DISSERTATION

A VECTOR MODEL OF TRUST TO REASON ABOUT TRUSTWORTHINESS OF ENTITIES FOR DEVELOPING SECURE SYSTEMS

Submitted by
Sudip Chakraborty
Computer Science Department

In partial fulfillment of the requirements
for the Degree of Doctor of Philosophy
Colorado State University
Fort Collins, Colorado
Summer 2008

COLORADO STATE UNIVERSITY

July 3, 2008

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY SUDIP CHAKRABORTY ENTITLED A VECTOR MODEL OF TRUST TO REASON ABOUT TRUSTWORTHINESS OF ENTITIES FOR DEVELOPING SECURE SYSTEMS BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

[illegible]
Outside Committee Member: Dr. John Hoxmeier

[illegible]
Committee Member: Dr. Yashwant K. Malaiya

Indrakshi Ray
Co-Adviser: Dr. Indrakshi Ray

Indrajit Ray
Adviser: Dr. Indrajit Ray

[illegible]
Department Head: Dr. L. Darrell Whitley

ABSTRACT OF DISSERTATION

A VECTOR MODEL OF TRUST TO REASON ABOUT TRUSTWORTHINESS OF ENTITIES FOR DEVELOPING SECURE SYSTEMS

Security services rely to a great extent on some notion of trust. In all security mechanisms there is an implicit notion of trustworthiness of the involved entities. Security technologies like cryptographic algorithms, digital signature, access control mechanisms provide confidentiality, integrity, authentication, and authorization thereby allow some level of 'trust' on other entities. However, these techniques provide only a restrictive (binary) notion of trust and do not suffice to express more general concept of 'trustworthiness'. For example, a digitally signed certificate does not tell whether there is any collusion between the issuer and the bearer. In fact, without a proper model and mechanism to evaluate and manage trust, it is hard to enforce trust-based security decisions. Therefore there is a need for more generic model of trust. However, even today, there is no accepted formalism for specifying and reasoning with trust. Secure systems are built under the premise that concepts like "trustworthiness" or "trusted" are well understood, without agreeing to what "trust" means, what constitutes trust, how to measure it, how to compare or compose two trusts, and how a computed trust can help to make a security decision.

To help answer such questions, this dissertation proposes a new vector model of trust. The model has several powerful features such as the ability to numerically evaluate different parameters influencing trust and to express different degrees of trust quantitatively, the ability to model the dependence of trust on time and on trust itself, and the formalization of trust comparison and trust composition operations. This work also formally defines trust context and relationships between different contexts and shows the importance of these in trust evaluation.

The primary contributions of the dissertation are: (1) A flexible quantitative model of trust based on different parameters and providing multilevel of trust. The model is extensible as the parameters are independent to each other. Addition of new parameters does not affect the other features of the model. The model can evaluate trust even when all the relevant information to do so is not available. (2) Formalism of trust context and relationship between different contexts. This formalism can help to make reasoned decisions about trust in a context when no information is available for that context. These demonstrate that the model is useful in making fine-grained security related decisions in different security contexts where other mechanisms or other trust models are not sufficient to make such decisions.

The effectiveness of the model is validated by estimating the relative trustworthiness of two security solutions (namely, cookie solution and filtering mechanism) to denial of service attacks in an e-commerce platform and comparing the outcome with the result known from practice. Trust-based decision making in different security scenarios are also discussed to show potential application of the model.

Sudip Chakraborty
Computer Science Department
Colorado State University
Fort Collins, Colorado 80523
Summer 2008

ACKNOWLEDGEMENTS

It is my privilege to convey my sincere thanks and gratitude to the people who have helped me, directly or indirectly, to complete this dissertation. Their positive influence has made this journey, for the past five years, much easier.

First, I would like to thank my advisor Dr. Indrajit Ray and co-advisor Dr. Indrakshi Ray for being my advisors and my mentors. They have introduced me to the domain of computer security, have helped me to develop my research and teaching skills, and motivated me to publish my work in the best places. Their valuable guidance and thoughtfulness has helped me to overcome the difficulties in my academic as well as personal life. Without their continuous support, encouragement and guidance, this dissertation would not have been possible. Thank you Dr. Indrajit and Dr. Indrakshi for my professional grooming and helping me achieving this degree. I would also like to thank my graduate committee members, Dr. Yashwant Malaiya from Computer Science Department and Dr. John Hoxmeier from Computer Information Systems Department for their time and valuable input on the directions of my research.

A special word of appreciation to staffs at Computer Science Department. Carol Calliham and Sharon Van Gorder have helped me to deal with official aspects in timely manner while Kim Judith helped me to resolve the financial issues. Thanks to Lisa Knebl for her guidance and help to prepare job application package. I am indebted to Dr. James Peterson and Dr. Willem Bohm for helping me to deal with last minute bureaucratic issues.

I am thankful to my friends and colleagues at Colorado State University and in Fort Collins for their friendship and support. They have always encouraged me during hard times and I have always found them when I wanted to share even a small achievement. I want to extend my sincere thanks to Dr. Sudipto Ghosh of Computer Science Department for always helping me in difficult situations in my professional life as my professor and in my personal life as my friend.

I would like to thank my family for making me who I am today. I thank my dearest wife, Poulomi Ghatak, for her love, patience, understanding and faith on me. Without her continuous inspiration this journey would have been a difficult task. Thank you Mamon for keeping my spirit high during hard times. My sincere respect and love go to my parents. Thank you Baba and Ma for giving priority to my well-being over your own interests. A special thanks to my father for being my idol, and for inculcating the significance of honesty and hard work in life, which has been a foundation for my life.

Finally, I would like to thank the sponsors of my research. Funding from Federal Aviation Administration, Air Force Research Lab and Air Force Office of Scientific Research has helped us to manage the research-related expenses as well as has helped me to get financial support as a research assistant. I am also thankful to Department of Computer Science at Colorado State University for providing me financial support as teaching assistant.

To my loving wife Poulomi, for always being there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Information technology is increasingly driven by the requirements of confidentiality, integrity, availability, and usability of systems and information resources. These are primary requirements for securing systems and we have mechanisms like certificates, cryptographic techniques, authentication and authorization mechanisms to achieve these requirements. Rasmusson and Jansson [RJ96] have named these traditional information security mechanisms as *"hard security"*. These "hard" mechanisms return binary result – either "good" or "bad". For example, if a certificate is verified then the certificate-bearer is assumed to be benign, otherwise malicious; if a program is digitally signed, it is assumed to be developed properly. However, different factors (e.g., malicious intentions of entities, collusion among entities, incompetence) make it difficult to guarantee the above assumptions. As pointed out by Abdul-Rahman and Hailes in [ARH97], "Cryptographic algorithms, for instance, cannot say if a piece of digitally signed code has been authored by competent programmers and a signed public-key certificate does not tell you if the owner is an industrial spy." This shows that the 'hard' security mechanisms are not adequate to reason about uncertainties involved in different security scenarios. Therefore, we need an alternative *"soft"* approach which can reason about certain level of uncertainty involved in different security contexts. Rasmusson and Jansson [RJ96] have used the term *"soft security"* to denote the security mechanisms using social control to reason about uncertainties. The notion of 'trust' provides such reasoning, where certain level of 'trustworthiness' of an entity indicates the level of assurance that the entity will behave according to one's expectations. This justifies the need to incorporate trust in current security services such that the security decisions are guided by reasoning about the trustworthiness of the entities. Many a times we use the term "trusted" to mean secure, or dependable, or reliable. This concept of 'trusted system' can arise in open environments like the Internet or in other types of regular networks connecting several systems together. Let us first discuss one of the ways the notion of 'trust' arises in the Internet.

The Internet is currently used as one of the primary means for exchanging information. There is a significant growth in recent years in the use of Internet for requesting and offering different services in several domains (e.g., business, education, health care etc.). This growth, consequently, makes large-scale open systems like virtual organizations, peer-to-peer networks, and other e-commerce applications more popular in these domains. However, unlike conventional systems, which rely more on direct interactions, the Internet relies on the virtual identity of entities involved in the interaction. Communication through the Internet may involve heterogeneous entities that include actual human beings, machines, applications, or processes running on a machine. That is, the communication may be between humans, human and machine, or machine and machine. Nonetheless, we assume that these passive entities like machines or applications involved in a communication, are under control of active entities like humans. Since we have assumed involvement of humans in these interactions, any entity, irrespective of its nature (i.e., active or passive), can be benevolent as well as malevolent. Therefore, both end-parties involved in an interaction, before releasing sensitive information (e.g., resources from service-provider side, credentials from user side), want to have some guarantee that the interaction would be fair. This type of guarantee would help the entities to view the system as 'trusted' according to their own perspectives and consequently, would help them to take appropriate decisions regarding dissemination of information. However, due to the anonymous nature of the Internet, it is not easy to have such guarantee. The problem aggravates as number of service providers for a specific service is large and there are potentially unbounded number of users to seek such services. We illustrate this by considering a more specific example (online travel reservation system, as shown in Figure 1.1). There are several sites (e.g., Travelocity.com, Orbitz.com,

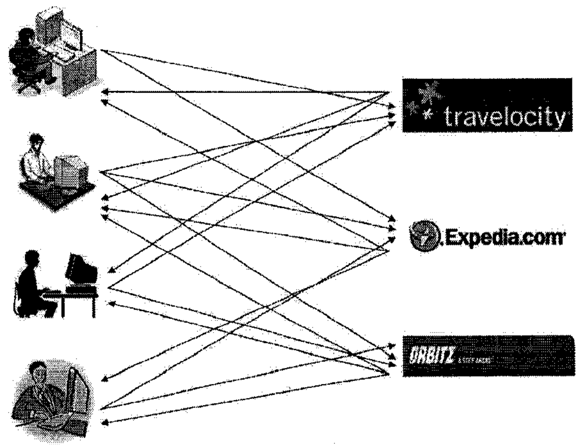


Figure 1.1: Online travel reservation & purchase system

Priceline.com, Expedia.com etc.) which provide different travel related services like hotel booking, air-ticket reservation and purchase, renting car, etc. to users. On the other side, there are millions of users who use Internet to interact with these service providers. The user's dilemma is: *Which service provider is more likely to provide the best service?* The service provider, on the contrary, tries to find the answer to the questions like *Is the user benign? To what level the user can be relied to have access to the services?* Making appropriate decisions about the next actions by the user or making intelligent authorization decisions by the service provider in this type of applications are not easy. Therefore, there is a need for mechanisms that alleviate these problems (i.e., which can help the users and the service providers to find the answers to their respective questions).

Let us consider a second example. We consider the example of a collaborative network defense system deployed within a network. A schematic diagram of the system is shown in Figure 1.2. A major component of the system is a distributed network intrusion detection (NID) module that can gather information from other network intrusion detection systems deployed elsewhere on the network that are under separate administrative controls. The NID module monitors the local network for possible intrusion scenarios and also seeks information about intrusion alerts from some of the other similar modules deployed in other parts of the network. The module then analyzes the information and advises the local administrator about the possibility of a network attack in the near future. The other NID

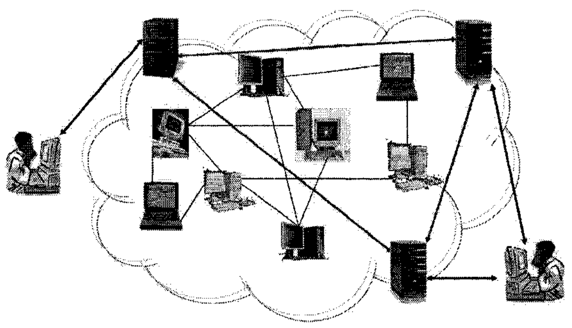


Figure 1.2: Collaborative network defense system

modules behave in a similar manner. However, problem can arise as these NID modules can be attacked and compromised. Therefore, the local administrator needs some assurance that the NID module is providing correct intrusion alerts. Also, since the local NID's ability to provide an accurate description of the activities is dependent on the behavior of the other NIDs, the administrator needs to make appropriate decision about the dependability of the composed information provided by the NID module. The local NID also needs to have certain level of reliance on the other NIDs deployed elsewhere for the information that it gathers from them. Because, it may be too naïve for the local NID to have complete faith on other NIDs. Here are several situations where this will be the case. Assume that one of the remote NIDs bears a certificate from an independent testing agency that attests to the fact that the NID application was submitted by its developer for testing and has been found to be free of malicious code and other defects. However, the certification agency may not have followed proper procedure in the certification process; the certification agency's own credentials may have been revoked but the information may have not trickled down to the end user, or the developer may have tweaked with the software after the certification. Under such circumstances although the certificate attests to the competence of the system, it does not mean much. This, again, shows that there is a need for mechanisms to assess the level of assurance about the behavior of other entities.

The above two examples show that in current information technology, it is not always sufficient to use traditional security mechanisms to make appropriate security related decisions. We need mechanisms which will allow us to reason about the uncertainties involved in such decisions. Measuring 'trustworthiness' of entities involved in the system provides a social control over the security decisions as the notion of trust can reason about certain level of uncertainties.

1.1 Notion of Trust

'Trust' is a socio-psychological concept, which is used everyday in human society. According to Merriam-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."

That is, trust gives one entity to have some kind of confidence or assurance about another entity to behave or act according to the expectation of the former entity. Thus, in our previous example of interaction between a service-provider and a user, if the service provider can establish a *trust relationship* with the user in the context of the interaction, then it has some assurance that the user will be benevolent to a certain degree. This assurance will also help the service provider to take appropriate access control decisions. Alternatively, a similar trust relationship from user to service provider will give the user some confidence about the service provider to provide him/her the service he/she is expecting. This trust will also influence the user's decision to choose a specific service provider. Therefore, there is a need for establishing such mutual trust relationships between a service-provider and a user to mitigate the problem mentioned in the above discussion. However, establishing such trust relationships are not easy, as we do not have any scope of traditional, face-to-face contact to establish trust in electronic world. Similarly, in the example of collaborative network defense system, if an administrator can impose a trust on the NID, it will help him/her to measure his/her confidence on the NID and thereby will help him/her to take appropriate decisions about the next actions. Mutual trust relationships between the cooperative NIDs address the problem discussed in the example. Nonetheless, in this example, the problem is how to establish trust between a human (administrator) and a machine (NID) or between two machines (cooperative NIDs). In both the examples, traditional security mechanisms like certificates can provide some notion of trust. However, this notion of trust is restrictive (binary) and is not suitable to make more reasoned decisions about 'trustworthiness' of the systems.

The above discussions show that we need a flexible mechanism to establish and evaluate multilevel trust in electronic world. There is a plethora of work which explores establishment and evaluation of trust in open systems like the Internet and systems involving heterogeneous entities. Nonetheless, these works vary from each other in terms of semantics, representation, evaluation, and management of trust and make it difficult to use the notion of trust in designing trustworthy systems. The next section summarizes the problem and presents an overview of our motivation in this research.

1.2 Overview of Problem Description and Motivation

It is clear from the above discussions that the notion of trust is cornerstone of information security. Intuitively, trust allows one to qualify one's level of assurance on the perceived or measured security of a system involving different entities. Secure systems are built under the premise that concepts like "trustworthiness" or "trusted" are well understood, without agreeing to what "trust" means, how to measure it, and how to compare or compose two trust values. The problem arising from this lack of agreement is two fold. First, it is difficult to make reasoned security related decisions using one's level of assurance about trustworthiness of a system without having precise methods of measuring, comparing or composing trust. Defining precise methods to evaluate trust is important for deployment of trust in designing secure systems. Most of the existing research is not clear about these methods. Second, it is difficult for systems using different semantics and representations of trust to function in a cooperative manner. For example, a system may view entities' past experience as trust and another may use entities' reputation to build trust. These two systems work fine individually. Nevertheless, it is hard to make these two systems function in a cooperative manner to measure trustworthiness of an entity in some application. Also it is difficult to reason about the relative performance of the two systems in measuring trustworthiness as the trust levels indicated by the two systems are based on two different notions. Analyzing the existing research, we observe the following:

1. Most of the existing trust models use parameters, based on specific meaning, to evaluate trust, thereby making them suitable for specific environment. Also, the models are not extensible to accommodate other parameters. Hence, there is a need for a flexible trust model which is generic enough to accommodate different parameters and is applicable in different environments.

2. Many of the trust models view trust as a binary notion – 'complete trust' or 'no trust'. Though this type of representation is intuitive and may be suitable for certain purpose, it is definitely not suitable for making fine-grained decision about trustworthiness. Therefore, we need to move ahead from this binary paradigm to a better and finer representation of trust.

3. Most of the trust models do not clearly specify how a trust is measured. Specifying precise methods and algorithms of measuring and updating trust is important to implement trust in real systems.

4. Most of the proposed models ignore to specify the factors that influence trust. Identifying definite factors influencing trust is needed to define a good trust model. Otherwise, it would be difficult to judge what might affect the trustworthiness of a system build around the trust model.

5. The existing trust models do not propose any precise method or algorithm to compare two trust levels. Comparison is left on the intuition of the user that is, it is assumed that 'very trustworthy' is better than 'trustworthy' or trust level 0.8 is better than trust level 0.6. Composition of two trust is ignored. Therefore, precise methods of comparison and composition of trust levels are need to be developed.

