of these utility web services.
In federated medical services, it is infeasible to have centralized control for the required trust management; the involved applications often span several networks and there are multiple administrative or organizational boundaries; the dynamic properties of trust must be considered and trust must be evaluated and established at real time. The trust management system in federated medical services has a distributed nature, and its computing components can run on different servers with different operating systems. The web services technology provides the required easy integration of computing components, and high degree of interoperability for distributed trust management in federated medical services.

In the implementation, the persistent data storage mechanism is implemented by Microsoft SQL Server 2000. Trust relationships, instances of trust relationships, and trust parameters are maintained in multiple tables in the database that is named “implementation”. This database is the implementation of TrustDatabase in TrustEngine architecture. There is another database that is referred to as “bussdata”. The “bussdata” provides a persistent data storage mechanism for the business information that is out of the scope of TrustDatabase but is used by some computing components in TrustEngine architecture.

There are 17 computing components in TrustEngine architecture. Sixteen of these have been developed as web services except the “Direct Trust Consuming Controller” that has a straightforward function and it is not suitable to be implemented as an independent web service. It is implemented by a function in
“Application Consuming Controller”. The “Direct Trust Consuming Controller” is in the TrustEngine architecture because it logically parallels other computing components of the package ApplicationConsuming (see Section 5.2). In this implementation, the development environment is Visual Studio 2005 on Windows XP. The web services are written in the programming language C# 2.0 with the support of Microsoft .NET Framework version 2.0.

For each web service, there is a .asmx file and a .cs file. The .asmx file defines XML Web service specific attributes used by the ASP.NET parser and compiler. For example, the web service TrustEngineController has the TrustEngineController.asmx file as:

```xml
<%@ WebService Language = "C#" CodeBehind = "~/App_Code/TrustEngineController.cs" Class = "TrustEngineController" %>
```

The TrustEngineController.cs file includes the C# codes for the web service. All business functions and the interfaces of the web service TrustEngineController are defined in the .cs file. In the current implementation of TrustEngine for federated medical services, there are 16 web services corresponding to the generic computing components of TrustEngine architecture. The web services include TrustEngineController in package TrustControl; LocatingTrustController, TrustRelationshipLocator, and AuthenticationController in package LocatingTrust; TrustEvaluationController, CredentialEvaluation, ReputationEvaluation, StoredDataEvaluation, and EnvironmentEvaluation in package EvaluatingTrust; Trust-
ConsumingController, ApplicationConsumingController, CredentialGeneratorConsuming, DataStorageConsuming, SystemConsumingController, TrustEngineConsuming, and AuditingConsuming in package ConsumingTrust. The solution hierarchy of these web services is expressed in Figure 6.1.

Figure 6.1: Solution Hierarchy of Web Services for Trust Management
6.3 Web Services for Trust Management

This section describes the specifications of web services listed in the last section and three examples of XML messages of these web services. In the implementation, these web services are developed and deployed on one server. However, it is possible for them to be developed and deployed on multiple servers. Because of the nature of web services technology, the trust management system based on these web services has minimal changes for the case of a single server, or the case of multiple servers. For some web services, it is necessary to invoke other web services for performing tasks at runtime. Web services technology supports the easy integration of web services and provides interoperability.

The specifications of web services for trust management are listed as the follows:

**TrustEngineController** supports the following operations:

- LocatingTrust
- EvaluatingTrust
- ConsumingTrust

The operation LocatingTrust employs web service LocatingTrustController for finding the trust relationship based on the request from applications. The input is an XML message for a trustor, a trustee, and a requested property. The output is an XML message with details of the requested trust relationship or an XML
message for information of the unsuccessful operation. The operation EvaluatingTrust employs web service TrustEvaluationController for checking whether the conditions of the requested trust relationship can be satisfied or not. This operation accepts the output of the operation LocatingTrust as its input. The output of this operation is an XML message with information about the evaluation result.

The operation ConsumingTrust employs web service TrustConsumingController to deal with how to consume the evaluation result of the requested trust relationship. This operation accepts the output of the operation EvaluatingTrust as its input. This operation performs the consuming tasks and provides an output as an XML message with consuming information.

**LocatingTrustController** has the operation LocatingTrustControllerMR. The input is an XML message for a trustor, a trustee, and a requested property. The LocatingTrustControllerMR employs web service TrustRelationshipLocator for finding the requested trust relationship and employs web service AuthenticationController for the authentication of the trustee. At last, the operation returns a XML message about the requested trust relationship.

**TrustRelationshipLocator** has the operation TrustRelationshipLocatorMR that accepts the input as an XML message for a trustor, a trustee, and a requested property. The operation performs its function by connecting with the database “implementation” for finding the requested trust relationship. It returns an XML message about the requested trust relationship.
AuthenticationController has the operation AuthenticationMR that accepts an XML message for an entity as the input. It performs the function for the authentication of the entity and returns an XML message for the status of the authentication. In the current implementation, the check of authentication is against the data stored in the database “implementation”.

TrustEvaluationController has the operation TrustEvaluationControllerMR that accepts the output of the web service LocatingTrustController as its input. The operation employs and assigns tasks to web services CredentialEvaluation, ReputationEvaluation, StoredDataEvaluation, and EnvironmentEvaluation. The operation returns an XML message with information about the evaluation result.

CredentialEvaluation has the operation CredentialEvaluationMR that accepts an XML message for a trustee and a credential condition as its input. In our implementation, credentials are stored with a fixed path in the system, and the evaluation way is predefined for a specific type of credential condition. Three types of credentials are defined in the current system. They correspond to medical practitioner licences, cardiologist licences, and cardiac surgery licences. When a credential is evaluated, the target trustee, the credential type, the signature of authority, and the expired date are checked. This operation returns an XML message with information about credential evaluation result.
ReputationEvaluation has the operation ReputationEvaluationMR that accepts an XML message for a trustee and a reputation condition as its input. The reputation evaluation is performed and the existing utility of reputation calculation can be employed in this operation. In the current implementation, it is assumed the reputation can be labelled as NegativeRep, NonnegativeRep, GoodRep, and ExtragoodRep. A dummy function is employed to take the place of community-based reputation calculation. This operation returns an XML message with information about the result of reputation evaluation.

StoredDataEvaluation has the operation StoredDataEvaluationMR that accepts an XML message for a trustee and a stored data condition as its input. In the current implementation, the stored data is maintained in the database “bussdata”. The information about the involved cardiac surgeon in the database is retrieved. The evaluation is performed based on the experience of attending successful heart surgeries in the specified hospital. This operation returns an XML message with information about the stored data evaluation result.