6. Most of the existing models do not discuss how the models can reason about trustworthiness in different security contexts. There is no formalism of trust context.

7. Last but not the least, existing mechanisms are not suitable for reason about trustworthiness when all relevant information are not available or when the available information is ambiguous.

1.3 Objective and Significance

Motivated by the above issues, we propose a trust model that can be used to make more reasoned decision regarding trustworthiness of systems. For this purpose, we believe, a quantitative approach is more suitable. Quantifying trust through numeric values helps to define fine-grained trust levels alleviating finer trust-based decisions. Also, mathematical operations on levels of trust can be defined that allow proper comparison of levels from different domains and combine them. Therefore, goals in this dissertation research can be summarized as:

1. To propose a new model of trust that will reason about trustworthiness of an entity in a more reasoned manner. In particular, the model should be capable of

    (a) providing an answer to the problem arising from the semantic mismatch among various trust models. That is, the model should be generic enough to incorporate different interpretations of trust such that systems having different meanings of trust can function in a cooperative manner. In particular, the model should incorporate different parameters arising from different interpretations and should be easily extensible to accommodate new parameters.

    (b) evaluating different factors that influence trust as well as measuring the overall trust using the evaluated factors. The evaluation mechanism should be flexible enough to accommodate user's specific preferences regarding trust evaluation.

    (c) comparing trust imposed on different entities. The model should have a method to compare two trust relationship established in same context as well as in different but similar contexts.

    (d) defining mechanisms to compose two or more trust relationships. The mechanisms should consider different scenarios of compositions. For example, when a particular truster wants to compose trust relationships with two or more different trustees. Another scenario is when a group of entities forms a coalition and wants to combine their individual trust relationship with a particular trustee.

    (e) capturing importance of the underlying context in trust evaluation. In particular, the model should be able to reason about trustworthiness even when the information is incomplete or ambiguous in a context.

2. Another goal is to check the validity of the aforementioned model by comparing the result using it, in a security context, to the result known from practice.

3. Last but not the least, this dissertation investigates the applicability of the model as a decision aid in different security scenarios. For example, how the model can be used to make access control decisions in open systems or, how the model helps to find a secure path for sending packets in an ad hoc network etc.

1.4 Dissertation Organization

The rest of the dissertation is organized as follows:

- Chapter 2 presents the literature review. It includes, in Section 2.1, a list of definitions of trust that have been proposed by researchers from several disciplines followed by Section 2.2, presenting different models proposed by scholars. We focus on the models proposed within information science domain and categorize the models according to the interpretation of trust used in these models.

- Chapter 3 presents the details of the proposition. In this chapter we discuss the definition of trust, representation of trust, formalism of trust parameters and the methods to evaluate them, and dynamics of trust. In Sections 3.4 and 3.5, we present the approach proposed for addressing trust comparison and trust composition.

- Chapter 4 discusses how we can reason about trust in different contexts. The chapter includes formal definition of context and a context ontology describing relationships between contexts, context graph, and 'closeness' (or, similarity) among different contexts.

- Chapter 5 proofs the effectiveness of the model by evaluating relative trustworthiness of two security solutions of denial of service attacks in an e-commerce system. Result found corroborates the fact known from practice.

- Chapter 6 discusses application of the trust model as decision-aid in three different security contexts. Section 6.1 shows applicability of the model to make access control decisions in open systems. Section 6.2 presents a trust-based routing scheme in pervasive computing environment. Section 6.3 discusses how the model can be used to allow finer control over user privacy on the Internet.

- Finally, Chapter 7 concludes this dissertation. Sections 7.1 summarizes the contributions and importance of this dissertation and Section 7.2 concludes the dissertation with a pointer to the future directions.

Chapter 2

Related Work

Researchers from several disciplines have given significant attention to the notion of trust. Sociologists (Luhmann [Luh79], Meeker [Mee84], Baier [Bai86], Shapiro [Sha87], Good [Goo00]) and psychologists (Deutsch [Deu60], [Deu73], Swinth [Swi67], Rotter [Rot67], Mathews et al.[MS79], Sato [Sat88]) have been doing research on trust for quite a long time. Scholars from economics (Dasgupta [Das88], Humphrey and Schmitz [HS96]), management (Zand [Zan72], Driscoll [Dri78], Zaheer et al.[ZMP98], Adler [Adl05], Lewicki et al.[LTG06]), and marketing research (Andaleeb [And92], Brashear et al. [BBBB03]) also have explored influence, interpretation, and use of trust in these areas. From mid to late 90's, theorists from computer and information science have been taking interest in trust research. During recent years, this interest has grown several times and as a result we have an abundant amount of works on trust in computer science. These works can be broadly classified into three distinct areas (i) trust models, (ii) trust management and negotiation and (iii) application of trust concepts. In this dissertation, we are primarily interested in trust models. Consequently we focus our discussion on this aspect of the literature. But before that, let us present the diversity in the definitions of trust.

2.1 Different Definitions of Trust

Following is a list of definitions among several that can be found in literature:

In [Zan72], Zand defines trust as

``the willingness to be vulnerable based on positive expectations about the `action of other.''`

Mayer et al. [MDS95] defines trust in a similar fashion as

`''the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other.''`

According to Morton Deutsch [Deu73], trust is

`''the confidence that one will find what is desired from another, rather that what is feared.''`

In his book "Logic and Limits of Trust" [Bar83], Bernard Barber expresses trust as

`''expectation of the persistence and fulfillment of the natural and social orders.''`

Barber also emphasizes that there are two meanings of trust. First is *"trust as the expectation of technically competent role performance"* and the second is *"trust as expectation of fiduciary obligation and responsibility, that is, the expectation that some others in our social relationships have moral obligations and responsibility to demonstrate a special concern for other's interest above their own."*

Lewis and Weigert [LW85] posit that trust can be represented as

`''observations that indicate that members of a system act according to and are secure in the expected futures constituted by the presence of each other for their symbolic representations.''`

Barney and Hansen [BH94] view trust as

`''the mutual confidence that no party to an exchange will exploit another's vulnerability.''`

Curral and Judge [CJ95] affirm that trust is `‘‘an individual’s reliance on another party under conditions of dependence and risk.’’`

In [KC98], Kini and Choobineh express trust, using concepts like belief and opinion, as

`‘‘a belief that is influenced by the individual’s opinion about certain critical system features.’’`

In a technical report of The European Commission Joint Research Center [JM99], Jones and Morris defines trust as

`‘‘the property of a business relationship, such that reliance can be placed on the business partners and the business transactions developed with them.’’`

In their survey on trust [GS00], Grandison and Sloman define trust as

`‘‘the firm belief in the competence of an entity to act dependably, reliably and securely within a specific context.’’`

Mui et al. propose a probabilistic view of trust in [MMH02]. They define trust as

`‘‘a subjective expectation an agent has about another’s future behavior based on the history of their encounters.’’`

We end this list with the definition given by Avižienis et al. [ALRL04]. They define trust to be *accepted dependence* where

`‘‘... the dependence of system A on system B, [thus] represents the extent to which system A’s dependability is (or would be) affected by that of system B.’’`

From the above list of definitions it is evident that view of trust is widely varying among scholars from different fields. This type of varying semantics have been used in different trust models also. We discuss some of the trust models proposed within computer and information science domain.

2.2 Different Models of Trust

Researchers have proposed several models of trust. In these models they have used their own interpretation and representation of trust. Methodologies to express and evaluate trust also differ from model to model.

2.2.1 Trust Models using Forms of Logic

A number of logic-based formalisms of trust have been proposed by researchers. Almost all of these view trust as a binary relation. Forms of first order logic [BAN90, JSS97, JF00], and modal logic or its modification [Ran88] have been variously used to model trust in these cases. Simple relational formulas of the form $T_{a,b}$ (stating a trusts b) are used to model trust between two entities. Each formalism extends this primitive construct to include features such as temporal constraints and predicate arguments. Given these primitives and the traditional conjunction, disjunction, negation and implication operators, these logical frameworks express trust rules in their language and reason about these properties.

Rangan [Ran88] proposes a model using modal logic approach where the modal operator used is 'belief' of an agent. He views trust as "a proper axiom added to the logic of belief i.e., any WFF that is assumed to be valid in addition to the axioms in the logic." The simple trust statement, in this model, is of the form $B_i f$ meaning agent i believes proposition f, where f is a well-formed formula. In his model a distributed system is a collection of agents communicating with each other by message passing. The state of an agent is the agent's message history. The state of the systems is the state of all agents. Rangan uses modal logic to define a set of properties of the trust statements like transitivity etc. These constructs are then used to specify system and analyze them with respect to the property of interest. The general approach to obtain trust in a system is done by encoding the required security properties of the system as well-formed formulas in the language of the belief logic. He asserts that "the required trusts are such that in the theory obtained by adding the trusts to the belief logic, the security properties are theorems." Though the formalism of trust, in this model, is based on properties of mathematical logic, it also makes the model very abstract. One of the difficulties with the model is it requires translation of security properties of a system as well-formed formula in logic to obtain trust. The model also does not provide method to measure trust levels, or methods to compare and/or compose trust.

Burrows et al. propose a logic-based formalism of trust (called BAN-logic) [BAN90] that can be used for verification of correctness of security protocols, especially authentication protocol. They follow similar approach as Rangan and translate steps of a security protocol as logical formulas. These are then manipulated using first-order logical rules like "P believes X", "P sees X", "P controls X" etc. where P is a principal and X is a statement. The model, like Rangan's model, is very abstract and too complex to be used in reality. The logical rules specified in the work are rather intuitive and there is no proof that they are true. Another major problem of the model is that it can verify the trustworthiness of a security protocol only when it is performed according to the specification. The BAN-logic fails to verify the correctness of a security protocol if it is executed in an unconventional manner.

Jones and Firozabadi [JF00] identify "that there are at least two different types of trust: trust in an agent's ability, and trust in the reliability of the information transmitted by an agent." In their work, they address the issue of reliability of an agent's transmission. They use a variant of modal logic, which is termed as "deontic logic" to model various trust scenarios like "b's belief that a sees to it that m". They also use their language to model the concepts of deception, where an agent a tries to make another agent b believe a proposition which a himself does not believe. They also use the same logical constructs to model the concept of an entity's trust in another entity and correctly assert that this trust is not always transitive. However, the model suffers from the same type of problems as in Rangan's or Burrows et al.'s model. Use of variant of mathematical logic makes the concept of trust very abstract, thereby making it difficult to reason about trustworthiness.

2.2.2 Trust Models using 'Direct' and 'Indirect' Information to Evaluate Trust

Several existing trust models use *direct* as well as *indirect* information to derive trust. Direct information is obtained from the experience or interactions between agents or entities.

Indirect information is typically collected from third parties, that is, these information are some type of feedback provided by others. Though the underlying semantics of these information are same ('experience' for direct information and 'recommendation' for indirect information), the models vary in naming, collecting, and managing these information.

2.2.2.1 Yahalom et al.'s Work

Yahalom et al. [YKB93, YKB94] propose a formal model for deriving new trust relationships from existing ones. In [YKB93] the authors propose a formal model for expressing trust relations in authentication protocols, together with an algorithm for deriving trust relations from recommendations. The authors propose seven different classes of trust, namely identifying entities, quality random key generation, keeping secrets, not interfering, clock-synchronization, performing correctly algorithmic steps and providing recommendations. Being trusted for a particular class means that an entity can be trusted to perform a specific task. Each of these classes of trust can have two types of trust: 'direct trust' and 'recommendation trust'. Direct trust is when an entity trusts the other entity without including an intermediary in the trust relationship. Recommendation trust involves trusting an entity based on a recommendation of a third party. There can be multiple trust relationships between the same pair of entities. Trust with respect to one of the seven classes is independent of trust with respect to another class of trust. In [YKB94] rules and algorithms for obtaining public keys based on trust relationships are developed.

These works correctly identify the importance of direct as well as indirect information for evaluating trust. The classes of trust captures the notion of 'trust context' where trust in one context may differ from trust in another context. Consequently, they allow multiple trust relationships between same pair of entities. However, neither of these works defines trust itself. They use constructs like "$A\ trusts_C\ B$" to denote that entity A trusts entity B in the specific trust class C. But they do not specify how the trust is computed.

2.2.2.2 Beth et. al.'s Work

Beth et al. [BBK94] extend the ideas presented by Yahalom et al. to include relative trust. The work presents a method for extracting trust values based on experiences from the real world and also a method for deriving new trust values from existing ones within a network of trust relationships. Such trust values can be used in models such as the ones by Yahalom et al. The method proposed is statistical in nature. It is based on the assumption that all trusted entities have a consistent and predictable behavior. To model degrees of trust, the notion of "numbers of positive or negative experiences" is used. The authors posit that for calculation of direct trust no negative experiences are accepted if an entity is to be trusted at all. For recommended trust, on the other hand, negative experiences are acceptable.

This model improves the model proposed by Yahalom et al. by providing methods to evaluate trust using 'number of experiences'. Nonetheless, the model suffers from the assumption that negative experiences are ignored in calculating direct trust. If that is the case then according to the trust expression proposed by the authors a direct trust level can only increase. A consequence of this is that a long history of experience implies either almost absolute trust or none at all. On the other hand, negative experiences are considered for recommendation trust. Thus it is possible to have a relatively low degree of trust after a long history. However, as shown by Jøsang [Jøs97], the expression for deriving recommended trust is unsuitable in environments where the trustworthiness of entities is likely to change. Further, the expression for deriving new direct trust from direct and recommended trust appears to be counter-intuitive and can lead to a case similar to the following - "If you tell me that you trust NN by 100% and I only trust you by 1% to recommend me somebody, then I also trust NN by 100%" [Jøs97].

2.2.3 Models Treating 'Trust' and 'Reputation' Synonymously

There have been several works which treat 'trust' and 'reputation' synonymously. These works posit that an entity's 'reputation' indicates its 'trustworthiness'. In most of these models reputation is computed based on experience, or recommendation, or some combination of both. Many of these models are proposed in decentralized systems like peer-to-peer (P2P) environment.

2.2.3.1 Abdul-Rahman and Hailes Model

In [ARH97], Abdul-Rahman and Hailes propose a trust model for decentralized system. Their proposal generalized the notion of trust by using *trust categories* and *trust values.* Trust category captures the aspect in which the trust is applicable and trust value expresses the extent of trust. The model, like the previous models, defines two types of trust relationships between two entities. The *direct trust* relationship specified between A and B implies A trusts B in some trust aspect. A *recommendation trust* implies A trusts B to provide recommendation about other entities. For each of these two types of trust relationship, the model defines a discrete set of values to represent degree of trustworthiness. A tuple containing entity id, trust category, and trust value is defined as a *reputation* and communicated trust information containing reputation is called *recommendation.* The model defines a structured form of recommendation containing fields like requester id, request id, recommender path, target id, trust category, trust value, and expiry. A protocol has been designed to request and send such recommendations. The protocol also has provision of refreshing or revoking previous recommendations. A recommender can provide recommendation about a recommender. If an entity requiring a service and does not know whom to ask, it can send recommendation requests to all his trusted recommenders. The recommendation trust along a recommendation path is calculated using product of requester's trust on recommenders and the recommendation value. The final recommendation value is the average of trust values of all such recommendation paths.

2.2.3.2 Aberer and Despotovic Framework

Aberer et. al. [AD01] propose a decentralized reputation management framework for P2P systems. The trust is assessed by computing a peer's reputation from its interactions with other peers. They assert that the reputation is an assessment of the probability that the peer will cheat. The global trust data consists of reports about transactions made between peers. A particular peer p has a behavioral data $B(p)(\subset B)$ which contains reports made about p as well as reports made by p. In this scheme, A peer q evaluating trustworthiness of p does not have access to these global data B and $B(p)$. Consequently, q has to rely on his own reports about p and reports of a set of 'witnesses' who have reports about p. The scheme assumes that a peer is by default trustworthy, However, in case of a malicious behavior of p, the peer q files a complaint $c(q,p)$ and these complaints are the only behavioral data. Reputation $T(p)$ of a peer p is calculated as the product of all complaints lodged by p and all complaints reported about p. The trust data that is, complaints is stored in a decentralized manner where each peer is responsible for a part of the total global data. If a peer does not have required data for a query it forwards it to another peer according to its routing table. To answer the problem of false data provided by malicious peer, the scheme assumes that a peer is malicious with a certain probability π which has a maximum value π_{max} and the number of replicas r satisfies π^r_{max} is less than an acceptable limit. So if same data about a peer is received for a sufficient number of replicas then there is no further check, otherwise the checking is continued.

The work has some advantages to manage trust in P2P systems. The distributed nature of the data storage and sharing of that data among peers to compute trust make the approach suitable for P2P systems. Use of a decentralized data storage method ensures replication of data which in turn ensures availability of data. However, the model has few limitations. The model assumes a binary notion of trust, thereby making it not suitable for fine-granular decision about trustworthiness. It also works under the premise that a peer is by default trustworthy which is not a reasonable assumption in many different security scenarios where presence of malicious peers cannot be ignored. The trust metric simply summarizes the complaints a peer receives and files. This is very limited way of computing trust and sensitive to the skewed distribution of the community and misbehavior of peers. If majority of peers are malicious then it is hard to decide about trust just by the number of complaints. Importance of trust context in a trust evaluation has been identified, but not discussed in details. The authors also point out two problems regarding third party information: one is of misleading reports from malicious witnesses and the other is unavailability of witnesses as and when required. The authors assert that in a transaction between a good peer p and a malicious peer q both will file a complaint against each other. To a third peer r both are same. Now if r finds that in a transaction between q and s both have a complaint against each other, then r can be sure with a probability that q is a bad peer though no probabilistic metric has been proposed. This way of inferring about q is not a very reasonable to do. Because, it may be the case that actually q is a good peer and is a victim of bad peers r and s. From all complaints against a peer, there is no way to measure how many are actual complaint and how many are false.

2.2.3.3 PeerTrust Framework

Xiong and Liu [XL03, XL04] present a reputation-based trust supporting framework. It includes an adaptive trust model which quantifies the trustworthiness of a peer based on transaction-based feedback that the peer receives from other peers. The trust model has five parameters: (1) *Feedback in terms of amount of satisfaction* – the feedbacks are quantitative measure of amount of satisfaction that other peers in the community have on the given peer based on their past experiences. (2) *Number of transactions* – it is a scope factor for comparing the feedbacks. The related metric is the ratio of the total amount of satisfaction a peer receives over the total number of transactions that peer has i.e., the average amount of satisfaction the peer receives for each transaction. (3) *Credibility of feedback* – it is the trustworthiness of the feedback provider. This metric scales the feedbacks according to the trust level of the peers providing the feedbacks. The effect of malicious peers providing false feedbacks get diminished by the metric. (4) *Transaction context factor* – this metric weights feedbacks according to the importance of the transaction. (5) *Community context factor* – this metric addresses some community-specific issues like adding a reward for peers who submit feedback. The general trust metric is a weighted sum of two parts with weights α and β. One part is a weighted average of amount of satisfaction, weighted by credibility factor and transaction context factor. The other part is the community-context factor. However there is no guideline on how to choose the weights.

The scheme uses feedback for recent transactions to compute the trust value. The scheme asserts that a peer's reputation is based on cumulative average of his lifetime ratings. Therefore, a peer has diminishing incentive for being benign after building a good reputation, as incremental rating does not significantly change his 'good reputation'. This gives rise to the problem of oscillating behavior of malicious peers. They can fool the system by building a good reputation and behaving maliciously in such a way that their reputation does not drop significantly. Discounting older feedbacks forces a peer to behave consistently to maintain a good reputation. PeerTrust proposes a window-based algorithm to address the above issue. Two trust values T and T_s are computed using feedbacks obtained within two recent time-windows w and w_s respectively, where w_s is the recent subset of w. If $T - T_s$ is less than a threshold then T_s is accepted otherwise T is accepted. T_s implies that the trustee peer is dropping his performance recently. This adaptive time-window based algorithm ensures that reputation can not be quickly increased by a few good transactions, but will be quickly lowered if the peer starts cheating.

PeerTrust addresses the issue of unauthorized manipulation of data in storage as well as during transmission respectively with data replication and a PKI-based scheme. A peer searching for a trust data sends request to more than one peer and combines the data using a majority voting. Thus manipulation made by bad peers gets detected. To prevent anomalies during transmission it uses a PKI-based scheme. It assumes each peer has a public-private key pair and includes the public key in each request. The responder encrypts response with this public key and digitally signs this encrypted message with his private key. This ensures confidentiality and integrity of the data. No peer other than the requester can get hold of the response as it is encrypted with his public key. Also no peer can selectively discard or modify the response as it is digitally signed by the responder.

In PeerTrust, the underlying trust metric considers parameters that are relevant to compute trust. However the trust metric ignores the peer attributes like quality of offered resource, file size, uploading speed etc. to measure its reputation. These properties can be important factors to decide about trustworthiness of a peer. One of the advantages of the scheme is it uses a decentralized scheme to store trust data over the network. This reduces storage requirement for each peer where each contains a portion of the global trust data. However, in such data location scheme peers may misbehave by providing false data in his response to a request. Though data replication is proposed to combat such situations, it does not prevent a malicious peer to produce a false rating. This can be worst when a majority voting gives a wrong information which can happen if the majority of the voters are part of a collusion. Effect of false rating is reduced to some extent by the credibility factor, but can not completely remove the problem. Another good point about PeerTrust is it tries to address the dynamic or oscillating behaviors of entities. Nonetheless, the use of two time windows for recent transactions is redundant. Because there is no end to narrowing the window unless one wants to call the last transaction as 'recent'. Therefore, a peer can start with a window suitably chosen according to his comfort level and policy. This can reduce the overhead of computing trust twice and comparing them.

2.2.3.4 Selçuk et. al.'s Work

A reputation-based distributed trust architecture to identify malicious peers and to prevent the spreading of malicious contents has been proposed in [SUP04]. The proposed protocol distinguishes a malicious response from a good one by using the reputation of the peer providing the response. In the scheme each peer maintains a trust vector for every other peer it has interacted with in the past. Each vector is a constant-length binary vector in which each bit-position represents the outcome of a transaction; 1 for good and 0 for malicious. New transaction is written in the most significant bit by shifting the vector to right. A vector with m 1's is read as an m-bit integer and divided by 2^m to get a scalar value corresponding to the vector. By this process each vector has a numeric value within $[0, 1]$. Complement of the vector gives the distrust rating. If the query for a resource f receives responses from G peers then a trust coefficient for f is calculated as the average of the trust value of top θ (pre-selected as threshold for f) most trusted peers among those G peers. If θ_T peers are not available then a query is issued to choose remaining peers. The trust of the responder is updated on the basis of the quality of the downloaded f from it. A trust query is similar to a resource query; the difference is that the subject of the query in the former is a peer about whom trust information is inquired. Responses include responders' rating about the queried peer. Each response is weighted by the credibility factor of the responder. The credibility factor is evaluated in the same way as the trust vector, only the underlying context is 'outcome of a judgment'. A similar threshold θ_C is considered and the trust is computed as the average of the qualified responses weighted by the credibility rating. Updating the credibility rating is done if its rating about the queried peer matches with the result of the download from the queried peer.

The model identifies trust and distrust separately. However, distrust is calculated from the same vector trust has been calculated. Hence, distrust does not really add any new trust information thereby does not strengthen the protocol any further. The trust is based on results of a fixed number of recent interactions. Though the outcome of an interaction is considered to have only binary values, this way of calculating trust requires a peer to perform consistently to maintain a good reputation. In the model, feedbacks about a queried peer is weighted by the credibility rating of the feedback provider. The credibility rating is based on the outcome of a judgment by the responder. This captures the concept of trust context which shows that a peer trusted as a resource provider may not be trusted at the same level as a recommender. This separation also reduces effect of collusion by preventing a malicious peer to manage a good trust and then use that trust to recommend another malicious peer who will do the harm. The basic protocol has some extensions to provide authenticity using digital signature and hashing, protection to DoS attacks by a kind of challenge-response scheme, and prevention of false file downloads using hashing during download.

2.2.3.5 Anomaly Detection Technique-based Model

In [SBWS05] Stakhanova et. al present a reputation-based trust framework for P2P networks where trust is quantified using anomaly detection technique. The anomaly detection technique identifies unusual behavior in the established normal behavior of a peer and indicates instability of a peer. The anomaly detection is done at predefined checkpoints during a session and at the end of each session. The trust is based on four actions: resource search, resource upload, resource download, and traffic extensiveness, each having binary outcome. The trust metric is the percentage of bad actions. The truster peer sets two trust thresholds x_1 and x_2; peer having trust below x_1 are fully trusted, peers having trust above x_2 are totally distrusted and peers with trust between x_1 and x_2 are partially trusted. A peer profile is the collection of information that expresses the usual behavior of the peer. Six features to characterize a peers behavior have been identified. They are (i) connection time, (ii) connection duration, (iii) number of search request, (iv) number of file downloads, (v) number of file uploads, and (vi) number of bytes uploaded by a peer. The above information, called session data, is collected for each peer throughout its online session. The behavioral profile is built using a training session data set and then deviations are detected in new data set. The anomaly detection algorithm detects an anomaly and estimates amount of change in the trust score. The degree of anomaly is based on mean and standard deviation of the session data. Two thresholds, similar to trust threshold, are set and depending on the deviation and the thresholds, bad action value is incremented by some predefined amount.

The anomaly detection technique is particularly helpful in identifying potential malicious activity by a peer. However a deviation in normal behavior does not distinguish between a malicious activity and a deviation due to failure of connection/hardware/software etc. It is also very hard to measure deviation for a peer whose natural behavior is oscillating in nature. This can be due to malicious nature or limited capability of the peer. There is an assumption that peer profile are collected and available in a trustworthy manner. There is no basis for such assumption. Another problem is for partially trusted peers, a part of traffic is accepted from them. However, there is no mechanism to choose such part. Among the behavioral features connection time has little significance as connection duration is more important to characterize a behavioral pattern. The trust metric considers only bad actions and does not allow any incentive for performing good actions. For newcomers in the network or peers having small number of interactions the profile will contain no data or insufficient data to measure the deviation and consequently the trust. Therefore, the approach is very limited and not sufficient to reason about trustworthiness in a flexible manner.

2.2.4 Trust Models using Direct 'Experience' to Evaluate Trust

In some trust models, an entity's direct experience (in terms of number of events or interactions) have been used to reason about trustworthiness of other entities.

Jonker & Treur [JT99] consider trust from a software agent's perspective, "that is, trust within software agents regarding the reliability of objects and tools, their own work, the behavior of others, and in the evolution of their environment (events and effects of actions performed by the agent)." They use an agent's experiences (i.e., evaluated events) to formalize trust evolution or trust updating. They start with an initial trust, which can have four states like *initial unconditional trust, initial conditional trust, initial unconditional distrust*, and *initial conditional distrust.* Then six different ways of trust dynamics (e.g., blindly positive, slow-positive-fast-negative, balanced slow etc.) have been defined. The authors propose two ways to model these dynamics. One is by defining a "trust evolution function" – a mathematical function that formalizes the dependency of trust on past sequence of experience (events); the other is using the current trust and current experience in a mathematical function, called "trust update function" to formalize the next trust. Each of these experiences or events are categorized into two classes – positive and negative. The model also has a property called "degree of memory based on window n back" which allows to forget all past experiences except last n events.

This is a good model of trust where the authors propose multi-level of trust based on truster's experience. The model correctly identifies the relative importance of past and current experiences in trust evaluation. The model propose trust dynamics and different ways to update trust. The formulas used are formalized using mathematical functions. However, the model uses very restrictive notion of trust. It considers only the experience of a truster to evaluate trust. This restricts the model to use 'blind trust' or 'initial unconditional trust' for an agent in the absence of any event. This can create problem if the agent is malicious. Also the model is not extensible and not applicable in scenarios where there is no scope of obtaining direct experience initially to establish trust.

2.2.4.1 EigenTrust framework

The EigenTrust algorithm, proposed by Kamvar et. al in [KSGM03], proposes to decrease the number of downloads of inauthentic files in a P2P file-sharing network. The algorithm uses number of 'satisfactory' and 'unsatisfactory' interactions between two peers to compute "local trust" of a peer. The network assigns a unique global trust value, evaluated using the local trust values, to each peer on the basis of their 'upload' history. Peers use these global trust values to choose the peers from whom they download files. The authors identify five issues that should be addressed while designing any P2P reputation system. (1) *Self-policing* – the "shared ethics" of peers are defined and enforced by the peers themselves.(2) *Anonymity* – a peer's reputation is associated with an indirect identifier of the peer. (3) *No profit to newcomers* – there is no advantage being a newcomer in the peer network. Reputation must be earned by constant good behavior. (4) *Minimal overhead* – "The system should have minimal overhead in terms of computation, storage, and message complexity". (5) *Robust to collusion* – the system should have mechanism to prevent a group of malicious peers from subverting the system.

The authors develop the algorithm following these design criteria. However, there are some advantages and disadvantages of these. For example, no profit to newcomers, enforcing to earn reputation by constant good behavior, can prevent a malicious peer to get away after any malicious act by changing his identification and posing as a newcomer. On the other hand, in an hostile environment, all the peers may not follow the self-policing rule. In fact, malicious peers may not follow the ethics that an inauthentic file should not be injected into the network intentionally.

EigenTrust algorithm aggregates *local trust values* of peers to compute a peer's global trust value. A peer i computes his local trust s_{ij} about a peer j by summing up the ratings of each transaction that peer i has with peer j. A transaction can have a positive rating (value = +1) or a negative rating (value = -1). This local trust values are then *normalized* to neutralize the effect of false rating given by malicious peers. The normalized trust of i on j is given by $c_{ij} = \frac{max(s_{ij},\, 0)}{\sum_j max(s_{ij},\, 0)}$. To aggregate the normalized trust values of a peer

k, a peer i takes his friends' (js) opinions about k and weight the opinions by the trust i places on js. Therefor, i's trust on k is, $t_{ik} = \sum_j c_{ij}c_{jk}$ which is written in matrix notation as $\vec{t_i} = C^T\vec{c_i}$ where $\vec{t_i}$ = vector of values t_{ik}s, $C = [c_{ij}]$ and $\vec{c_i}$ = normalized local trust vector of i. i can ask all friends of j about k to get a wider opinion about k. In this way, after n (large) iterations i will have a global view about k from the vector $\vec{t} = (C^T)^n\vec{c_i}$. For sufficiently large n, the vector t will be same for every peer. It will be the left eigen vector of C. Therefore $\vec{t}$ will give the global trust of all peers in the system. The EigenTrust algorithm assumes that there are some peers in the network that are pretrusted. It assigns an a priori notion of trust p to all peers; pretrusted peers have the value $\frac{1}{P}$ where P is the number of pretrusted peers and all other peers will have an a priori trust 0. For an inactive peer i (who does not interact with anyone) the algorithm assigns a priori trust of other peers j to c_{ij}. The algorithm addresses the problem of collusion by having each peer to place at least some trust in the pretrusted peers. The trust value of a peer i is computed by *scoremanagers* of i. A peer is assigned to the role of a score managers using distributed hash tables (DHT). The score manager also maintains the opinions of the peers covered by it. This ensures anonymity as a requester does not know from which peer it actually getting the opinion. Randomize nature of allocation of peers in DHT space ensures that a peer can not choose its own location to get the advantage of manipulating its own trust. Also there are several score managers to compute trust for a peer which provides greater availability of the trust scores.

EigenTrust algorithm provides a distributed way of evaluating trust in a P2P framework. It has a strong mathematical construct to measure trust – peers' global trust are obtained from the left eigen vector of the matrix where all the local trusts are stored. However, there are some problem with the scheme. The concept of pretrusted peers is essential to the scheme, but it gives rise to number of problems. First, it is very hard to ensure existence of a pretrusted peer in all the cases. At some instance, there may not be any pretrusted peer available in the network. In that case the algorithm fails. Also the choice of the pretrusted peer is important because no pretrusted peer should be a part of a collusion. There is no way to ensure it. Even if pretrusted peers exist there is no guarantee that they will always remain benign. A pretrusted peer may turn into a malicious peer. The other problem is that the normalized local trust value does not consider negative experiences. The value considers maximum of satisfaction score and 0. This makes no difference between a peer's satisfaction score about a peer with whom it does not have any interaction and about a peer with whom it has all bad experiences.

2.2.5 Probability-based Models

Cohen et al. [CPF97] propose an alternative, more differentiated conception of trust. They call it Argument-based Probabilistic Trust model (APT). It includes the more enduring concept as a special case, but emphasizes instead the specific conditions under which an aid will or will not perform well. According to their approach, the problem of decision aid acceptance is neither under-trust nor over-trust, but inappropriate trust. The authors define this notion as a failure to understand or properly evaluate the conditions affecting good and bad aid performance. They propose a simple framework for deriving benchmark models of verification performance, in situations where previously learned or explicitly identified patterns may be insufficient to guide decisions about user-aid interaction. The most important use of APT is to chart how trust varies, from one user to another, from one decision aid to another, from one situation to another, and across phases of decision aid use. The authors presented a benchmark model for deciding when to accept, reject, or verify a decision aid's conclusion. They also point out that trust evolves as the user moves through the various phases of a particular mission or task. This is due to active decisions by the user; that trust is considered as a product of the interaction between the decision aid and the user.

2.2.6 Manchala's Work

A business-oriented approach to trust has been taken by Daniel Manchala in [Man00]. In this work, he describes a model of trust for e-commerce and proposes metrics to evaluate risk in a transaction using trust. The trust variables that he considers are transaction cost, transaction history, indemnity, spending pattern and system usage. Manchala suggests two parameters – time and location, to fine-tune these trust variables. The variables are measured quantitatively and depending on the evaluated trust, different trust actions like verification, assessing security level, authorization are taken for the transaction. However, the model does not clearly specify how to evaluate the variables quantitatively or how to compose these values to obtain a value for trust. Also the model is restricted to verify transactions in e-commerce only.