EnvironmentEvaluation has the operation EnvironmentEvaluationMR that accepts an XML message for a trustee and an environment condition as its input. In the current implementation, the operation checks whether or not the specified hospital is on the active list for performing cardiac surgeries. This operation returns an XML message with information about environment evaluation result.
**TrustConsumingController** has the operation TrustConsumingControllerMR that accepts the output of the web service TrustEvaluationController as its input. The operation employs and assigns tasks to web services ApplicationConsumingController and SystemConsumingController. The operation returns a XML message with the consuming information.

**ApplicationConsumingController** has the operation ApplicationConsumingControllerMR that accepts the input of TrustConsumingController as its input. The operation has a function to deal with direct trust consuming. The operation employs and assigns tasks to web services CredentialGeneratorConsuming and DataStorageConsuming. The operation returns a XML message with information about the application consuming.

**CredentialGeneratorConsuming** has the operation CredentialGeneratorConsumingMR that accepts the input of ApplicationConsumingController as its input. This operation generates a credential and stores it in a file with a fixed path in the system. The operation returns an XML message with information about the credential generator consuming.

**DataStorageConsuming** has the operation DataStorageConsumingMR that accepts the input of ApplicationConsumingController as its input. This operation stores the evaluation result of a trust relationship in the database “bussdata”. The operation returns an XML message with information about the data storage.
SystemConsumingController has the operation SystemConsumingControllerMR that accepts the input of TrustConsumingController as part of its input, and the output of ApplicationConsumingController as another part of its input. The operation employs and assigns tasks to web services TrustEngineConsuming and AuditingConsuming. The operation returns an XML message with information about the system consuming.

TrustEngineConsuming has the operation TrustEngineConsumingMR that accepts the input of TrustConsumingController as its input. The operation stores the information about trustor, trustee, trust relationship, and consuming time in the database “implementation”. The operation returns an XML message with information about the TrustEngine consuming.

AuditingConsuming has the operation AuditingConsumingMR that accepts the input of SystemConsumingController as its input. The operation stores trustor, trustee, trust relationship, and all application consuming information in the database “bussdata” for the auditing purpose. The operation returns an XML message with information about the auditing consuming.

XML messages are used for all the input and output of the above web services.
These XML messages support a high degree of interoperability and they are important for the whole trust management system. There are quite a big number of formats of these XML messages and not all the details for them are provided in this dissertation. Here, three formats of XML messages are chosen an example for each of them is provided. These examples come from real executions of the web service TrustEngineController. All web services described in this section are involved in these executions.

The first example is the XML message for a successful operation of LocatingTrust of TrustEngineController. The XML message is:

```xml
<?xml version="1.0" encoding="utf-8"?>
<TrustRelationshipForEvaluation xmlns="TrustEngineWS">
  <Truster>trustorone</Truster>
  <Trustee>trusteeone</Trustee>
  <trustrelationship_no>TRREN0001</trustrelationship_no>
  <trusterset>TRUSTOR001</trusterset>
  <trusteeset>TRUSTEE001</trusteeset>
  <conditionset>
    <condition>CRE0000001</condition>
    <condition>RPE0000001</condition>
    <condition>SDE0000001</condition>
    <condition>ENE0000001</condition>
  </conditionset>
</TrustRelationshipForEvaluation>
```
The second example is the XML message for a successful operation of EvaluatingTrust of TrustEngineController. The XML message is:

```
<?xml version=“1.0” encoding=“utf-8”?>
  < truster > trustorone < /truster >
  < trustee > trusteeone < /trustee >
  < trustrelationship_no > TRRENO0001 < /trustrelationship_no >
  < trusterset > TRUSTOR001 < /trusterset >
  < trusteeset > TRUSTEE001 < /trusteeset >
  < EvalStatus >
    < SuccessCREEval xmlns = “TrustEngineWS” >
      < CredentialCondition > CRE00000001 < /CredentialCondition >
      < /SuccessCREEval >
    < SuccessRPPEval xmlns = “TrustEngineWS” >
      < ReputationCondition > RPE00000001 < /ReputationCondition >
      < /SuccessRPPEval >
  < /EvalStatus >
< /EvaluationMess >
```
The third example is the XML message for the successful operation of ConsumingTrust of TrustEngineController. The XML message is:

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

− < ConsumingMess xmlns = "TrustEngineWS" >
<truster>trustorone</truster>

<trustee>trusteeone</trustee>

<trustrelationship_no>TRRENO0001</trustrelationship_no>

<ApplicationConsumingMess xmlns="TrustEngineWS">
  <DirectTrustConsuming ConsumingStatus="OK">
    Return to trust request application
  </DirectTrustConsuming>

  <CredentialGeneratorConsuming ConsumingStatus="OK">
    Credential has been generated based on trust evaluation result
  </CredentialGeneratorConsuming>

  <DataStorageConsuming ConsumingStatus="OK">
    Trust evaluation result has been saved in database
  </DataStorageConsuming>

</ApplicationConsumingMess>

<SystemConsumingMess xmlns="TrustEngineWS">
  <TrueEngineConsuming ConsumingStatus="OK">
    Trust evaluation result has been saved in TrustEngine
  </TrueEngineConsuming>

  <AuditingConsuming ConsumingStatus="OK">
    Trust evaluation result has been saved in auditing purpose
  </AuditingConsuming>

</SystemConsumingMess>
This section describes an online booking system for federated medical services as a demonstration system that employs the set of web services, described in the last section, for its trust management tasks. The major benefit of such a solution is the separation of trust management layer from the online booking system. These web services provide an open and ubiquitous distributed computing platform for trust management tasks.

The online booking system provides patients with an easy way to book medical services. It is assumed that the bookings of medical services are based on requested trust relationships between patients and physicians. The online bookings for general medical practices, heart checks, and heart surgeries are based on different trust relationships of patients and physicians.

The online booking system is a web application built with ASP.NET 2.0. The web service TrustEngineController is employed as a web reference in the web application. Multiple facilities of ASP.NET 2.0 such as web configuration files, ASP.NET Master Pages, ASP.NET application file, and ASP.NET Themes are employed in the online booking system. These features are not the main concerns of this dissertation and more details are not provided. The solution hierarchy of
the online booking system for federated medical services is expressed in Figure 6.2.

Web services that perform the related tasks of trust management are developed and deployed at first. The ASP pages of the online booking system employ these web services for the requested trust management tasks in federated medical services. In the current implementation, the online booking system for federated medical services only directly uses the interfaces of web service TrustEngineController for all involved trust management tasks. The other web services are not directly called by the online booking system, but they are involved through the web service TrustEngineController.

The three trust relationships described in Section 6.1 are involved in the demonstration system. These trust relationships cover all the four generic trust mechanisms including credential, reputation, stored data, and environment parameters. It is assumed that all the three generic application consuming ways are required when $T_3$ is required. All the web services described in last section can be invoked to serve trust management tasks activated by online bookings.

The online booking system has graphic user interfaces to capture booking information. The first page of the system is the introduction page of the online booking system (see Figure 6.3). Then, it is a login page (not shown) that is followed by the main menu page (see Figure 6.4). On the main menu page, a user can choose a medical service as General Medical Service, Heart Check, or Heart Surgery. By clicking the chosen service, hospitals that can provide the
Figure 6.2: Solution Hierarchy of Online Booking System
required service are listed and the user can choose a hospital for the required service (see Figure 6.5). By clicking “Next Step” button, the booking page is displayed and the user can choose the doctor, provide patient ID, choose a date to book the required medical service (see Figure 6.6). By clicking the “Booking” button, the system returns the status message for the current booking (see Figure 6.7).

Figure 6.3: Introduction Page of Online Booking System
Online Booking System for Federated Medical Services

General Medical Practice
Heart Check
Heart Surgery
Log Out

Note: The Online Booking System for Federated Medical Services is a demo system for the implementation of TrustEngine Architecture with multiple web services for related trust management tasks.

Copyright, University of Western Sydney

Figure 6.4: Main Menu Page of Online Booking System
Online Booking System for Federated Medical Services

Please Select Hospital for General Practice Service

- Westmead Hospital
- Penrith Hospital
- Parramatta Family Medical Center
- Johnson Medical Center

Hospital Selected:

NSW00001 - Westmead Hospital

Note: The Online Booking System for Federated Medical Services is a demo system for the implementation of TrustEngine Architecture with multiple web services for related trust management tasks.

Figure 6.5: Hospital List Page
Figure 6.6: General Medical Practice Booking Page
Figure 6.7: Booking Result Message Page
When a patient books a general medical practice service through the online booking system, trust relationship $T_1$ in Section 6.1 is involved. The web service TrustEngineController is invoked as the controller of all related trust management tasks. At runtime, a series of web services are invoked including LocatingTrustController, TrustRelationshipLocator, AuthenticationController, TrustEvaluationController, CredentialEvaluation, TrustConsumingController, ApplicationConsumingController, SystemConsumingController, TrustEngineConsuming, and AuditingConsuming. These web services provide operations for the whole set of trust management tasks initiated by the involved trust request. The required trust relationship is located at first. Then the medical practitioner licence associated with the involved doctor is evaluated. The online booking system consumes the evaluation result of the trust relationship directly, and the consuming status of the trust relationship is stored in both TrustEngine database and auditing database. The flowchart for trust management tasks related with trust relationship $T_1$ is shown in Figure 6.8.

When a patient books a heart check through the online booking system, trust relationship $T_2$ in Section 6.1 is involved. All the web services in the above general practice booking are invoked in the same way. The only difference is that an additional web service ReputationEvaluation will be invoked after the invocation of CredentialEvaluation. The flowchart for trust management tasks related with trust relationship $T_2$ is shown in Figure 6.9.

When a patient books a heart surgery through the online booking system,
Figure 6.8: Trust Management Flowchart for $T_1$
Figure 6.9: Trust Management Flowchart for $T_2$
trust relationship $T_3$ in Section 6.1 is involved. This trust relationship is a complicated one which includes all the four types of trust mechanisms in our unified trust management framework. Beyond the direct usage of the trust evaluation result by the online booking system, credentials are generated and the trust evaluation results are stored in database. These two additional ways of trust consuming provide a potential for further management and usage of these trust evaluation results. In order to book a heart surgery, all web services described in Section 6.3 are invoked for performing trust management tasks. The flowchart for trust management tasks related with trust relationship $T_3$ is shown in Figure 6.10.

6.5 Summary

This chapter has provided the details of the trust management system for federated medical services that implements the TrustEngine architecture proposed in the last chapter. The development of the trust management system for federated medical services has been automated to a high level with the help of the TrustEngine architecture and the generic details of its computing components. The trust management system is composed of multiple web services for a range of trust management tasks related with trust locating, trust evaluating, and trust consuming.

An online booking system for federated medical services has been developed
Figure 6.10: Trust Management Flowchart for $T_3$
using a set of ASP pages. The separation and linkage between the online booking system and trust management system has been defined. The online booking system delegates all the trust management tasks to the trust management system by activating the appropriate web services.
Chapter 7

Trust Management and WS-
Trust/Role Models/CardSpace

This chapter provides three prospective views on the trust management approach including trust management in web services, the comparison between the trust management approach and role based access control, and trust management for digital identities.

Some specific trust relationships are discussed for the access control of web services or web service methods in the service oriented architecture paradigm. A new trust management layer is introduced in the web services architecture layers as an extension of WS-Trust. A web service trust management architecture is proposed with additional computing components that are beyond the trust management architecture proposed in Chapter 5. Some further discussion are also provided about trust management in web services.

The trust management approach is compared with role based access control. An overview is provided of role based access control and the prospective view of the trust management approach is discussed as an extension of role based access
control from a range of different aspects.

In order to discuss trust management for digital identities, an overview is provided of the identity metasystem and CardSpace [83, 26, 82, 6]. Based on the general trust management approach and the research results about trust management in web services, some general guidelines are provided for the analysis of trust requirements and development of a generic trust management system for digital identities that can be seen as an extension of CardSpace.

Section 7.1 proposes a trust management approach in web services as an extension of WS-Trust. Section 7.2 discusses the relationship between the trust management approach and existing role based access control. Section 7.3 discusses the trust management extension of CardSpace. Section 7.4 provides a summary of this chapter.

7.1 Trust Management in Web Services as Extension of WS-Trust

7.1.1 Introduction

The W3C defines a web service as “a software system designed to support interoperable machine to machine interaction over a network”. Web services are self-contained modular applications that can be invoked across the Web and they have self-describing ability. Web services communicate with each other
§7.1 Trust Management in Web Services as Extension of WS-Trust

using XML messages across the boundaries of different enterprises or organizations. On the one hand, web services technologies make distributed computing components to be easily integrated across business boundaries and computing platforms. On the other hand, the web services technologies can introduce a high degree of complexity of runtime operations. In web services, different kinds of business partners could be involved, and it may be possible for web services to require other services offered by third parties; the providers of web services may belong to different security domains; and the users of web services may be not predetermined. Undoubtedly, security is a key area to be addressed for delivering integrated, interoperable solutions under web services architecture.