2.2.7 Belief-based Models

2.2.7.1 Jøsang's Model

In [Jøs98, Jøs99] and in several subsequent papers over the last 10 years, Audun Jøsang has developed a model for trust based on a general model for expressing beliefs about the truth of statements. In this model, trust is an opinion and an opinion is a representation of a belief. An opinion is modeled as a triplet $< b, d, u > \in \{b, d, u\}$, where b is a measure of one's belief in a proposition, d is a measure of one's disbelief in the proposition and u is a measure of the uncertainty. The tuple is represented as a point in unit triangle (called "opinion space") and the three components are connected by the relation $b + d + u = 1$.

One of the main advantages of the model is it can assign certain degree of belief, disbelief, and uncertainty about a proposition simultaneously. This also helps us to formalize the trust levels like 'totally trust', 'total distrust' or 'totally uncertain'. However, for any other state of trust, the model does not provide a single value of trust. Trust is always represented as opinion tuple. Another contribution of the model is it provides mechanism to compare and compose two opinions. These features were added in the later version of the model. Nonetheless, the model has few restrictions. It does not identify the parameters to measure the components b, d, u. Another limitation of this model is that it does not acknowledge that trust changes over time and thus the model has no mechanism for monitoring trust relationships to reevaluate their constraints.

2.2.7.2 Bacharach and Gambetta's Work

Bacharach and Gambetta [BG00] embark on a re-orientation of the theory of trust. They define trust as a particular belief, which arises in games with a certain payoff structure. They also identify the source of the primary trust problem in the uncertainty about the payoffs of the trustee. The authors observe that in almost all games, the truster sees or observes trustee before making any decision and, therefore, can use the observations as evidence for the trustee's having, or lacking, trustworthy-making qualities. According to the authors, the truster must judge whether apparent signs of trustworthiness are themselves to be trusted. They also show how the secondary problem of trust can be treated as a particular class of signaling games. They do not exclude the possibility that the act of trusting itself can act as a signal of trustworthiness as trusting persons are believed to be more trustworthy than untrusting ones.

2.2.7.3 Trust Model based on Dempster-Shafer Theory

Teng et al.[TPC00] applies the Dempster-Shafer theory to propose a quantifiable measure of trust and develop a trust propagation model in the context of e-commerce environment. They attempt to answer the question "if A trusts B and B trusts C, then with how much certainty may A trust C?". The authors are primarily concerned about trust relationship between a customer and a trust authority and between a vendor and the trust authority. Several variables are used to define trust, some of which are borrowed from Manchala's work. The variables considered in this work are transaction cost, transaction history, indemnity, verification, authorization, transaction entity, trust authority, belief, and plausibility. At least two of these variables are used to capture the level of trust. A trust matrix is then formed with these trust variables. Each element of the matrix has two numbers, where the upper number is belief and the lower number is disbelief. "For the value of belief, 0 indicates 100% trust, 1 indicates 100% distrust." The Dempster-Shafer formula is used to merge two matrices and is applied to each element. Finally, the authors define a mechanism to compute the overall trust value between the customer and the vendor following a trust-chain relationship.

Although the model identifies different trust parameters, it does not provide any clue about how these parameters contribute in trust evaluation. For a transaction, the state of the transaction are expressed in terms of linguistic terms "Excellent, Good, Normal, Bad, Worst" and each constitute a row in the matrix. The columns are identified by number of micro-transactions and are denoted as "Small, Medium, Normal, High, Excessive". However, there is no clue provided what number of micro-transaction represents a particular level. Also the model is designed for e-commerce environment and cannot be used in reasoning about trustworthiness in different security contexts.

2.2.8 Graphical Approach to Model Trust

Purser [Pur01] presents a simple, graphical approach to model trust. He points out the relationship between trust and risk and argues that for every trust relationship, there exists a risk associated with a breach of the trust extended. The author observes that in managing a given risk, we extend trust to whatever mechanism we deploy to handle it. He then furnishes the elements of his model. Trust relationships are modeled as directed graphs. An entity can trust or be trusted by another entity; this is modeled by node in the graph that is labeled by the entities' names. The author defines trust as unidirectional and connects two entities; it is represented as a directed edge from the trusting entity to the trusted entity. The author also includes context, associated confidence level, associated risk and transitivity value. Context is anything that defines the scope of the trust; associated confidence level is the degree of confidence an entity has that the trusted entity will not breach the trust; associated risk is that materializes if the trust is breached and transitivity value indicates whether this trust can be passed on to a third party or not. The author also notes four rules that apply when using this model – (i) "A trust is considered to be fully defined only if all the associated values have been defined, (ii) Mutual trust is not allowed. Such a trust is modeled by two unidirectional trusts, (iii) A trust always has an associated context (unconditional trust is disallowed), (iv) A trust is only transitive or intransitive within a specific context." Finally the author demonstrates how to model a few IT related problems like outsourcing, Unix trusted host mechanism, and PKI using this graphical approach of modeling trust.

Though the approach is interesting and seems to be useful, it suffers from the limitation that it does not identify parameters to evaluate trust. It never mentions what are the "associated values" to "fully define" a trust. In fact, the model just provides a mechanism to represent and manage trust relationships graphically without actually defining trust or providing any meaning of trust. It does not provide any clue on how to establish a trust relationship. Trust, in this work, is used in a very generic sense and the author associates a "confidence level" to measure the degree of confidence that a trusted entity will not breach trust, when the confidence level itself could be the measure of trust.

2.2.9 Trust Model using Attributes to Evaluate Trust

Shankar et al. [SA02] propose a unified trust modeling system for the world of ubiquitous and pervasive computing. This is an "attribute vector model" which captures both the identity-based and context-based trust relationships. For cyberspace, the attribute vector is treated as a vector of credentials and for ubiquitous world, the vector is treated like a context vector, representing the real world contextual parameters. In the model, each entity S has an attribute vector $A(S) =< A_1, A_2, \ldots, A_k >$ and the trust relationship between entities is expressed as $D(S_1, S_2) = f(A(S_2))$ that is, the degree of trust or the trust value of S_2 is a function of attributes of S_2. This trust model is totally decentralized with the property that entities may or may not have any prior trust relationship. The trust decision is made if the trust value is above a threshold. The induced trust value serves as the "continuum of trust" upon which the participant can set any acceptable trust level. However, there is no method proposed to formalize these attributes. Also the function to evaluate trust is not specified to return a specific value of trust. The model is also not extensible as there is no scope of incorporating other factors in trust evaluation.

2.2.10 Bayesian Network-based Model

A Bayesian network-based trust model has been proposed in [WV03]. The model calculates peers' trust and reputation based on individual experience and recommendations from other peers respectively. This trust and reputation are used to find suitable peers. The trust is context-specific and dynamic; it can increase or decrease with further experiences and decays with time. A peer evaluates another peer's trust in two contexts: capability as a file provider and reliability as a recommendation provider. A peer considers 'truthfulness' and 'similarity' of another peer to compute the trust. The 'similarity' factor is particularly important for calculating trust in 'providing recommendation' context. The factor shows the basis of the judgment of the recommender about the trustee peer. A peer requesting for resource may consider different aspects of the provider namely, file type, file quality, download speed, to compute the trust on the provider. Bayesian network helps to aggregate trust on such different aspects to compute a single trust. A peer maintains a Bayesian network for each provider. The root represents the peer's overall trust on the trustee peer as a file provider. The node has two values 'satisfying' and 'unsatisfying'. The associated probability is percentage of satisfying (unsatisfying) interactions. Leaf nodes under the root represent the peer's trust on different aspects of the provider. A conditional probability table (CPT) is associated with each leaf node representing the conditional probability of the event corresponding to the node. These CPTs help a peer to compute probabilities that the corresponding file provider is trustworthy in different aspects. A peer updates its Bayesian network after each transaction. The degree of satisfaction is measured by the weighted sum of degree of satisfaction in file quality and in download speed. If the total satisfaction level is over a threshold, then the interaction is satisfactory, otherwise unsatisfactory. When a peer does not have information to compute trust on a file provider, it asks for recommendation from other peers. The peers answer that query consulting their own Bayesian network about the file provider. The requester peer calculates the reputation of the file provider as a weighted sum of averages of recommendations from trustworthy peers and unknown peers. Recommendations from untrustworthy peers are discarded. If the total recommendation is over a threshold, it is accepted. After each interaction the peer also updates the trust on the recommenders. The scheme proposes another way to measure the trustworthiness of a peer as recommender. Peers can exchange and compare their Bayesian network for a common file provider to update their trust on each other as a recommender. It is equivalent to compare all the past interactions of the two peers.

Bayesian network helps a peer to infer trust in different aspects of another peer using the CPTs. Number of interactions is an easy way to build the table. This approach saves time and effort to build trust in different aspects separately. Also change of conditions are easy to incorporate. However, the scheme assumes that all peers are truthful in telling their evaluation. This is not a reasonable assumption to make as malicious peer may give false recommendations. Sharing of Bayesian networks may help to find peers with similar preferences. But after few exchanges a malicious peer can tune its Bayesian network to manipulate trust about other peers. That is, they may present a good peer as bad and a bad as good. There is no mechanism to choose a reliable peer before exchanging the Bayesian networks. Effect of collusion is reduced as after each interaction trust of file provider as well as the reputation of the recommenders are updated. Finally, for a large network managing a Bayesian network for each resource provider can be a problem. The same problem will arise when there are many possible trust aspects. Therefore, though the framework can work well for small networks, it is not suitable for large networks with different aspects of trust.

2.3 Discussion

The notion of trust has definitely been enriched by the large volume of multidisciplinary research. However, at the same time, it suffers from lack of an unanimous meaning. The diversity in the research domains has given rise to this obfuscation and makes it difficult to find a consensus on the definition of trust. The list of definitions of trust presented in Section 2.1 shows the diversity in the definition of trust among the scholars from different fields. Their view of trust is different from each other. Also, as evident from the discussion in Section 2.2, even within computer and information science domain, we have varying models of trust. Each of these models has its own interpretation and representation of trust. Several secure systems have been developed and different applications have been designed using these concepts. Since, these models vary from each other in interpretation, representation, method of evaluation and management of trust, it is difficult to make use of these systems and applications in cooperative world. Today's business and market, especially like virtual organizations and e-commerce, are geared toward cooperative functioning of each entity involved in the business. The lack of agreement about the meaning and representation of trust among these entities hinders the proper function and growth of the cooperative business. These motivate us to define a single unified model for the specification and reasoning about trust. Because, no single existing trust model is suitable to be chosen as "standard" trust model. This is evident from the problem arising in the following scenarios.

2.3.1 Scenario 1

We take the example of *CyberCraft Initiative* [1] discussed by Michael Stevens in his thesis [Ste07]. "A CyberCraft is a system to provide command control and communications for packages that defend Air Force information systems." A CyberCraft is analogous to an actual air craft and can load different softwares to automatically defend air force information system. "The CyberCraft initiative is creating a set of standards that future cyberspace weapon system are to be built to. Because of the speed of operations in cyberspace, both offensive and defensive, the CyberCraft will operate mostly autonomously". One of the major concern regarding the CyberCraft Initiative is "What is required for a commander to trust a CyberCraft to autonomously defend military information systems?" A CyberCraft has some hardware/software construct called *agent* that resides on a host computer. This agent loads softwares, called *payloads*, which allow CyberCrafts to accomplish different tasks like intrusion detection, network vulnerability scanning, host-based firewall management, virus and worm detection and remediation, policy enforcement for unauthorized softwares, report host computer's or overall network health etc. The payloads can be sensors, decision engines, or effectors. Trust should be incorporated into all of these three type of payloads to perform in a cooperative manner. The commander in charge of network administration uses several such CyberCraft to protect the military information system. Different agents residing on different CyberCrafts monitors and provide information regarding the network and network administrator, depending on the information, take decision about appropriate action. If an agent reports that a target machine is running Windows XP and another

[1]For the details of CyberCraft Initiative of U.S. Air Force, the readers are referred to Michael Stevens' thesis [Ste07]

CyberCraft agent reports that the machine is running Linux, depending on the underlying operating system of the target machine, the commander triggers some CyberCrafts agent to load specific payloads (say, effectors).

2.3.2 Scenario 2

Consider the example of an e-commerce transaction where a customer Alice is interested to buy an air-ticket as well as reserve a hotel from an online travel system. The air-ticket reservation and purchase service and hotel reservation service are also available separately from two different service providers S_1 and S_2. Alice can avail both services from any one provider or she can avail one service from one provider and the other service from the other provider. During this transaction Alice is required to reveal some personal information like physical address, billing address, credit card information, phone number(s) etc. to make a successful purchase and reservation. Though she is willing to disclose the information to complete the transaction, she is reluctant to release those sensitive personal information to all and sundry. Therefore, before interacting with the service provider(s), Alice is willing to know to what extent the service provider(s) are capable of maintaining her privacy. That is, how much she can *trust* S_1 or S_2 to keep her personal information private, though one or both the providers may not be totally unknown to Alice. Let us assume that Alice read the privacy preservation policy of S_1 and S_2 and get the feeling that the providers offer different degrees of privacy preservation. Here we assume that Alice is aware of the recent technologies and security issues involved in online transactions.

2.3.3 Limitations of Existing Models

Most of the existing trust models view trust as a binary notion. Confidence is measured in terms of 'total trust' or 'no trust'. This binary model of trust differs considerably in semantics from the social models of trust used by policy makers. In sociology, trust means the assured reliance on the character, ability, strength or truth of someone or something. This assurance level can be of different degrees, leading to entities being labeled trusted to various levels. As we have seen, different interpretations and representations of trust creates inferential ambiguities, for example in the comparison or composition of information gathered from different sources or in the composition of systems that involve interaction between human and computational devices. Consider the scenario of CyberCraft initiative. The commander's decision to take appropriate action depends on the information gathered from different agents. The agents are trusted to different levels. Suppose, the agent with higher trust reports the OS in the target machine as Linux and the low trusted agent reports it as Windows. However, both views may be correct if the target machine is running Linux inside a virtual machine on top of Windows XP at the hardware level. Thus, using Avižienis et al.'s definition of trust [ALRL04], it is evident that it will be naïve for the commander to completely trust the information he gathers from the agents. Intuitively, it seems, that the composed information will be *trusted to a certain degree* but not necessarily *wholly trusted.* A single trust model from the existing models of trust is not sufficient to answer all of the following questions:

1. Which of the reports about the OS on the target machine is correct?
2. What expectations can the commander have about the usefulness of the composed information?
3. How much assurance the commander can have to load certain payloads based on the comparison or composition result?
4. What are the critical activities that the commander cannot fulfill using this information, with certain confidence?

In our e-commerce scenario, existing models of trust do not help Alice to answer questions such as:

1. How Alice can compare the 'correctness' of the information from different privacy preservation policies?
2. How Alice can combine the information if she decides to avail different services from different service providers?
3. What sensitive information can Alice reveal with certain confidence?

Some models from the existing set of trust models discussed in Section 2.2 can answer some of the above questions. However, to answer all these questions no single model is sufficient.

Also, some interpretation and representation of trust may be suitable to apply in scenario 1, but may not be suitable for scenario 2. Another problem with the existing models is that they are not easily extensible. The models use specific parameters and constructs to evaluate trust and it is difficult to incorporate new parameters without affecting the existing constructs. These make it difficult to use the systems and applications, having different meaning and mechanism to evaluate trust, in cooperative world. For example, a system built around a model of trust based on 'experience' only cannot function in a cooperative manner with a system built around a model of trust based on 'recommendation' (or any third party information) only. Last but not the least, existing trust models do not address the issue of evaluating trust when relevant information is partially or totally unavailable. These problems motivate us to integrate different perspective of trust to define a unified, extensible trust model that can answer the problems mentioned above. The following chapter presents our model.

Chapter 3

The Vector Trust Model

3.1 Overview of The Model

Among all the definitions of trust, discussed in Section 2.1, we find the definition proposed by Grandison and Sloman [GS00] most suitable for our purpose. Hence, in our model we adopt their definition of trust and define trust as

Definition 1 [Trust] *Trust is defined to be the measure of level of assurance that an entity is competent or benevolent within a specific context.*

In the same work, Grandison and Sloman define *distrust* as the "lack of firm belief in the competence of an entity to act dependably, securely and reliably". However, we believe distrust is somewhat stronger than just "lacking a belief". Grandison and Sloman's definition suggests the possibility of ambivalence in making a decision regarding distrust. We choose to be more precise and thus define distrust as follows

Definition 2 [Distrust] *Distrust is defined to be the measure of level of assurance that an entity is incompetent or malevolent within a specific context.*

Although we define trust and distrust separately in our model, we allow the possibility of a neutral position where there is neither trust nor distrust. As we elaborate on the model this will become more clear.