The core specifications of web services technologies include SOAP, UDDI, and WSDL. Other standards and specifications of web services are supposed to build upon these core specifications. To address the security and reliability issues, several efforts have been made by organizations such as Organization for the Advancement of Structured Information Standards (OASIS) and Web Services Interoperability Organization (WS-I).

IBM and Microsoft [28] have proposed a technical strategy and roadmap to deal with security in web services. The security roadmap is composed of a set of web services security specifications that are shown in Figure 7.1. These specifications provide building blocks for web services and they are based on a SOAP extensibility model (except the WS-Authorization, that is currently not available). These specifications are designed to be composed with each other to provide a
rich messaging environment.

In web services paradigm, there is no existing integrated solution and architecture that addresses access control and related trust management for web services in a consistent manner. Previous chapters provided a formal definition of a trust relationship, discussed the taxonomy framework of trust, the methodology for analysis and modelling of trust relationships, and a unified framework of trust management. These research results will be employed as a general foundation for the concerns about trust management in web services. Beyond classical access control systems, the proposed trust management approach includes multiple trust mechanisms and multiple ways of trust consumption. In particular, the trust management architecture proposed in Chapter 5 provides a basis for discussions about trust management in web services paradigm. This section discusses trust relationships in web services, trust management architecture in web
services, and web services architecture layers. Some further discussion is also provided about trust management in the web services paradigm.

### 7.1.2 Trust Relationships

The analysis and modelling of trust relationships is the starting point for the development of a trust management layer in web services. How to define and model trust relationships between involved parties or computing components are important and challenging issues in the design of web services. The formal model of trust relationship and the unified taxonomy framework of trust were described in Chapter 3 and the general methodology for analysis and modelling of trust relationships was discussed in Chapter 4. These results have provided a solid theoretical base for the understanding of possible trust issues and the analysis and modelling trust relationships in web services paradigm.

Trust relationships play an important role in distributed information systems, and they are normally used as preconditions for some related operations or transactions. The specific characteristics of web services will provide some additional requirements for the involved trust relationships. In web services paradigm, there are some specific trust relationships that are related to access control of web services or web service methods. The web services or web service methods are the resources to be protected. These trust relationships have a set of specific characteristics that are only existent in the web services paradigm. They capture the requirements for access control of the involved web services or web services
methods. Based on the formal model of trust relationship in Section 3.1, there are the following concerns in the analysis and modelling of trust relationships in the access control for web services:

- The trustor set is normally modelled as a set of web services or a set of web service methods. The case of one element represents the specific situation with one web service or web service method.

- The trustee set includes the entities that request the web services or web service methods in the trustor set. The trustee set may be composed of a set of web services or other types of requesters.

- The condition set may include the trust mechanisms such as credentials, reputation, data storage, and environment parameters. For web services, security tokens in SOAP messages defined in WS-Trust provide evidences for some conditions in the condition set and WS-Trust is the standard trust mechanism at the messaging level. However, it is possible to have conditions that are irrelevant to these security tokens.

- The property set may include the basic operations [68] on web services or web services methods such as (1) execute: access and execute the web services or web service methods; (2) update: update web services or web services methods. The executable codes and metadata could be updated. This property is normally useful for administrators and developers for updating
computing components; (3) find: provide search capabilities for properties and metadata associated with the web services or web service methods.

There are some trust relationships that are not in the scope of the above specific discussion on access control for web services. In the web services paradigm, these trust relationships may not be strongly coupled with the web service technologies. These trust relationships can be included in the trust management solutions for web services as well.

Here there is only a general discussion about trust relationships in web services paradigm. There may be complex situations of trust relationships in web services. Different web service providers may have different security needs regarding the access control of web services based on required trust relationships. These trust relationships take the place of the classic identification requirements for principals, and extend the identity-based control to a much more flexible trust-relationship-based control. The trust management approach, based on trust relationships, should have the ability to facilitate the integration of different trust mechanisms for the access control of web services. Security tokens of WS-Trust are supported but other kinds of trust mechanisms can be employed as well. In our trust management approach, the analysis/ modelling of trust relationships and the development of trust management systems are covered in a consistent manner. The trust management architecture for web services that will be described in next sub section focuses on the processing of specified trust relationships and it can provide another view angle for the analysis and modelling of
trust relationships in web services.

7.1.3 Trust Management Architecture

Section 2.1.5 provided a detailed review of WS-Trust. The goal of WS-Trust is to enable applications to construct trusted [SOAP] message exchanges [91]. WS-Trust provides a solution to establish trust relationships based on security tokens from security token services that act as trusted authorities. The web service trust model defined in WS-Trust is based on the proof of a set of claims (e.g., name, key, permission, capability, etc.) in incoming messages. Security token services behave as trusted authorities to issue security tokens. When web services are used for conducting e-commerce and providing interoperability across organisations, there are a broad range of trust situations that go beyond the simple case defined in the trust model of WS-Trust. Trust mechanisms, other than security tokens from trusted third parties, may be required to establish trust relationships between involved entities. It is reasonable to expect that any trust mechanism in distributed information systems should be embraced in the web services paradigm.

Chapter 5 proposed a unified framework for trust management in distributed information systems. The trust management solution targets at a trust management layer that can be separated from distributed information systems. The proposed unified framework for trust management provides an infrastructure and tools for the development of this trust management layer in distributed informa-
tion systems. The trust management layer will deal with trust management tasks that can be separated from applications.

The trust management architecture for web services will be built on the general trust management architecture for distributed information systems according to the specific characteristics of web services. The existing security standards and specifications of web services should be adapted in the trust management architecture for web services. In other words, the target trust management architecture for web services is an extension of WS-Trust that can cover a broad range of trust mechanisms and not just the security tokens in SOAP messages. For instance, a broad range of trust mechanisms including credentials, reputation, data storage, and environment parameters could be involved in trust management solutions in web services.

The trust model of WS-Trust is based on a process whereby a web service can require that an incoming message prove a set of claims (e.g., name, key, permission, capability, etc.). WS-Trust provides the support for more specific security models such as identity-based authorization, access control lists, and capabilities-based authorization at the messaging level. When considering trust management in web services paradigm, WS-Trust should be integrated in the general trust management architecture proposed in Chapter 5. The handling of SOAP messages that include security tokens, and other trust-relevant information, is a specific task in web services paradigm. It is necessary to have a computing component in the trust management architecture to adapt the incoming
WS-Trust SOAP messages and generate the outgoing WS-Trust SOAP messages. This computing component is referred to as “WS-Trust Message Handler”.

In web services paradigm, there is a rich messaging environment supported by multiple modular SOAP-based security specifications such as WS-security, WS-Policy, WS-Privacy, WS-SecureConversation, and WS-Federation. These specifications provide building blocks for a broad range of security needs at the messaging level. A broad variety of contextual information of web services is embedded in these building blocks of SOAP-based messages. For web service, some contextual information could be critical to trust decisions. In particular, the contextual information in the building blocks defined by web services security specifications may play an important role in some trust relationships in web services. It is necessary to have a computing component to capture and manage trust-relevant contextual information in web services. This computing component is referred to as “TrustContext Handler”.

The computing components “WS-Trust Message Handler” and “TrustContext Handler” together with the computing components “LocatingTrust”, “EvaluatingTrust”, and “ConsumingTrust” in the original TrustEngine (defined in Chapter 5), provide a top level view of the web services trust management architecture as shown in Figure 7.2.
7.1.4 Web Services Architecture Layers

In web services paradigm, multiple web services standards and specifications have been published and adopted, particularly there has been a stack of web services security specifications (see Figure 7.1, as an exception, WS-Authorization is not available). The proposed trust management layer for web services is on the top of the stack of web services security specifications. Web services security specifications can be employed to provide modular support to the trust management layer for web services. On the top of the trust management layer, it is the web services coordination layer. The trust management layer will serve WS-Coordination and WS-Transaction layers. This leads to the web services architecture layers shown
The trust management layer is more complicated compared with other entities. The other entities are specifications for proving modular building blocks in web services paradigm. These specifications target well-defined technical aspects of web services. The target of trust management is more generic and trust management is normally a challenging task because of possible complicated business situations in real information systems. Compared with normal distributed information systems, web services will provide additional trust requirements that should be addressed by the trust management layer. A broad range of trust mechanisms in distributed information systems must be embraced in the web

<table>
<thead>
<tr>
<th>Web Services Coordination Layer</th>
<th>WS-Transaction</th>
<th>WS-Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Services Security Layer</td>
<td>Web Services Trust Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WS-SecureConversation</td>
<td>WS-Federation</td>
</tr>
<tr>
<td></td>
<td>WS-Trust</td>
<td>WS-Policy</td>
</tr>
<tr>
<td></td>
<td>WS-Security</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web Services Messaging Layer (SOAP)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 7.3:** Web Services Architecture Layers
services paradigm.

Web services standards must be adapted to the trust management layer for web services. WS-Trust, WS-Policy, WS-Privacy, WS-SecureConversation, and WS-Federation are modular SOAP-based specifications based on the SOAP extensibility model. These specifications are designed to be composed with each other to provide a rich messaging environment. In web services, the building blocks defined by these SOAP-based specifications may be involved in the trust management layer. The trust management layer should facilitate the integration of the building blocks, defined by the above web services specifications, and computing components for trust management tasks that are not directly related with SOAP messages.

7.1.5 Discussion

Trust management is a challenging task in web services paradigm. The complexity of the trust management layer of web services could be very high because web services normally run over open networks and across multiple security domains; web services normally have a highly dynamic nature, and the involved entities may have a broad variety of degrees of familiarity with each other; web services may have complicated contextual information based on the rich messaging environment of web services; and hence trust related business requirements could be very complicated.

This section presents the concerns towards a comprehensive trust manage-
ment solution in web services paradigm that is based on the general theoretical results about trust relationships and trust management in previous chapters of this dissertation. The results in this section can assist developers in specifying and modelling trust relationships in web services, and provide high level architecture for the development of trust management layer in web services. The analysis/modelling of trust relationships, and the development of trust management systems, are considered in a consistent manner. The goal of the trust management architecture in web services is to substantially increase the level of automation in development and deployment of the trust management layer in web services. The proposed trust management architecture in web services provides a foundation whereon trust management systems can be developed more easily than before. The trust management architecture in web services covers a broad variety of mechanisms for trust establishment in web services paradigm, with the ability to adapt the web services security specifications defined in security roadmap of web services [28].

The trust management layer in web services extends WS-Trust and covers existing trust mechanisms including credentials, reputation, data storage, and environment parameters. The results in this section aim at providing high level guidelines, rather than a panacea, for the development of trust management solution in web services. It is hoped that the research results described in this section can provide a solid starting point and useful high level tools for the development of the trust management solution in web services paradigm.
7.2 Trust Management as Extension of Role Models

7.2.1 Role Based Access Control

Role-Based Access Control (RBAC) model was first presented by D. Ferrailo and R. Kuhn in 1992 [40]. Actually, the role based access control can be traced back much earlier than when it was presented as a formal model. The features of roles have been employed in some products in 1970s and 1980s such as IBM’s RACF. RBAC model has received great interest as a powerful and popular approach to management of access control in information systems. The role model provides tools for managing access control in a complex environment, with a large number of users and data items to access. RBAC can reduce the complexity and cost of security administration in information systems. In the history of security technology, RBAC is a newer alternative approach to mandatory access control (MAC) and discretionary access control (DAC) for restricting system access to authorized users. RBAC has the power to express a wide range of security policies including the MAC and DAC.

Users, roles, and permissions are the basic components in all RBAC models [40, 88, 89, 108, 90, 42, 43, 61]. Users are the entities who will get access to objects and resources in information systems. In ANSI standard, users are defined to be human beings [61]. Roles are normally designed according to the job func-
tions in companies or organizations such as teller, manager, and administrator. Permissions are access control of objects with operations.

In RBAC models, there are three important relationships: the user-role relationship is a many to many relationship that maps users to roles; the role-permission relationship is a many to many relationship that maps permissions to roles; the role-role relationship is a many to many relationship that allows building role hierarchies [40, 89, 41, 107, 84, 90, 42]. The advantage of role hierarchies is to improve administrative efficiency. With RBAC, roles are designed and created for various job functions performed in a company or organization; permissions are granted to specific roles; and users are assigned to roles based on their specific job responsibilities and qualifications. In RBAC models, security administration is simpler as it is based on roles rather than on permissions. The permissions associated with a role are more stable than the particular combination of users and permissions brought together by a role [108].