Trust in our model is specified as a trust relationship between a truster – an entity that trusts the target entity – and a trustee – the target entity that is trusted. The truster is always an active entity (for example, a human being or a subject). The trustee can either be an active entity or a passive entity (for example, a software). We use the following notation to specify a trust relationship – $(A \xrightarrow{c} B)_t^N$. It specifies A's *normalized* trust on B at a given time t for a particular context c. This relationship is obtained from the simple trust relationship – $(A \xrightarrow{c} B)_t$ (which is shown in Figure 3.1) – by combining the latter with a *normalizing factor*. This normalized trust relationship is expressed as a tuple of ordered

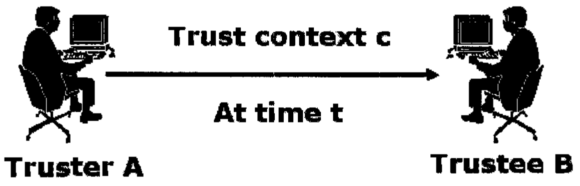


Figure 3.1: Trust relationship

values, where each value corresponds to the measure of the parameter influencing trust evaluation. We also introduce a concept called the *value* of a trust relationship. This is denoted by the expression $\mathbf{v}(A \xrightarrow{c} B)_t^N$ and is a number in $[-1,1] \cup \{\perp\}$ that is associated with the normalized trust relationship. A trustee is completely trusted (or distrusted) if the value of the trust relationship is 1 (-1). Values in the range (0,1) represent different levels of trust and the values in the range (-1,0) represents different levels of distrust. The 0 value represents trust neutrality that is, the trustee is neither trustworthy nor un-trustworthy. The special symbol $\perp$ is used to denote the value when there is not enough information to decide about trust, distrust, or neutrality.

We want to make a note here that the term "vector" is not used in a strict sense. Like the regular mathematical notion of a vector, we have a tuple (set of values) and a single value (dependent on the set of values) to represent a trust relationship. This is why we choose to name our trust model as 'vector trust model'. However, we do not assign the mathematical properties of a regular vector to our trust vector nor we define the standard vector operations like 'dot product', 'cross product' etc. of vectors on our trust vector. Also, the numerical value associated with the vector is not calculated in the standard way that is, taking square-root of sum of squares of the component values. Because that would have removed the effect of negative values of a component on the overall value.

3.1.1 Qualitative vs. Quantitative Approach

Our trust model aims to provide the notion of a trust value to represent levels of trust. We face two choices for trust values – qualitative or quantitative. If qualitative values are used, then degree or level of trust can be expressed in terms of a set of discrete values such as *high, medium, low* or *very trustworthy, trustworthy, untrustworthy, very untrustworthy* etc. The advantage of such a scheme is that it is quite intuitive. However, the challenges are significant. First, it is rather difficult to define precisely the semantics of such levels; semantics across different systems can vary. Second, determining the appropriate number of such degrees for a particular system is not straightforward; in fact it can tend to be rather ad hoc. Third, determining how the degrees from different domains can be compared and combined is most difficult. For example, what is the resultant of the trust composition of "high" of one domain and "medium" of another domain. Fourth, use of such qualitative terms limits number of distinct trust levels to a small number of levels. This is, in particular, not suitable for fine-granular trust comparison. As indicated by Stephen Marsh in his thesis [Mar94], with this limited set of trust levels, ".. it is not possible to say 'I trust him more than her, by a small amount,'..". Last but not the least, a problem with such discrete degrees is that it is not easy to represent ignorance or neutrality with respect to trust or distrust.

Quantifying trust through numeric values alleviate such problems. Moreover, mathematical operations on degrees of trust can be defined that allow proper comparison and combination of degrees from different domains. Comparison using a continuous range of quantitative values allows fine-granular comparison result, which may help significantly in making appropriate security decisions. For example, a small difference in trust level may make significant changes in organizational policies or actions (as observed by Marsh [Mar94], "In a formalism using quantitative data, it is possible that small differences in individual values produce relatively large differences in the overall result"). The above reasons lead us to adopt numeric values with continuous range for trust levels. Instead of limiting ourselves to a single value for trust (or distrust), we define a trust value in terms of a vector of numeric values. We use a vector so that we can specify the effects of the many different factors that influence trust. However, we also believe that there are times when a single numeric value is more intuitive than a vector of values – particularly when making comparisons in an informal manner. This leads us to define the notion of a "value" for a trust vector. It is a either single numeric value in the range $[-1, 1]$ or a special value $\perp$.

3.1.2 Trust Dynamics & Propensity to Trust

At this stage we point to two characteristics of trust (or distrust) that shapes our model. The first is the dynamic nature of trust. Trust changes over time. Even if there is no change in the underlying factors that influence trust over a time period, the value of trust at the end of the period is not the same as that at the beginning of the period. Irrespective of our initial trust or distrust decision, over a period of time we gradually become non-decisive or uncertain about the trust decision. This leads us to claim that trust (and alternately distrust) decays over time - both tends towards a non-decisive value over time. The second characteristic is, what is often called the *propensity* to trust [GS00]. Given the same set of values for the factors that influence trust, two trusters may come up with two different trust values for the same trustee. We believe, that there are two main reasons for this. First, the reason behind this is, during evaluation of a trust value, a truster may assign different weights to different factors that influence trust. The weights will depend on the truster's policy. If two different trusters assign two different sets of weights, then the resulting trust value will be different. The second reason is applicable only when the truster is a human being and is completely subjective in nature – one person may be more trusting than another. We believe that this latter concept is extremely difficult to model in an objective manner. We choose to disregard this feature in our model. We capture the former factor using the concept of a *trust-parameter weight policy vector*, which is simply a vector of weight values. Using this weight vector on the simple trust relationship provides the normalized trust relationship.

3.2 Trust Parameters

We begin by identifying four different parameters that influence trust values – *interactions, properties, reputation,* and *recommendation.*

3.2.1 Interactions

Definition 3 **[Interactions]** *The* interactions *of a truster about a trustee is defined as the measure of the cumulative effect of a number of events that were encountered by the truster with respect to the trustee in a particular context and over a specified period of time.*

The trust value of a truster on a trustee for some context can change because of the the truster's *interactions* with the trustee in the particular context. Consider the following scenario with our example scenario 2 from Section 2.3.2. From now onwards we will consider this scenario as our running example.

Alice has been witnessing that the information she has provided to S_1 has not been disclosed for the past five months, that is her privacy has not been breached through *information disclosure* by S_1. Initially Alice was neutral towards S_1's capability; however having benefited from it, she now trusts S_1 more to protect her privacy during an electronic transaction.

A truster can categorize each interaction with a trustee as *trust-positive*, *trust-negative* or *trust-neutral* interaction. A trust-positive interaction increases trust degree whereas a trust-negative interaction increases distrust degree. A trust-neutral interaction contributes neither way.

3.2.2 Properties

Definition 4 **[Properties]** *The* properties *of a trustee for a particular context is defined as a measure of the characteristic attributes or information of the trustee for which the truster can have some assertion to be truly related to the trustee.*

The trust value of a truster on a trustee can change because of some *properties* that the trustee possesses. Information about these properties of trustee may be obtained by the truster in some earlier time for some purpose or, it may be a piece of information about the trustee for which the truster can have a proof to be true. As with interactions, we have *trust-positive*, *trust-negative*, and *trust-neutral* property. In the context of our example, Alice may come to possess a certificate from a well-known certifying authority that S_1 uses a secure channel established through SSL3 to do all the transaction and the data is encrypted with 512-bit RSA key. This information gives her more confidence on S_1 that her privacy will not be violated through a confidentiality and/or integrity breach during a transaction with S_1.

3.2.3 Reputation

The notion of *reputation* has been used for quite a long in discipline like Economics [SK03, HW04] as well as in Computer Science, especially in trust related research. Some of the recent works on reputation ([JI02, DVPS03, SUP04, XL04, SFWC04, BB04, KNS05]) in Computer Science use it as a measure of trustworthiness of agents/peers/systems etc. For example, in [DVPS03, SUP04, XL04, SFWC04] reputation has been used to find a reliable peer in a Peer-to-Peer (P2P) system, [BB04] proposes a reputation system in mobile ad-hoc networks, [KNS05] uses reputation system in history-based access control. In all the above works reputation is measured in terms of either trustee's behavior or properties, or truster's experience about the interactions with the trustee, or some information obtained from a third party or, a combination of these. In our model we propose separate notions for truster's actual experience about the interaction behavior of trustee by the component *interactions*, about trustee's properties by the component *properties*, about third party information about trustee by *recommendation.* So we propose a notion of 'reputation' different from the earlier proposed notions and define it as follows:

Definition 5 [Reputation] *A* reputation *of a trustee is defined as a measure of the non attributable information (in terms of feedback or properties) about the trustee to the truster in a particular context.*

We posit that a trustee's reputation is totally non-attributable. The truster does not have any guarantee for those to be true. The component *reputation* is difficult to compute objectively. It is more subjective in nature and completely depends on the truster's discretion. The truster may get the idea about the reputation of trustee from various sources like reviews, journals, news bulletin, people's opinion etc. However, with this reputation, without having specific information about the trustee, the truster can build an opinion about the trustee in the context. In the light of our example, the truster Alice will trust S_1 more if she reads some positive comments, made by other customers, about S_1's capability to protect personal information during an online transaction. Being unknown to her, Alice considers all these customers as non-attributable and hence the information is considered as 'reputation' rather than 'recommendation'.

3.2.4 Recommendation

Definition 6 [Recommendation] *A* recommendation *about a trustee is defined as a measure of the subjective or objective judgment of a recommender about the trustee to the truster.*

The trust value of a truster on a trustee can change because of a *recommendation* for the trustee. For example, a truster can ask someone close to him, who happens to know the trustee, about the latter's credibility (within the scope of the trust context). If that third person says "good words" about the trustee, the truster tends to have faith on the trustee. It is important to note that the importance of the judgment of the third person to the truster depends on how much the truster trusts the third person's ability to judge others. In our model we use the degree of trust between a truster and a recommender to evaluate the recommendation for the trustee. As before we can have a *trust-positive*, *trust-negative*, and a *trust-neutral* recommendation. Recommendations can be obtained by the truster from more than one source and these together will contribute to the final trust relationship. In our running example, Alice may request her friend(s) or associate(s), whom she knows to have similar online transactions with the service providers, to provide some feedback about the providers' capability of protecting privacy during online transaction.

3.3 Trust Evaluation

3.3.1 Evaluation of the Parameters

To compute a trust relationship we assume that each of these four factors is expressed in terms of a numeric value in the range $[-1, 1]$ and a special value $\perp$. A negative value for the component is used to indicate the *trust-negative* type for the component, whereas a positive value for the component is used to indicate the *trust-positive* type of the component. A 0 (zero) value for the component indicates trust neutral. To indicate a lack of value due to insufficient information for any component we use the special symbol $\perp$. There will be some situation where a truster may need to compute a value involving a real number a and the special value $\perp$. For that we define the following properties of $\perp$. If $\mathbb{R}$ is the set of real numbers, then

1. $a \cdot \perp = \perp \cdot a = \perp, \ \forall \ a \in \mathbb{R}$,
2. $a + \perp = \perp + a = a, \ \forall \ a \in \mathbb{R}$,
3. $\perp + \perp = \perp$ and $\perp \cdot \perp = \perp$

3.3.1.1 Evaluating Interactions

We model interactions in terms of the number of events encountered by a truster, A, regarding a trustee, B in the context c within a specified period of time $[t_0, t_n]$. We assume that A has a record of the events since time t_0. An event can be trust-positive, trust-negative or, trust-neutral depending whether it contributes towards a trust-positive interaction, a trust-negative interaction or, a trust-neutral interaction.

Let $\mathbb{N}$ denote the set of natural numbers. The set of time instances $\{t_0, t_1, \ldots, t_n\}$ is a totally ordered set, ordered by the temporal relation $\prec$, called the *precedes-in-time* relation, as follows: $\forall i, j \in \mathbb{N}, \ t_i \prec t_j \Leftrightarrow i < j$. We use the symbol $t_i \preceq t_j$ to signify either $t_i \prec t_j$ or $t_i = t_j$. Let e_k denote the k^{th} event and each event happens at a specific time instance. We define the concept *event-occurrence-time* as follows:

Definition 7 [Event-occurrence-time] *The* event-occurrence-time, *ET, is a function that takes an event e_k as input and returns the time instance, t_i at which the event occurred. Formally, $ET(e_k) = t_i$.*

We divide the time period $[t_0, t_n]$ into a set $\mathcal{T}$ of n intervals, $[t_0, t_1], [t_1, t_2], \ldots, [t_{n-1}, t_n]$ such that for any interval $[t_i, t_j], t_i \prec t_j$. A particular interval, $[t_{k-1}, t_k]$, is referred to as the k^{th} interval. We extend the $\prec$ relation on $\mathcal{T}$ and the time intervals are also totally ordered by the $\prec$ relation as follows: $\forall i, j, k, l \in \mathbb{N},\ [t_i, t_j] \prec [t_k, t_l] \Leftrightarrow t_j \prec t_k$. The intervals are non-overlapping except at the boundary points, that is $\forall i, j, k, l \in \mathbb{N},\ [t_i, t_j] \cap [t_k, t_l] = \emptyset$. Lastly, for two consecutive intervals $[t_i, t_j]$ and $[t_j, t_k]$ if $ET(e_k) = t_j$ then we assume $e_j \in [t_i, t_j]$.

Let $\mathcal{P}$ denote the set of all trust-positive events, $\mathcal{Q}$ denote the set of all trust-negative events, and $\mathcal{N}$ denotes all trust-neutral events (that is, $\mathcal{E} = \mathcal{P} \cup \mathcal{Q} \cup \mathcal{N}$). We assume that all trust-positive events contribute equally to the formation of a trust value and all trust-negative events also do the same. The trust-neutral events contribute nothing. We assign equal numeric weights to all events, trust-positive or trust-negative. within the same interval. Let v_k^i be the weight of the k^{th} event in the i^{th} interval. We assign a weight of $+1$ if an event is in the set $\mathcal{P}$, -1 if the event is in the set $\mathcal{Q}$, and 0 if the event is in $\mathcal{N}$. Formally, if e_k^i denote the k^{th} event in the i^{th} interval, then

$$v_k^i = \begin{cases} +1 & \text{, if } e_k^i \in \mathcal{P} \\ -1 & \text{, if } e_k^i \in \mathcal{Q} \\ 0 & \text{, if } e_k^i \in \mathcal{N} \end{cases}$$

Definition 8 [Incidents] *The* incidents *IN_j, corresponding to the j^{th} time interval is the sum of the values of all the events, trust-positive, trust-negative, or neutral for the time interval. If no event happened in j^{th} time interval, then $IN_j = \perp$. If n_j is the number of events that occurred in the j^{th} time interval, then*

$$IN_j = \begin{cases} \perp & \textit{, if } \nexists e \in \mathcal{E} \textit{ such that } ET(e) \in [t_{j-1}, t_j] \\ \sum_{k=1}^{n_j} v_k^j & \textit{, otherwise} \end{cases}$$

Events far back in time does not count as strongly as very recent events for computing trust values. We give more weight to events in recent time intervals than those in distant intervals. To accommodate this in our model, we assign a *non-negative* weight w_i to the i^{th} interval such that $w_i > w_j$ whenever $j < i,\ \ i, j \in \mathbb{N}$.

Interactions has a value in the range $[-1, 1] \cup \{\perp\}$. To ensure that the value is within this range we restrict the weight w_i for the i^{th} interval as $w_i = \frac{i}{S},\ \forall i = 1, 2, \ldots, n$, where $S = \frac{n(n+1)}{2}$. Then the interactions of A with regards to B for a particular context c is given by

$$ {}_A\mathrm{I}^c_B = \frac{\sum_{i=1}^{n} w_i IN_i}{\sum_{i=1}^{n} n_i} \tag{3.1} $$

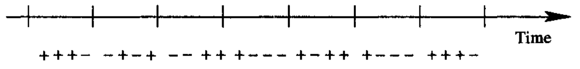
If A does not have any interaction with B in the j^{th} interval, then $IN_j = \perp$. Thus,

$$\begin{aligned} \sum_{i=1}^{n} w_i IN_i &= \sum_{i=1}^{j-1} w_i IN_i + w_j IN_j + \sum_{i=j+1}^{n} w_i IN_i \\ &= \sum_{i=1}^{j-1} w_i IN_i + \perp + \sum_{i=j+1}^{n} w_i IN_i \quad \text{(by property (1) of } \perp) \\ &= \sum_{i=1}^{j-1} w_i IN_i + \sum_{i=j+1}^{n} w_i IN_i \quad \text{(by property (2) of } \perp) \end{aligned}$$

If there is a situation where nothing happened between the truster A and the trustee B over the entire time period $[t_0, t_n]$, then $IN_i = \perp,\ \forall\ i = 1, 2, \ldots, n$. As a result, we have $w_i IN_i = \perp,\ \forall\, i = 1, 2, \ldots, n$ which implies ${}_A\mathrm{I}^c_B = \perp$. The above is different from the situation when ${}_A\mathrm{I}^c_B = 0$. If the number of positive events is equal to number of negative events in each interval, then $IN_i = 0,\ \forall\ i = 1, 2, \ldots, n$ and as a result we get ${}_A I^c_B = 0$. But the former case occurs only when there is no interaction between the truster and the trustee over the entire time period.

To illustrate our concept of interactions we use the following example. We use the symbol "+" to denote positive events and the symbol "-" to denote negative events.