Session [39, 108] is an important concept in RBAC models that defines a one to many mapping from a user to roles, for modelling the runtime activation of roles of a user. A session is established when a group of roles is activated for a specific user. A session can have multiple roles, and a user can have multiple sessions opened simultaneously. The user can obtain permissions based on the active roles in all opened sessions of the user.

To address the issue of conflict of interest, advanced RBAC models use separation of duty constraints [45, 115, 69, 47] to identify sets of roles that should not
be assigned to the same user. There are static and dynamic ways for achieving the separation of duty. In the static separation of duty, constraints are enforced in user-role assignment mapping to prevent a user being assigned to two or more conflicting roles at the same time. In the dynamic separation of duty, constraints are enforced by preventing simultaneous activation of roles in a session when these roles are in conflict with each other. The dynamic separation of duty provides greater flexibility than the static separation of duty.

7.2.2 Extensions of Role Based Access Control

Traditional RBAC models can not cover the broad variety of practical scenarios related with access control, in particular, when the dynamic aspects of roles and contextual information must be considered. Multiple extensions of RBAC models have been proposed. E. Bertino et al [8, 64] have proposed a temporal role-based access control model that allows specification of a comprehensive set of temporal constraints. Li et al [73] have introduced RT, a family of role-based trust management languages for representing credentials and policies in distributed authorization. RT can be viewed as an extension of RBAC by combining some features of trust management using credentials, and some notations denoting relationships, in information systems. RT uses roles as a central notion and it allows selective role activation and delegation of these activations. In IBM’s trust establishment (see Section 2.2), RBAC is extended by using certificates as input to identify roles [53]. The trust establishment system handles multiple certificates
for a subject and maps the subject to a role based on its certificates. Actually, all
the above extensions of RBAC are based on acceptance of temporal constraints
or certificates in the assignment of roles.

The trust management solution, including the taxonomy framework of trust
and unified framework of trust management, has provided additional extensions
to RBAC in multiple aspects. Based on our formal model proposed in Section
3.1, a trust relationship has four tuples as trustor set, trustee set, condition set,
and property set. The trustee set and property set are corresponding individually
to a role and a set of permissions in role models. In role models, role-permission
mapping can be viewed as trust relationships with “system” as the generic trustor
set, roles as the trustee sets, and permissions as property sets. The formal model
of trust relationship can be regressed to the role models.

In role models, a role provides an abstraction of users for the assignment
of permissions. Roles express the abstraction of system’s view about users and
can bring advantages in security analysis and administration. Similar to the
system’s view of users in role models, a generalized trustors’ view about trustees
is considered in the formal model of trust relationship. Similar to roles for users,
trustor set and trustee set can be considered as abstractions for trustors and
trustees respectively. Li et al have described the usage of roles to represent
permissions in $RT_2$ [73] and the o-sets can be viewed as examples of trustor sets
under our formal model of trust relationship. The facilities related with roles
such as role hierarchy can be adopted in trust relationships.
In the formal model of trust relationship, the property set covers broader situations than permissions in the role models that are limited in the scope of access control of resources. The property set can include trusted targets that are beyond the permissions in role models. It is impossible for all attributes and some actions in property set to be included in permissions in the role models. Trust relationships can provide more flexible trust claimants that are beyond role-permission mappings in role models. Some facts can be included in property sets of trust relationships, and these facts themselves are not permissions.

In role models, roles provide an abstraction layer between users and permissions for facilitating access control management. The role-permission mappings are normally statically defined. Under the umbrella of RBAC, sometimes it is difficult to design suitable roles for dynamic and context based trust requirements for access control. The formal model of trust relationship includes a condition set for capturing dynamic trust requirements. A broad range of trust mechanisms may be involved in the condition set. The dynamic and context based trust requirements can be easily expressed by the condition set.

RBAC was originally developed for access control in a single organization. In decentralized collaborative systems, resources, and subjects requesting them may belong to different security domains. There have been some research projects that have extended traditional role models by adding computing components to accept credentials [53, 73]. The trust management solution has gone beyond the existing extensions. The research results in role models such as role abstrac-
7.3 Trust Management Extension of CardSpace

7.3.1 Identity Metasystem and CardSpace

In the real world, there are different types of identities such as drivers’ licenses, passports, birth certificates, student cards, and credit cards. Different types of identities are used in different contexts. For example, a bank card can be accepted by the bank’s ATM but it is out of the context when crossing borders between countries. In the Cyberspace, there are a broad variety of digital identities that simulate the real world’s identities and there are some digital identities that do not have corresponding entities in the real world. The digital identities are associated with different information and are expressed in different ways. A set of principles, referred to as “Laws of Identity” [25], has been proposed and refined as key concerns in the design of the architecture of an identity metasystem. With the boom of E-commerce and E-services, digital identities are necessary and some challenging issues are emerging such as online identity theft, fraud, and leakage of privacy.

CardSpace, formerly known as InfoCard, is a client software for identity metasystem developed by Microsoft. CardSpace is designed as an identity metasystem [83, 26, 82, 6] to provide a reliable way to establish who is connecting with
whatanywhere on the Internet. CardSpace uses a consistent way to deal with the portfolio of digital identities. It targets a standards-based solution for storing and delivering diverse digital identities for the Internet. Microsoft claims that CardSpace can provide the support for any digital identity system, consistent user control of digital identity, replacement of password-based web login, and improved user confidence in the identity of remote applications.

In any digital identity system, there are three generic entities, namely the subject, identity provider, and relying party. A subject is an entity that is associated with the digital identity about whom claims are made. Examples of subjects include people, companies, applications, and machines. An identity provider issues digital identities associated with the subjects. Digital identities issued by different identity providers carry different information for the claimed identity. A relying party is an entity that requires digital identities. Normally, a relying party is an application that frequently uses digital identities for authentication and authorization. A typical example of a relying party is a web service of an online bookstore that accepts digital credit cards for payment.

CardSpace provides an intuitive user interface for working with digital identities. Digital identities can be selected under a consistent user control. In CardSpace identity selection screen [26], a digital identity is displayed as an information card that can potentially present to a relying party. Information cards are selected and sent to relying parties, while the technical complexity of information cards is hidden.
CardSpace allows replacing password-based web login with using information cards. Relying parties use information cards for authentication and authorization rather than authenticating users with passwords. Windows CardSpace, and the identity metasystem, are helpful in reducing the phishing problem by tricking users into revealing passwords. CardSpace can improve user confidence in the identity of web applications by using certificates with more information, rather than the traditional certificates such as Secure Sockets Layer certificates. The new certificate can include the name, location, and logo of the organization that it has been issued to. CardSpace has graphic user interfaces and functions to require every user to approve the use of every identity provider and relying party that he or she wishes to access.

7.3.2 Trust Management for Digital Identities

The client software CardSpace provides a secure and interoperable identity layer for the Internet. CardSpace has a unified interface for selecting digital identities, and provides support to store and deliver digital identities in a secure manner. The information about attributes of digital identities is shared with the relying parties to facilitate authentication of users and access control. Digital identities normally convey sensitive information of their subjects. There are some critical users’ privacy issues. Problems like identity theft, and sensitive information protection, have pushed users to request more control on their digital identities. The disclosure of digital identities must be under the control based on the satisfaction
of related trust requirements. The analysis and modelling of trust relationships, and trust management for digital identities, become important and challenging tasks that are beyond the original objectives in the CardSpace. It is believed that it is necessary to add trust management extension to CardSpace for building up a secure identity layer for the Internet. The trust management extension to CardSpace is composed of computing components that carry out the trust management tasks associated the usages of digital identities.

The analysis of trust requirements and development of the trust management system for digital identities can be a quite complex and costly task. Digital identities are regarded as the same type of objects with common characteristics from the viewpoint of CardSpace. Multiple trust mechanisms may be involved in the evaluation and establishment of trust relationships that are required for the releases of digital identities to relying parties. For the trust management extension of CardSpace, it is desirable to have a general methodology for the analysis and modelling of trust relationships related with the digital identities, and have a suitable trust management architecture that can facilitate the development of the trust management system for digital identities.

The research results described in the previous chapters provide a solid base for developing a trust management solution for digital identities.

The taxonomy framework of trust in Chapter 3 and the general methodology for analysis and modelling of trust relationships in Chapter 4 provide a set of tools for the analysis and modelling of trust relationships for digital identities.
In the identity metasystem, there are a number of digital identities, users of digital identities, and relying parties. The users are normally the subjects of digital identities but there can be other entities. For example, a user of a digital identity is a proxy of the subject of the digital identity who can make the decision on the usage of the digital identity. A typical trust relationship has the trustor set as a group of users, trustee set as a group of relying parties, condition set as a group of trust conditions, and property set as a group of digital identities. The formal model of trust relationship proposed in Section 3.1 provides a commonly used notation and data structure for trust management for digital identities.

The unified framework for trust management in Chapter 5 can provide high level architecture and generic details of computing components to enable the development of the trust management system for digital identities. At the target, the secure identity layer, multiple trust mechanisms including reputation, credentials, local data storage, and environment parameters can be integrated easily under the unified framework for trust management. The development of the trust management system for digital identities is the implementation of general trust management architecture with generic considerations of CardSpace and specific considerations of business requirements for trust management tasks.

There is a close relationship between CardSpace and web services technology. The users and relying parties in CardSpace can be web services. It is possible for a rich messaging environment provided by multiple web services specifications such as WS-Security, WS-Trust, WS-MetadataExchange and WS-SecurityPolicy.
to integrate with CardSpace. It is necessary to deal with SOAP messages defined by web services specifications when trust management extension of CardSpace is considered. We suggest that the trust management extension of CardSpace share the same high level architecture for web services trust management including computing components “WS-Trust Message Handler” and “TrustContext Handler” described in Section 7.1. However, the trust management extension of CardSpace targets trust management tasks about digital identities rather than web services. The characteristics of CardSpace and digital identities must be considered in analysis and modelling of trust relationships and the development of the trust management system. The LocatingTrust is for finding the required trust relationship based on the user, relying party, and selected digital identity from CardSpace. The computing components of EvaluatingTrust described in Chapter 5 should be customized based on trust mechanisms involved. The ConsumingTrust can pass the evaluation results of trust relationships to CardSpace for control on the disclosure of digital identities to relying parties. The ConsumingTrust may include other computing components described in Chapter 5. The development of trust management for digital identities can be viewed as an implementation of trust management architecture. Here only some high level design discussion is provided for trust management extension of CardSpace.
7.4 Summary

This chapter discussed the trust management solution in web services, the relationship between our trust management solution and role models, and trust management for digital identities as an extension of CardSpace.

A new trust management layer in web services architecture layers has been introduced to deal with trust management tasks that can be separated from applications. The trust management layer is the place for the integration of all trust mechanisms and web services messages supported by web services protocol stack. The analysis and modelling of trust relationships have been discussed, and, in particular, major specific concerns are listed about trust relationships for the access control of web services. A trust management architecture of web services has been proposed by considering specific characteristics of web services based on the general trust management architecture.

The relationship between the trust management solution and existing role models has been discussed. A detailed discussion has been given on how the trust management solution extends RBAC in multiple aspects.

Trust management extension of the digital identity system CardSpace has been discussed. Also discussed was the general methodology for the analysis and modelling of trust relationships related with digital identities, and the trust management architecture that can facilitate the development of the trust management system for digital identities.
Chapter 8

Conclusions

Trust has become a hot research topic with the migration of systems from centralized to distributed environments, and with the boom in services and transactions over the Internet. A typical distributed information system is composed of many interconnected computing components and resources that belong to different domains without central control. Trust has been recognized as a challenging issue with immense significance. Following the philosophy of separating the trust management layer from applications, this dissertation proposes a trust management approach as an overall solution for related trust issues. After giving an introduction and an overview of several major conceptions of trust and several well-known trust management systems, this dissertation provides the research results that include a unified taxonomy framework of trust, a methodology for analysis and modelling of trust relationships, a unified framework for trust management, a trust management system for federated medical services, and three prospective views on the trust management approach providing extensions of WS-Trust, role based access control, and CardSpace digital identity system.

This chapter concludes this dissertation. Section 8.1 highlights the principal
conclusions. Section 8.2 provides some discussion on future research directions related with the work presented in this dissertation.

## 8.1 Contributions

The trust management approach presented in this dissertation provides tools and guidelines for building up a trust management layer that can be separated from the applications. The trust management approach covers both the analysis and modelling of trust requirements, as well as the development of trust management systems in a consistent manner. This dissertation provides concepts, enabling tools, scenario examples, and a general methodology and guidelines for the analysis and modelling of trust relationships. This dissertation also provides the research results on the development of trust management systems including a trust management architecture, an implementation example of trust management architecture, a trust management layer for web services, and trust management extension of CardSpace. The proposed trust management approach has the ability to embrace existing trust standards and computing components in a consistent manner.