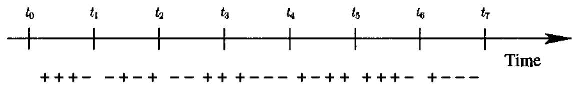
Example 1 *Let us assume that the customer Alice from our e-commerce example, encounters the following events related to S_1 over the time period $t_0 - t_7$*



where typical trust-negative interaction is receiving an unsolicited e-mail/phone-call/letter advertising different deals offered by S_1 *showing that* S_1 *is breaching her privacy through* information exploitation.

To compute Alice's interactions for S_1*, we divide the time period into the intervals –* $[t_0, t_1], \ldots, [t_6, t_7]$. *Applying our theory, we have the following incidents:* I_0 *for interval* $[t_0, t_1] = +2, IN_1 = 0, IN_2 = 0, IN_3 = -2, IN_4 = +2, IN_5 = -2$ *and* $IN_6 = +2$. *The weights assigned to each time interval are as follows –* w_0 *(for interval* $[t_0, t_1]$*) = 0.04,* $w_1 = 0.07, w_2 = 0.11, w_3 = 0.14, w_4 = 0.18, w_5 = 0.21$ *and* $w_6 = 0.25$ *(for interval* $[t_6, t_7]$. *Thus the value for Alice's interactions regarding* S_1*, over the period* $[t_0, t_7]$ *is* $0.0086 \approx 0.009$.

Example 2 *Consider the second set of events that Alice encounters over the same time period* $t_0 - t_7$ *for* M_2.



The difference between this set of events and the one in example 1 is that we have more negative events that have happened recently. The total number of trust-positive and trust-negative events are the same in both. We get a value of $0.0029 \approx 0.003$ *for interactions with this set of events.*

3.3.1.1.1 Comparison of Interaction with Bayesian Reputation System Researcher [HW04, MMH02, JI02, WJI05] have previously used Bayesian systems to rate entities based on results of positive and negative past events. In such works, the updated rating or reputation score is calculated using previous score and new rating information. The reputation score is represented in the form of beta probability density function. The beta-family of distribution (or the beta-function) involves two parameters α and β representing amount of positive and negative ratings respectively. These parameters are continuously updated based on the occurrence of events. The beta distribution $f(\theta|\alpha, \beta)$ can be expressed as follows:

$$f(\theta|\alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{Beta(\alpha, \beta)} \tag{3.2}$$

The function can also be expressed using *gamma function*, as

$$f(\theta|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \tag{3.3}$$

where, $0 \leq \theta \leq 1$ and α, $\beta > 0$. The Γ function is defined using the recursive formula $\Gamma(x) = x\Gamma(x-1)$, $\Gamma(1) = 1$. Alternatively, $\Gamma(x) = x!$. In this equation, $\alpha =$ number of positive events and β= number of negative events.

Our formulation of *interactions*, using the above distribution, could be as follows: Let p and q be number of trust-positive and number of trust-negative events within $[t_0, t_n]$ respectively (we can ignore the trust-neutral events, as they do not affect the parameter *interactions* to a great extent). Then the probability distribution function of observing trust-positive event in future can be expressed as a function of past events by setting

$$\alpha = p + 1 \text{ and } \beta = q + 1, \text{ where } p, q \geq 0$$

and the previous equation (Equation 3.3) takes the form

$$f(\theta|p, q) = \frac{\Gamma(p+q+2)}{\Gamma(p+1)\Gamma(q+1)}\theta^{p}(1-\theta)^{q} \tag{3.4}$$

However, we choose not to use beta-function to compute 'interactions'. This is because, for fixed number of p and q, the expected value of $f(\theta|p, q)$, given by $E(\theta) = \frac{p+1}{p+q+2}$, is a probability value representing the average relative frequency of positive events in future. Thus, it cannot capture the effect of recent events for computing interactions. On the contrary, our equation to compute interaction is sensitive to the distribution of positive and negative events over the time period $[t_0, t_n]$.

To show this, we execute a little experiment. We take 50 events with 25 positive events ($p = 25$) and 25 negative events ($q = 25$). The following two cases are considered:

- **Case 1:** All 50 events distributed equally over 5 intervals, each interval having 10 events.

- **Case 2:** All 50 events distributed unevenly over 5 intervals.

For each of these cases, we consider three sub-cases.

1. **Sub-case 1:** Concentration of positive events are more toward the recent intervals.
2. **Sub-case 2:** Concentration of positive events are more toward the past intervals (we just reverse the event-distribution in sub-case 1).
3. **Sub-case 3:** Positive and negative events are distributed in oscillating manner.

For each of these cases, we observe that the value of 'interactions' varies depending on the distribution. Nonetheless, the $E(\theta)$ value is same for each of the sub-cases. The following Table 3.1 summarizes the results. If the number of intervals is changed then the 'interactions' values also change. But the $E(\theta)$ values still remain the same. Table 3.2 shows the results of the experiment with all 50 events distributed equally over 10 intervals. The above results show that our way of evaluating 'Interactions' captures the relative importance of events over time as well as the distribution of positive and negative events over the entire time period.

#Intervals	#+ve events	#-ve events	Distribution of events	I	$E(\theta)$
5	25	25	Sub-case 1 of case 1	0.056000	0.500000
5	25	25	Sub-case 2 of case 1	-0.056000	0.500000
5	25	25	Sub-case 3 of case 1	0.000000	0.500000
5	25	25	Sub-case 1 of case 2	0.068000	0.500000
5	25	25	Sub-case 2 of case 2	-0.068000	0.500000
5	25	25	Sub-case 3 of case 2	-0.001333	0.500000

Table 3.1: Comparison of I and $E(\theta)$

#Intervals	#+ve events	#-ve events	Distribution of events	I	$E(\theta)$
10	25	25	Sub-case 1	0.038909	0.500000
10	25	25	Sub-case 2	-0.038909	0.500000
10	25	25	Sub-case 3	-0.001091	0.500000

Table 3.2: Comparison of I and $E(\theta)$ with 10 intervals

3.3.1.2 Evaluating Properties

The parameter "properties" is more difficult to compute and is, to some extent, subjective. To begin with, each truster must define its own criteria for gradation of properties regarding a particular entity. To assign a value to the *properties* component, the truster must assign a value between -1 and +1 for each attribute of the trustee. How the values are assigned, depends on the scheme and policy (called, *property evaluation policy*) of the truster. Also the truster solely is responsible for assigning the relative weights to different attributes or information. Average of these values gives the value for the component *properties*. It is possible that the truster has insufficient information to assign a value to properties. For these types of cases, we assign $\perp$ to the component. Thus, if the truster is aware of k attributes of the trustee, then properties of trustee B according to truster A in context c is evaluated as

$${}_AP_B^c = \frac{\sum_{i=1}^{k} v_i}{k} \tag{3.5}$$

where $v_i \in [-1, 1]$, $\forall i = 1, 2, \ldots, k$. v_i is the value assigned to i^{th} attribute of B and is determined by the underlying policy of the truster. Note, ${}_AP_B^c = \perp$ is different from ${}_AP_B^c = 0$. Value 0 implies that after evaluating the information according to trust policy, the truster's decision is neutral, whereas the value '$\perp$' implies "lack of information" regarding the attributes of the trustee, that is that there is not enough data to determine 'properties' of the trustee.

Example 3 *Let Alice determine that S_1 uses SSL3 as the secure communication method and assigns a value of 0.65 for this attribute. She then receives a digital credential from S_1 issued by, say* Better Business Bureau *which proves latter's authority to work as a Web service provider related to travel. By evaluating this certificate about S_1 as a service provider, Alice assigns a value, say 0.8 to this piece of information. Then she calculates the 'properties' component of $(Alice \xrightarrow{pp} S_1)_t$ as, ${}_{Alice}P_{S_1}^{pp} = \frac{0.65+0.8}{2} = 0.725$. By pp we denote the context of "reliability to access data and provide a privacy protection service".*

3.3.1.3 Evaluating Reputation

The component *reputation* is the most difficult one to compute. We believe, it is more difficult to assign values to reputation information than to a property information. Consequently this assignment is more subjective in nature and completely depends on the truster's discretion. We evaluate 'reputation' in the following way: The truster A gathers different reputation values about the trustee B from different sources and scales them within the range $[-1, 1]$. Note, the values may have been represented in different scales. Hence, A uses the simple translation formula $\frac{x-(-1)}{1-(-1)} = \frac{value-a}{b-a}$, where x is the transformed reputation value within $[-1, 1]$, and *value* is the collected reputation value, represented within a range $[a, b]$. The truster may also translate qualitative values like *low, medium, high* to a numeric value within $[-1, 1]$ with the translation *low = -1, medium = 0, high = 1.* Here we assume that all these collected values (qualitative and quantitative) are in the same reputation context. This type of translations helps the truster to aggregate different representation of reputation information by bringing them to a uniform platform. Finally, A computes average of all these translated values as the value of reputation component ${}_{A}REP_{B}^{c}$. Formally, if A has k such (translated) values $rv_1, \ldots, rv_k$, then

$$ {}_{A}REP_{B}^{c} = \sum_{i=1}^{k} rv_i/k \tag{3.6}$$

Note, ${}_{A}REP_{B}^{c} = \perp$ is different from ${}_{A}REP_{B}^{c} = 0$. Value 0 implies that the sum of the translated reputation values is zero, whereas the value '$\perp$' implies there is not enough data to determine 'reputation' of the trustee.

Example 4 *Let Alice come across a website where online customers have put some rating regarding service provider S_1's ability to preserve customers' privacy. Suppose the average rating is 3.5 in a scale of 5. Let Alice read another report which rates S_1 as 8 out of 10 for its use of secure method of communication to preserve customer's privacy during communication. A popular travel related magazine says S_1 is at 'medium' level regarding customers' privacy protection. Alice translates the above values as:* $\frac{rv_1+1}{2} = \frac{3.5}{5} \Rightarrow rv_1 = 0.4$, $\frac{rv_2+1}{2} = \frac{8}{10} \Rightarrow rv_2 = 0.6$ *and the rating 'medium' is translated to* $rv_3 = 0$. *Therefore, the value of the reputation component is calculated as,* ${}_{A}REP_{S_1}^{pp} = (rv_1 + rv_2 + rv_3)/3 = (0.4 + 0.6 + 0)/3 = 0.333$.

3.3.1.4 Evaluating Recommendation

An initial recommendation score, V_ψ, is a value in the range $[-1, 1]$ that is provided to the truster by the recommender ψ. To assist the recommender in generating this score for his feedback, the truster may provide a questionnaire to the recommender. The recommender uses the positive values to express his faith in the trustee while uses negative values to express his discontent. If the recommender has no conclusive decision, he uses zero as recommendation. It is quite possible that the recommender does not return a recommendation sore. In such a case A assigns the value $\perp$ to V_ψ.

A truster A, has a trust relationship with the recommender ψ. The context of this trust relationship will be to act "reliably to provide a service (recommendation, in this case)". This trust relationship will affect the score of the recommendation provided by the recommender. For example, let us say that A trusts ψ to a great extent to provide an appropriate recommendation for B but does not trust ψ' as much as ψ. ψ provides a recommendation score of -0.5 to A and ψ' also provides the same recommendation score. To A, ψ's -0.5 score will have more weight for computing the trust value on B than ψ's, although A will consider both the scores. Scaling the recommendation score based on the trust relationship between the truster and the recommender has one important benefit. Suppose that the recommender tells a lie about the trustee in the recommendation in order to gain an advantage with the truster. If the truster does not trust the recommender to a great degree then the score of this recommendation will be low with the truster. Note that the feedback from an unknown recommender or from a recommender with whom the truster does not have any prior trust relationship is scaled with the trust value $\perp$. Note also that if the truster distrusts a recommender to properly provide a recommendation, it will not ask for the recommendation to begin with or it can discard their recommendations immediately.

We use the trust of the truster on the recommender as a weight to the initial recommendation score returned by the recommender. We had introduced the expression $\mathbf{v}(A \xrightarrow{c} B)_t^N$ earlier to denote the *value* of a normalized trust relationship. We use this value as the weight. At this stage we do not specify how we generate this value. We leave that to a later section (Section 3.3.4). Following the above discussion, the *recommendation* ${}_{\psi}REC^{c}_{B}$ of a recommender ψ for an entity B to the truster A in a context c is given by ${}_{\psi}REC^{c}_{B} = (\mathbf{v}(A \xrightarrow{rec} \psi)^{N}_{t}) \cdot V_{\psi}$. In addition, the truster A may get recommendations about the trustee B from many different recommenders. Thus the recommendation value that the truster uses to compute the trust in the trustee is specified as the sum of all recommendation scores scaled to the range $[-1,1] \cup \{\perp\}$. If Ψ is a group of n recommenders then,

$$ {}_{\Psi}REC^{c}_{B} = \frac{\sum_{j=1}^{n}(\mathbf{v}(A \xrightarrow{rec} \psi_j)^{N}_{t}) \cdot V_{\psi_j}}{\sum_{j=1}^{n}(\mathbf{v}(A \xrightarrow{rec} \psi_j)^{N}_{t})} \tag{3.7} $$

where $\Psi = \{\psi_1, \ldots, \psi_n\}$. ${}_{\Psi}REC^{c}_{B} = \perp$ is different from ${}_{\Psi}REC^{c}_{B} = 0$. In the former case, either nobody responded or $\forall j,\ \mathbf{v}(A \xrightarrow{rec} \psi_j)^{N}_{t} = \perp$. In the latter case either all recommenders returned a score '0', that is all of them are neutral about the trustee in the trust context or $\forall j,\ \mathbf{v}(A \xrightarrow{rec} \psi_j)^{N}_{t} = 0$ that is, the truster is neutral about all the recommenders.

Example 5 *We continue with our example of Alice trying to establish a trust relationship with the web service provider(s). Let Alice now ask her friend Charlie, with whom she already has an established trust relationship, to recommend S_1 in the context of pp. Let she trust Charlie (denoted by F_1) in the context of "providing recommendation" with 0.8. The recommender F_1 returns a value (recommendation score) 0.55 for S_1. Then Alice evaluates the recommendation component of $(Alice \xrightarrow{pp} S_1)_t$ as ${}_{F_1}REC^{pp}_{S_1} = 0.8 \times 0.55 = 0.44$.*

Next, we consider the case where 4 other friends of Alice, namely F_2, F_3, F_4, and F_5 give recommendation about S_1 in the context pp and their recommendation scores are -0.7, 0.3, 0.8, 0.6 respectively (Let F_2 do not like the service provider S_1 for some reason and hence give a negative recommendation for that). Let Alice trust F_2, F_3, F_4 and F_5 with a degrees 0.4, 0.2, 0.75 and 0.5 respectively, in the context of "providing recommendation" (we assume, for the time being, that these values have been derived somehow from the corresponding trust relationships). Let $\Psi = \{F_1, F_2, F_3, F_4, F_5\}$. Then the recommendation is calculated as ${}_{\Psi}REC^{pp}_{S_1} = \frac{0.8\times0.55+0.4\times(-0.7)+0.2\times0.3+0.75\times0.8+0.5\times0.6}{0.8+0.4+0.2+0.75+0.5} = 0.423$. Note that F_2's bias has been offset to a great extent.

3.3.2 Simple Trust Vector

After evaluating the parameters within the range $[-1,1] \cup \{\perp\}$, we are in a position to specify the "simple" trust relationship between the truster A and the trustee B in a context c at time t. We denote it as $(A \xrightarrow{c} B)_t$ and is specified as

$$(A \xrightarrow{c} B)_t = [{}_AI^c_B,\ {}_AP^c_B,\ {}_AREP^c_B,\ {}_\Psi REC^c_B]$$

Therefore, simple trust vector between a truster and a trustee represents the vector of parameter values as computed at time t.

Example 6 *After computing the parameters, Alice's simple trust relationship with the service provider S_1 is $(Alice \xrightarrow{pp} S_1)_t = [0.009, 0.725, 0.333, 0.423]$.*

3.3.3 Normalizing the Trust Vector

We mentioned earlier in Section 3.1.2 that a truster may give more weight to one of the parameters than others in computing a trust relationship. For example, a truster A may choose to emphasize interactions and properties more than reputation and recommendation. In such case the truster will want to consider the reputation as well as recommendation factor to a lesser extent than interactions and properties about the trustee. Which particular component needs to be emphasized more than the others, is a matter of trust evaluation policy of the truster. The truster's policy can be trustee specific or can be the same for all trustees. Similarly it can be context specific or context independent. This policy is represented by the truster as a trust-parameter weight policy vector.