The dissertation provides the contributions on trust in distributed information systems in the following different aspects:

- A formal model of trust relationship is defined to capture a range of commonly understood notions of trust in distributed information systems. The
formal model of trust relationship is the cornerstone of the research presented in this dissertation. With a strict mathematical structure, the formal model has a strong expressive power, and provides a solid foundation to define properties of trust in a broad variety of situations. The formal model covers different trust mechanisms and allows them to be integrated under the same umbrella. The formal model addresses trust management challenges in open and distributed environments and extends existing role models in multiple aspects.

- A unified taxonomy framework of trust is proposed based on the formal model of trust relationship. The classification of trust is discussed and trust relationships are categorized into two layers. A hierarchy of trust relationships is provided based on the nature of four tuples of a trust relationship. A trust scope label is defined to model the properties of the scope and diversity of trust relationships. A set of definitions for the properties of direction and symmetry of trust relationships is outlined. A set of operations on trust relationships and relative types of trust relationships is presented based on the relations of trust relationships. The taxonomy framework of trust is composed of a series of definitions about the properties of trust relationships. The taxonomy framework of trust provides terminologies, enabling tools, and scenario examples for the analysis and modelling of trust relationships in different aspects.
• A general methodology for analysis and modelling of trust relationships is presented. The general methodology provides guidelines for analysis and modelling of trust relationships in their whole life cycle. In the general methodology, the analysis and modelling of both individual trust relationships and the whole set trust relationships are discussed. The linkage between analysis/modelling trust relationships and trust management systems is also discussed.

• A unified framework for trust management is proposed with a trust management architecture as its kernel part. The trust management framework leverages established trust standards, and it covers a broad variety of trust mechanisms including reputation, credentials, local data, and environment parameters. The trust management architecture is composed of computing components for handling data storage, trust request, trust evaluation, and trust consuming in a consistent manner. The architecture includes all the computing components for trust management tasks that can be separated from applications. The trust management architecture provides interfaces to embrace existing computing components in the systems for performing its tasks. The architecture is module-based and open to accept new computing components for trust management tasks. The details of generic computing components in the trust management architecture are provided. With the help of the proposed trust management architecture, the devel-
opment of trust management systems in the real world can be automated to a substantially high level.

- A trust management system for federated medical services is devised, as an implementation example, to illustrate the proposed trust management architecture and its implementation. The trust management system is composed of a series of web services that support a range of trust management functions. The example shows the power of the proposed trust management architecture for facilitating the development of trust management systems with low cost.

- A new web service trust management layer is introduced within the web services architecture layers. A trust management architecture in web services is devised with the ability to integrate the message building blocks supported by web services protocol stack and other trust mechanisms. The comprehensive trust management solution in web services paradigm is based on the general theoretical results about trust relationships and general trust management architecture proposed earlier. The trust management approach for web services addresses the specific characteristics in web services, and provides a high level architecture and guidelines for the development and deployment of trust management layer in web services.

- A trust management extension to CardSpace is introduced. The analysis and modelling of trust relationships and development of trust management
systems for digital identities are discussed.

Some parts of the above research work have been published in [1], [2], [3], [4], [5], [6], and [7] of Appendix A.

8.2 Future Work

The proposed trust management approach is still at the developing stage, and it only provides a starting point for challenging trust management solutions in the real world. It is necessary to undertake further research related with the trust management approach presented in this dissertation. This section discusses some potential future work.

The properties of trust relationships that have been defined are a partial list. Other properties of trust relationships could be defined based on the upcoming requirements from both real information systems and other academic research. It is an open research area to study the properties of trust from different angles. For example, the notions for delegation, propagation, and transitivity of trust have not been defined in the current taxonomy framework; similar to role models, it may be necessary to introduce some hierarchy structures of trustor sets and trustee sets considering the relations of trust relationships; similar to separation of duties in role models, it may be necessary to introduce some notations for constraints on trust relationships. These notations for properties of trust relationships will become new elements of the unified taxonomy framework.
of trust proposed in this dissertation.

As reviewed in Section 2.2, the trust establishment in current trust negotiation approach is only based on the exchanges of credentials. It is possible to require other trust mechanisms such as reputation, storage data, and environment parameters to be involved in some steps of trust negotiation processes. From the viewpoint of trust negotiation, the research results provided in this dissertation only focus on one step of trust relationships rather than the whole negotiation process. However, the proposed trust management framework provides a solid basis for the future trust negotiation framework with the ability to embrace multiple trust mechanisms.

The trust management approach presented in this dissertation provides an extensible general framework. In the real world, there is a vast number of distributed information systems with different trust requirements. It is believed future research is required to study the specific characteristics of business requirements and emerging technologies in these information systems. It may be necessary to extend the current unified trust management framework to embrace emerging business requirements and technologies. In the proposed unified trust management framework, four types of trust mechanisms have been discussed, namely credentials, reputation, stored data, and environment parameters. There has been a vast amount of research work about credentials and reputation systems. However, there is still a clear need to have further research on credentials and reputation systems with the dramatic growth in the diversity of online trans-
actions and services. More requirements and issues are still emerging for trust management approaches.

When multiple trust relationships are defined in the security policy of an open system, policy conflicts can arise. Conflicts can also arise due to differences in interests among trustors and trustees in systems. Further research work is needed to represent and resolve such conflicts in trust relationships in distributed information systems.

Finally, as possible future work, software tools need to be developed to aid the analysis/modelling, reasoning, establishment, and monitoring of trust relationships in some specific information systems. It is also necessary to develop tools for specifying trust policies and relationships using a policy based language. This is particularly the case when it comes to large scale distributed systems.
Appendix A

Publication List from This Dissertation


Appendix B

Source Codes

The source codes of web services for trust management and the online booking system for federated medical services can be found in the attached CD.
This page is intentionally left blank.
Bibliography


[33] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing


2000.


232 Bibliography


[81] Ueli Maurer. Modelling a public-key infrastructure. In 1996 European Sym-
Bibliography

...posium on Research in Computer Security (ESORICS 96), volume 1146 of...


[94] ITU-T (Telecommunication Standardization Sector of International Telecommunication Union). Itu-t recommendation x.509: Public


[114] K.E. Seamons, M. Winslett, and et al. Requirements for policy languages for trust negotiation. In *3rd Intl Workshop Policies for Distributed Systems and Networks (POLICY 02)*, pages 68–79, Los Alamitos, California, USA,


[136] Phil Zimmermann. Why do you need pgp?