Definition 9 [Trust-Parameter Weight Policy Vector] *The* trust-parameter weight policy vector ${}_AW^c_B$ *of a truster A with regards to trustee B in context c is a vector that has the same dimension as the simple-trust vector. The elements are real numbers in the range* $[0,1]$ *and the sum of all elements is equal to 1.*

If the truster has the same trust-parameter weight policy for all trustees but different for different contexts then we will use the symbol ${}_AW^c$; for same trust-parameter weight policy for all context but different trustees we will use the symbol ${}_AW_B$; finally for same trust-parameter weight policy for all trustees and for all contexts we will use the symbol ${}_AW$. Using this trust-parameter weight policy vector the normalized trust relationship between a truster A and a trustee B at a time t and for a particular context c is given by

$$(A \xrightarrow{c} B)_t^N = {}_A\mathrm{W}_B^c \odot (A \xrightarrow{c} B)_t \tag{3.8}$$

The $\odot$ operator represents the normalization operator. Let $(A \xrightarrow{c} B)_t = [{}_AI_B^c,\ {}_AP_B^c,\ {}_AREP_B^c,\ {}_\Psi REC_B^c]$ be a trust vector such that ${}_AI_B^c,\ {}_AP_B^c,\ {}_AREP_B^c,\ {}_\Psi REC_B^c \in [-1,1] \cup \{\perp\}$. Let also ${}_AW_B^c = [W_I, W_P, W_{REP}, W_{REC}]$ be the corresponding trust-parameter weight policy vector such that $W_I + W_P + W_{REP} + W_{REC} = 1$ and $W_I,\ W_P,\ W_{REP},\ W_{REC} \in [0,1]$. The $\odot$ operator generates the normalized trust relationship as

$$\begin{aligned}
(A \xrightarrow{c} B)_t^N &= {}_AW_B^c \odot (A \xrightarrow{c} B)_t \\
&= [W_I \cdot {}_AI_B^c,\ W_P \cdot {}_AP_B^c,\ W_{REP} \cdot {}_AREP_B^c,\ W_{REC} \cdot {}_\Psi REC_B^c] \\
&= [{}_A\hat{I}_B^c,\ {}_A\hat{P}_B^c,\ {}_AR\hat{E}P_B^c,\ {}_\Psi R\hat{E}C_B^c]
\end{aligned}$$

Each element ${}_A\hat{I}_B^c,\ {}_A\hat{P}_B^c,\ {}_AR\hat{E}P_B^c,\ {}_\Psi R\hat{E}C_B^c$ of the normalized trust vector also lies within $[-1,1] \cup \{\perp\}$.

Example 7 *Continuing with the example, the simple trust relationship between Alice and* S_1 *at time* t *is specified as,* $(Alice \xrightarrow{pp} S_1)_t = [0.009, 0.725, 0.333, 0.423]$. *She decides to put 40% weight on interactions, 30% on properties, 10% weight on reputation and rest 20% on recommendation for all trust relationships. Then her trust-parameter weight policy vector is,* ${}_{Alice}\mathbf{W} = [0.4, 0.3, 0.1, 0.2]$. *Hence the normalized trust vector* $(Alice \xrightarrow{pp} S_1)_t^N$ *is* $[0.4, 0.3, 0.1, 0.2] \odot [0.009, 0.725, 0.333, 0.423] = [0.004, 0.217, 0.033, 0.085]$.

3.3.4 Value of the Normalized Trust Vector

So far we have defined a trust relationship in terms of a vector which is *normalized* by a trust policy. Recall from section 3.3.1.4 that there is at least one scenario in which we need to use a trust value as a weight for a real number, namely for computing recommendations.

Thus it seems appropriate to define the concept of a *value* corresponding to the normalized trust vector. This value indicates the truster's level of positive (or, negative) assurance about the trustee in the specified context in terms of distance from the neutral level i.e., how far the truster is from the neutral level in the positive (negative) side.

Definition 10 [Value of Normalized Trust Vector] *The* value *of a normalized trust relationship* $(A \xrightarrow{c} B)_t^N = [{}_A\hat{I}_B^c,\ {}_A\hat{P}_B^c,\ {}_A\hat{REP}_B^c,\ {}_\Psi\hat{REC}_B^c]$ *is a number in the range* $[-1,1] \cup \{\perp\}$ *and is defined as*

$$\boldsymbol{v}(A \xrightarrow{c} B)_t^N = {}_A\hat{I}_B^c + {}_A\hat{P}_B^c + {}_A\hat{REP}_B^c + {}_\Psi\hat{REC}_B^c \tag{3.9}$$

The value for a trust relationship allows us to revise the terms "trust" and "distrust" as follows:

$$\mathbf{v}(A \xrightarrow{c} B)_t^N = \begin{cases} [-1,0) \Rightarrow \text{ it is } \textbf{distrust} \\ 0 \Rightarrow \text{ it is } \textbf{neutral} \\ (0,1] \Rightarrow \text{ it is } \textbf{trust} \\ \perp \Rightarrow \text{ it is } \textbf{undefined} \end{cases}$$

Example 8 *With our running example, the value of the normalized trust between Alice and* S_1 *in the context pp at time t is given as,* $\boldsymbol{v}(Alice \xrightarrow{pp} S_1)_t^N = 0.004+0.217+0.033+0.085 = 0.339$. *Since the value lies within the range* $(0,1]$, *the "level of trust" of Alice with* S_1 *in the context pp at time t is* 0.339. *The value implies that, Alice's level of assurance with* S_1 *to protect her privacy during online service is* 0.339 *that is, with the given information the truster Alice could reduce her 'neutrality' about* S_1 *by* 0.339 *in the positive side. Alternatively, Alice is only* 33.9% *'positive' about* S_1 *in context pp and* 66.1% *'neutral' about* S_1 *in pp.*

3.3.5 Trust Dynamics

Trust (and distrust) changes over time. Let us assume that we have initially computed a trust relationship $\vec{T_{t_i}}$ at time t_i, based on the values of the underlying parameters at that time. Suppose now that we try to recompute the trust relationship $\vec{T_{t_n}}$ at time t_n. We claim that even if the underlying parameters do not change between times t_i and t_n, the trust relationship will change. To model *trust dynamics* (the change of trust over time) we borrow from observations in the social sciences that indicate that human abilities and skills respond positively to practice, in a learning-by-doing manner, and negatively to non-practice [Hir84]. We observe that the general tendency is to forget about past happenings. This leads us to argue that trust (and distrust) tends towards neutrality as time increases. Initially, the value does not change much; after a certain period the change is more rapid; finally the change becomes more stable as the value approaches the neutral (value = 0) level. How fast trust (or distrust) will decay over time, is, we propose, dependent on the truster's policy. The truster may choose to forget about trust relationships which are 3 weeks old or 5 years old. The model cannot dictate this. Our goal is to provide a basis by which the truster can at least estimate, based on the truster's individual perception about this, the trust at time t_n. We call this *trust dynamics policy,* in which the truster specify two time instants τ_1 and τ_2 along with two percentage thresholds, say p_1 and p_2 for the respective time instants. Rationale is, during the period $[0, \tau_1]$ the initial trust will decay at most to $p_1\%$ of the initial value. During $[\tau_1,\ \tau_2]$ the trust value will decay to at most $p_2\%$ of the initial value and after that the trust value tends toward neutral level (0 value). We assert that the trust decay follows an exponential decay and $\lim\limits_{t \to \infty} \mathbf{v}(\vec{T_t}) = 0$.

Let $\mathbf{v}(\vec{T_{t_i}})$, be the value of a trust relationship, $\vec{T_{t_i}}$, at time t_i and $\mathbf{v}(\vec{T_{t_n}})$ be the decayed value of the same at time t_n. Then the *time-dependent value* of $\vec{T_{t_i}}$ is defined as follows:

Definition 11 [Time-dependent Value of Trust Relationship] *The* time-dependent value *of a trust relationship $\vec{T_{t_i}}$ from time t_i, computed at present time t_n, is given by*

$$v(\vec{T_{t_n}}) = \begin{cases} v(\vec{T_{t_i}}).e^{(\frac{\ln(p_1/100)}{\tau_1})t} & ,\ 0 \le t \le \tau_1 \\ (\frac{p_1}{100}).v(\vec{T_{t_i}}).e^{(\frac{\ln(p_2/p_1)}{\tau_2 - \tau_1})(t-\tau_1)} & ,\ \tau_1 \le t \le \tau_2 \\ (\frac{p_2}{100}).v(\vec{T_{t_i}}).e^{-(t-\tau_2)} & ,\ t \ge \tau_2 \end{cases}$$

where $\tau_1 < \tau_2$ *and* $p_1 > p_2$.

Thus trust dynamics can be represented by the graph shown in Figure 3.2. The values τ_1, τ_2, p_1, p_2 determine the rate of change of trust with time and is assigned by the truster based on its perception about the change. The truster can control the decay by suitable choosing the values of these four parameters. Figure 3.3(a) shows the decay of trust when

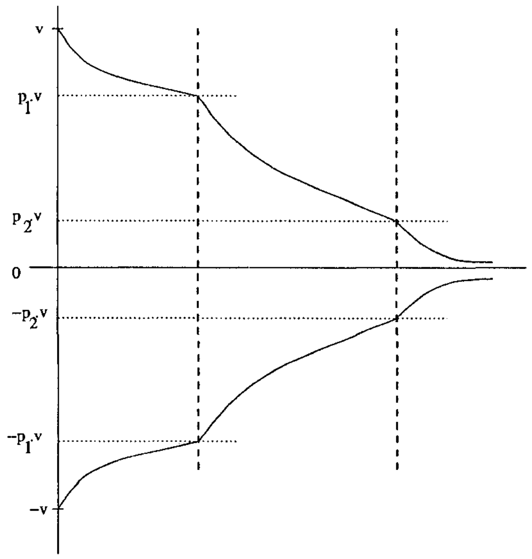
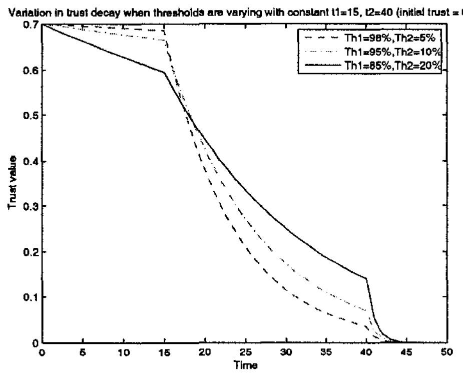
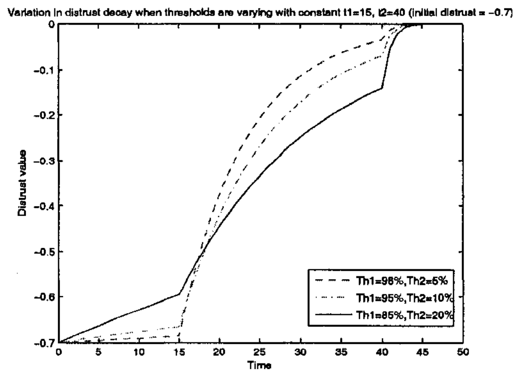


Figure 3.2: Graph showing the nature of trust dynamics

$\tau_1 = 15$ and $\tau_2 = 40$ are kept constant but the percentage thresholds p_1 and p_2 are varied. The initial trust is 0.7 i.e., $\mathbf{v}(\vec{T_{t_0}}) = 0.7$. Similar decay in distrust is shown in the Figure 3.3(b) where $\mathbf{v}(\vec{T_{t_0}}) = -0.7$.



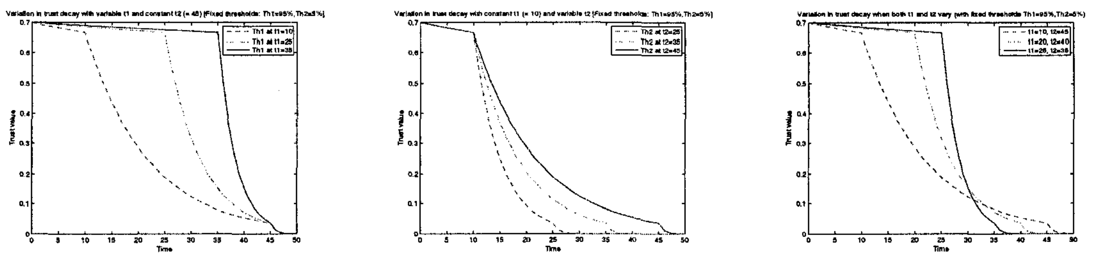
(a) Decay in trust



(b) Decay in distrust

Figure 3.3: Decay in trust (and distrust) for varying percentage threshold at constant time instants

We show the variation in trust decay with fixed percentage thresholds but varying τ_1 only, τ_2 only, and both τ_1, τ_2 in Figure 3.4(a), 3.4(b), and 3.4(c) respectively.



(a) For varying τ_1 (b) For varying τ_2 (c) For varying τ_1 and τ_2

Figure 3.4: Decay in trust for varying time instants with fixed percentage thresholds

Note, number of such time instants and corresponding percentage thresholds need not be restricted to two. The truster, according to his/her policy, can define as many levels as he/she wants. The equation computing the time-dependent value needs to be extended accordingly.

To obtain the trust vector $\vec{T_{t_n}}$ at time t_n, we distribute the value $\mathbf{v}(\vec{T_{t_n}})$ evenly over the components. The rational behind this is that between t_i and t_n we do not have sufficient information to assign different weights to the different components. Thus we have the time-dependent vector as $\vec{T_{t_n}} = [\frac{\mathbf{v}(\vec{T_{t_n}})}{4}, \frac{\mathbf{v}(\vec{T_{t_n}})}{4}, \frac{\mathbf{v}(\vec{T_{t_n}})}{4}, \frac{\mathbf{v}(\vec{T_{t_n}})}{4}]$.

Example 9 *In our example, we assume that the customer Alice has set the dynamics policy as follows: For first 3 months she keeps at least* 95% *of the old trust value; for next 9 months i.e., till 1 year of the previous calculation she keeps* 10% *of the original value, and after 12 months the value gradually approaches zero. Therefore,* $\tau_1 = 3$, $\tau_2 = 12$, $p_1 = 95$, *and* $p_2 = 10$. *Let us also assume that the last time Alice has evaluated* S_1 *is 5 months back and the trust value was* 0.9. *The decayed value of that trust at present time is calculated as:* $v = 0.95 \times 0.9 \times e^{(\frac{\ln 0.1 - \ln 0.95}{12-3})(5-3)} = 0.855 \times (2.7183)^{-0.5} = 0.855 \times 0.6065 = 0.518$, *where we take* $e \approx 2.7183$. *The time-dependent trust vector at current time is obtained as* $[\frac{0.518}{4}, \frac{0.518}{4}, \frac{0.518}{4}, \frac{0.518}{4}] = [0.129, 0.129, 0.129, 0.129]$.

We further believe that trust relationship at present time is not only dependent on the values of the underlying parameters, but also on the "decayed" value of the previous trust. We discuss this in more details in the next section.

3.3.6 Trust Vector at Present Time

The trust of a truster A on a trustee B in a context c at time t_n depends not only on the underlying components of the trust vector but also on the trust established earlier at time t_i. Consider for example that at time t_i, Alice trusts S_1 to the fullest extent (value = 1). At time t_n, Alice re-evaluates the trust relationship and determines the value to be -0.5 (distrust). However, we believe that Alice will lay some importance to the previous trust value and will not distrust S_1 as much as a -0.5 value. So, the normalized trust vector at t_n is a linear combination of time-dependent trust vector and the normalized trust vector calculated at present time. The weight Alice will give to old trust vector and present normalized trust vector is, again, a matter of policy. However, this leads us to refine the expression for normalized trust vector at time t_n as follows. Let $\hat{T}$ be the time-dependent trust vector derived from $\mathbf{v}(\vec{T_{t_i}})$ at time t_n. Also, let α and β be the weights corresponding to present normalized vector and time-dependent vector, respectively.

Definition 12 [**Normalized Trust Relationship**] *The normalized trust relationship between a truster A and a trustee B at time t_n in a particular context c is given by*

$$(A \xrightarrow{c} B)^N_{t_n} = \begin{cases} [{}_A\hat{I}^c_B, {}_A\hat{P}^c_B, {}_AR\hat{E}P^c_B, {}_\Psi R\hat{E}C^c_B], & \text{if } t_n = 0 \\ [\frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}], & \text{if } t_n \neq 0 \text{ and } {}_A\hat{I}^c_B = {}_A\hat{P}^c_B = {}_AR\hat{E}P^c_B = {}_\Psi\hat{R}^c_B = \perp \\ \alpha \cdot [{}_A\hat{I}^c_B, {}_A\hat{P}^c_B, {}_AR\hat{E}P^c_B, {}_\Psi R\hat{E}C^c_B] + \beta \cdot [\frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}], & \text{if } t_n \neq 0 \\ \quad \text{and at least one of } {}_A\hat{I}^c_B,\ {}_A\hat{P}^c_B,\ {}_AR\hat{E}P^c_B,\ {}_\Psi R\hat{E}C^c_B \neq \perp & \end{cases}$$

where $\alpha \cdot [{}_A\hat{I}^c_B, {}_A\hat{P}^c_B, {}_AR\hat{E}P^c_B, {}_\Psi R\hat{E}C^c_B] + \beta \cdot [\frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}, \frac{v(\hat{T})}{4}] = [\alpha \cdot {}_A\hat{I}^c_B + \beta \cdot \frac{v(\hat{T})}{4},\ \alpha \cdot {}_A\hat{P}^c_B + \beta \cdot \frac{v(\hat{T})}{4},\ \alpha \cdot {}_AR\hat{E}P^c_B + \beta \cdot \frac{v(\hat{T})}{4},\ \alpha \cdot {}_\Psi R\hat{E}C^c_B + \beta \cdot \frac{v(\hat{T})}{4}]$, $\alpha, \beta \in [0,1]$ *and* $\alpha + \beta = 1$.

Example 10 *In our running example, let Alice put* 70% *weight on the current vector and* 30% *on the time-decayed vector. Then her final trust vector for the service provider S_1 is evaluated as:*

$$\begin{aligned} (Alice \xrightarrow{pp} S_1)^N_{t_n} &= 0.7 \times [0.004, 0.217, 0.033, 0.085] + 0.3 \times [0.129, 0.129, 0.129, 0.129] \\ &= [0.042, 0.191, 0.062, 0.098] \end{aligned}$$

and the final trust value is obtained as $v(Alice \xrightarrow{pp} S_1)^N_{t_n} = 0.393$.

3.4 Comparison Operation on Trust Vectors

We are now in a position to determine the relative trustworthiness of two trustees. The need for such comparison occurs in many real life scenarios. Consider the following example. Suppose entity A gets two conflicting pieces of information from two different sources B and C. In this case A will probably want to compare its trust relationships with entities B and C and accept the information that originated from the "more" trustworthy entity. This motivates us to define a comparison operator on trust relationships.

Let T and T' be two normalized trust relationships at time t. We introduce the following notion of compatibility between two trust relationships.

Definition 13 [Compatibility of Trust Relationships] *Two trust relationships, T and T' are said to be* compatible *if the trust relationships have been defined under the same trust-parameter weight policy vector, the trust relationships are at the same time instances, and the context c for the trust relationship T is the same as the context c' for T', that is $c = c'$. Otherwise the two trust relationships are said to be* incompatible.

Note that the definition of compatibility between two trust relationships does not explicitly involve information about the trusters or trustees. The most intuitive way to compare two trust relationships T and T' is to compare the values of the trust relationships in a numerical manner. Thus for A to determine the relative levels of trustworthiness of B and C, A evaluates $\mathbf{v}(A \xrightarrow{c} B)_t^N$ and $\mathrm{v}(A \xrightarrow{c} C)_t^N$. If $\mathbf{v}(A \xrightarrow{c} B)_t^N > \mathbf{v}(A \xrightarrow{c} C)_t^N$, then A trusts B more than C in the context c. We say that T *dominates* T', given by $T \succ T'$. However, if $\mathbf{v}(A \xrightarrow{c} B)_t^N = \mathbf{v}(A \xrightarrow{c} C)_t^N$, A cannot judge the relative trustworthiness of B and C. This is because there can be two vectors whose individual component values are different but their scalar values are the same. For such cases we need to compare the individual elements of the two trust relationships to determine the relative degree of trustworthiness. In addition, for the same reasons, it is better to determine relative trustworthiness of B and C on the basis of component values rather than breaking the tie arbitrarily.

Let $(A \xrightarrow{c} B)_t^N = [{}_A\hat{I}_B^c,\ {}_A\hat{P}_B^c,\ {}_A\hat{REP}_B^c,\ {}_\Psi\hat{REC}_B^c]$ and $(A \xrightarrow{c} C)_t^N = [{}_A\hat{I}_C^c,\ {}_A\hat{P}_C^c,\ {}_A\hat{REP}_C^c,\ {}_\Psi\hat{REC}_C^c]$ such that $\mathbf{v}(A \xrightarrow{c} B)_t^N = \mathbf{v}(A \xrightarrow{c} C)_t^N$. Let also the underlying trust-parameter weight policy vector be given by ${}_{A}W = (w_1, w_2, w_3, w_4)$ where $\sum_{i=1}^{4} w_i = 1$ and $w_i \in [0, 1]$, $\forall i = 1, \ldots, 4$. To determine the dominance relation between T and T' we first determine the *ordered* trust relationships $\bar{T}$ and $\bar{T}'$ corresponding to T and T'.

Definition 14 **[Ordered Trust Relationship]** *The* ordered *trust relationship $\bar{T}$ is generated from a normalized trust relationship T as follows:*

1. *Order the w_i's in the trust-parameter weight policy vector corresponding to T in descending order of magnitude.*
2. *Sort the components of the trust vector T according to the corresponding weight components.*

We compare the two ordered trust relationships $\bar{T}$ and $\bar{T}'$, corresponding to T and T', component-wise to determine the dominance relation between the two. Note that we assume that the same underlying trust-parameter weight policy vector has been used to determine the trust relationships. If the first component of $\bar{T}$ is numerically greater than the first component of $\bar{T}'$ then $T \succ T'$. Else if the first components are equal then compare the second components. If the second component of $\bar{T}$ is greater than the second component of $\bar{T}'$ then $T \succ T'$, and so on. If weights are equal for first three (or, all four) components in the ordered trust relationships, then $T \succ T'$ only when the three components (or, all four components) of T are numerically greater than those of T'. In the comparison process we assume that the value $\perp$ is dominated by all real numbers. If we cannot conclude a dominance relation between the two trust relationship, then we say that the two trust relationships are *incomparable.* This is formalized by the following definition.

Definition 15 **[Dominance between Trust Relationships]** *Let T and T' be two trust relationships and $\bar{T}$ and $\bar{T}'$ be the corresponding ordered trust relationships. Let also $\bar{T}_i$ and $\bar{T}'_i$ represent the i^{th} component of each ordered trust relationships and w_i represent the i^{th} weight component in the corresponding trust-parameter weight policy vector. T is said to dominate T' if any one of the following holds:*

1. *$v(T) > v(T')$; or*

2. *if* $\forall\, i, j,\ i \neq j,\ (w_i = w_j)$ *then* $\forall\, i,\ \bar{T}_i > \bar{T}'_i$; *or*

3. *if* $\exists\, i, \bar{T}_i > \bar{T}'_i$ *and for* $k = 0 \ldots (i-1),\ \bar{T}_{i-k} \not< \bar{T}'_{i-k}$

Otherwise T *is said to be* incomparable *with* T'.

Algorithm 1 describes the comparison of the trust relationships T and T'.

Example 11 *Continuing with our e-commerce example, let the customer Alice evaluate the normalized final trust vector with the other service provider* S_2 *at the same time as,* $(Alice \xrightarrow{pp} S_2)_t^N$ $=$ $[0.042,\ 0.191,\ 0.078,\ 0.083]$. *Now she wants to decide which service provider is more trustworthy in the context of protecting her privacy. Here she can not decide simply from the final trust values as, we can see from the above vector,* $v(Alice \xrightarrow{pp} S_2)_t^N = 0.393 = (Alice \xrightarrow{pp} S_1)_t^N$. *However* S_1 *has high recommendation than* S_2 *though* S_2*'s* reputation *is higher than that of* S_1. *Therefore Alice applies the method described above to compare the trustworthiness of the service providers. Her trust-parameter weight policy vector was* $[0.4, 0.3, 0.1, 0.2]$ *which, in descending order of magnitude, results in* $[0.4, 0.3, 0.2, 0.1]$. *Consequently, the two ordered trust vectors are*

$$\begin{aligned} ordered\ (Alice \xrightarrow{pp} S_1)_t^N &= [0.042, 0.191, 0.098, 0.062] \\ ordered\ (Alice \xrightarrow{pp} S_2)_t^N &= [0.042, 0.191, 0.082, 0.078] \end{aligned}$$

This shows that the service providers have equal values for $\bar{w}_1$ *and* $\bar{w}_2$ *where* ${}_A\vec{W} = [\bar{w}_i]$. *So Alice cannot make any decision based on* $\bar{w}_1$ *and* $\bar{w}_2$ *i.e., interaction or properties. However* $S_1 > S_2$ *with respect to* $\bar{w}_3$ *which is 'recommendation'. Hence she concludes that the service provider* S_1 *is more trustworthy than* S_2 *in the context of pp i.e., "reliability to access data and provide a privacy protection service".*

3.5 Combination Operation on Trust Vectors

We have defined a basic trust relationship as a binary relation between two different entities – a truster and a trustee. However, today's world of information exchange involves many cooperative entities in a relationship within a specified context. Combination of trust is

Algorithm 1 Trust comparison algorithm

```
Input: Two trust vectors $T = (t_i)$ and $T' = (t_i')$, and the trust-parameter weight policy
vector $W = (w_i)$.  /* $W$ is same for $T$ and $T'$ */
Output: Comparison result for $T$ and $T'$

Procedure $CompareTrust(T, T', W)$
if $v(T) > v(T')$ then
  return $T \succ T'$
else if $v(T) < v(T')$ then
  return $T' \succ T$
else
  /* when $v(t) = v(T')$ */
  if for all $i$, $t_i = t_i'$ then
    return $T = T'$
  else
    Order the $w_i$ s in the descending order of magnitude. The new vector is $\bar{W} = (\bar{w}_i)$.
    Sort the components of $t_i$s of $T$ according to $\bar{w}_i$. The new vector is $\bar{T} = (\bar{t_i})$.
    Sort the components of $t_i'$s of $T'$ according to $\bar{w}_i$. The new vector is $\bar{T'} = (\bar{t_i}')$.
    for all $i, j, i \neq j$ and $\bar{w}_i = \bar{w}_j$ do
      if for all such $i$, $\bar{t_i} > \bar{t_i}'$ then
        return $T \succ T'$
      else if for all such $i$, $\bar{t_i} < \bar{t_i}'$ then
        return $T' \succ T$
      else if for all such $i$, $\bar{t_i} = \bar{t_i}'$ then
        if $\exists$ $i$, such that $\bar{t_i} > \bar{t_i}'$ and $\forall$ $k = 0 \ldots (i-1), \bar{t}_{i-k} = \bar{t}_{i-k}'$ then
          return $T \succ T'$
        else if $\exists$ $i$, such that $\bar{t_i} < \bar{t_i}'$ and $\forall$ $k = 0 \ldots (i-1), \bar{t}_{i-k} = \bar{t}_{i-k}'$ then
          return $T' \succ T$
        else
          return "$T$ and $T'$ are incomparable"
        end if
      else
        return "$T$ and $T'$ are incomparable"
      end if
    end for
  end if
end if
```

needed for the interoperability of these cooperating agents. Whenever a group of agents are working together, combination of their individual trust relationship is necessary to have an idea about the expected behavior of the group. Keeping this in mind we now formalize combination operators for trust relationships. Different possibilities like one-to-many, many-to-one, and many-to-many relationships are addressed. We also formalize the effect of reconfiguration of these groups on the corresponding trust relationships.

3.5.1 Trust Relationship between a Truster and a Group of Trustees

In real life, we often encounter situations where we have to take decisions based on information coming from different sources. Consider the scenario where an entity has existing trust relationships with different service providers for a particular service. The truster expects some service which is provided collectively by the service providers. The truster has some expectation from each individual provider. To have an idea about the service provided by the group, the combined trust of the service providers needs to be estimated. Therefore, the receiver needs a mechanism to combine the existing trust relationships to estimate an initial *composite trust relationship*. The group of service providers is considered as a single entity (trustee). Once the combination is done, the truster no longer considers the trust relationships with individual trustee. The truster begins with the combined group as a single entity and subsequently a trust relationship with the group evolves.

Let an entity A have trust relationships with two different entities B and C in the same context c at time t. A decides to have a trust relationship with the combined group BC in the same context. It is clear that individual trust relationships of both B and C will have effect on the resulting trust vector. However, the individual trust relationships will have different degrees of effect. This is represented by A putting different weights on the trustees to evaluate their relative importance in the trustee group. Once the group is formed the trust $(A \xrightarrow{c} BC)_t^N$ evolves as a new trust relationship. Thus we define the *initial* trust relationship between A and BC in context c as follows.

Definition 16 [**Trust Combination**] *Let at time t a truster A have two trust relationships, $(A \xrightarrow{c} B)_t^N$ and $(A \xrightarrow{c} C)_t^N$ with trustees B and C respectively. If $\oplus$ is the* trust combination operator *then the trust relationship between A and the group BC is defined as* $(A \xrightarrow{c} BC)_t^N = (A \xrightarrow{c} B)_t^N \oplus (A \xrightarrow{c} C)_t^N$.

The trust combination operator $\oplus$ depends on the semantics of trust combination in a specific application. It can be specified as, but not limited to, *max, min, multiplication* of two trust relationships. That is, we can have

$$\begin{aligned}
(A \xrightarrow{c} B)_t^N \oplus (A \xrightarrow{c} C)_t^N &= max(\ (A \xrightarrow{c} B)_t^N, (A \xrightarrow{c} C)_t^N) \\
(A \xrightarrow{c} B)_t^N \oplus (A \xrightarrow{c} C)_t^N &= min(\ (A \xrightarrow{c} B)_t^N, (A \xrightarrow{c} C)_t^N) \\
(A \xrightarrow{c} B)_t^N \oplus (A \xrightarrow{c} C)_t^N &= (A \xrightarrow{c} B)_t^N \times (A \xrightarrow{c} C)_t^N
\end{aligned}$$

where the '$\times$' is defined as component-wise multiplication of two vectors or some variation of that. However, we prefer to specify the $\oplus$ operator as the 'weighted sum of corresponding components' where for each component, the corresponding weights add up to 1. That is, for each component of the combined trust vector $(A \xrightarrow{c} BC)_t^N$, the operator $\oplus$ takes a weighted sum of respective components of $(A \xrightarrow{c} B)_t^N$ and $(A \xrightarrow{c} C)_t^N$. Thus, if $(A \xrightarrow{c} B)_t^N = ({}_A\hat{I}_B^c,\ {}_A\hat{P}_B^c,\ {}_A\hat{REP}_B^c,\ {}_{\Psi_B}\hat{REC}_B^c)$, $(A \xrightarrow{c} C)_t^N = ({}_A\hat{I}_C^c,\ {}_A\hat{P}_C^c, {}_A\hat{REP}_C^c,\ {}_{\Psi_C}\hat{REC}_C^c)$ be the trust vectors, then the combined vector is given by $(A \xrightarrow{c} BC)_t^N = ({}_A\hat{I}_{BC}^c,\ {}_A\hat{P}_{BC}^c,$ ${}_A\hat{REP}_{BC}^c,\ {}_{\Psi_{BC}}\hat{REC}_{BC})$ where,

$$\begin{aligned}
{}_A\hat{I}_{BC}^c &= w_B^I \cdot {}_A\hat{I}_B^c + w_C^I \cdot {}_A\hat{I}_C^c, & {}_A\hat{REP}_{BC}^c &= w_B^{REP} \cdot {}_A\hat{REP}_B^c + w_C^{REP} \cdot {}_A\hat{REP}_C^c, \\
{}_A\hat{P}_{BC}^c &= w_B^P \cdot {}_A\hat{P}_B^c + w_C^P \cdot {}_A\hat{P}_C^c, & {}_{\Psi_{BC}}\hat{REC}_{BC}^c &= w_B^{REC} \cdot {}_{\Psi_B}\hat{REC}_B^c + w_C^{REC} \cdot {}_{\Psi_C}\hat{REC}_C^c
\end{aligned}$$

Here, $w_B^{comp} + w_C^{comp} = 1, \forall comp \in \{I, P, REP, REC\}$. The weights w_i^{comp} is weight assigned to i^{th} trustee for the component *comp* and $w_i^{comp} \in [0,1], \forall i,\ \forall comp$. The above way of combining two trust relationships is generic. Because the other possible ways of combining trust relationships can be expressed as some variations of this method by suitably adjusting the weights. Consequently this way of combining trust is semantic-independent.

Note that, A has two groups of recommenders Ψ_B and Ψ_C for B and C respectively. There are five relations possible for these two groups, namely

1. $\Psi_B = \Psi_C$,

2. $\Psi_B \cap \Psi_C = \emptyset$,

3. $\Psi_B \subset \Psi_C$,

4. $\Psi_B \supset \Psi_C$, and

5. $\Psi_B \cap \Psi_C \neq \emptyset$ and none of 1, 3, 4 hold.

The truster A forms a new list of recommender Ψ_{BC} for the combined group BC where, $\Psi_{BC} = \Psi_B \cup \Psi_C$, irrespective of the above five relations between Ψ_B and Ψ_C. If the truster has m trust relationships with trustees $B_1, B_2, \ldots, B_m$, we can easily generalize the above concept for the group of trustees $\mathcal{G} = \{B_1, B_2, \ldots, B_m\}$ as $(A \xrightarrow{c} \mathcal{G})_t^N = (A \xrightarrow{c} B_1)_t^N \oplus (A \xrightarrow{c} B_2)_t^N \ldots \oplus (A \xrightarrow{c} B_m)_t^N$. The operator $\oplus$ takes the weighted sum of the corresponding components of the vectors.

Example 12 *Consider in the example that both the service provider declare a collaboration between them. That is, they together might process some order from the customers. In this scenario, Alice wants to check how much she can trust this combined provider to protect her personal privacy during online transactions. Let she assign* 50% *weight to both interactions and properties of both providers. That is,* $w^I_{S_1} = w^I_{S_2} = 0.5$ *and* $w^P_{S_1} = w^P_{S_2} = 0.5$. *She also assigns* 40% *and* 60% *respectively to* S_1 *and* S_2*'s reputation and* 60% *and* 40% *to the service providers recommendations respectively. We also assume that Alice use the same set of recommender for both the providers i.e.,* $\Psi_{S_1} = \Psi_{S_2}$. *Therefore Alice's trust components on the combined trustee service provider, say* S, *have following values,*

$$_{Alice}\hat{I}^{pp}_S = 0.5 \times 0.042 + 0.5 \times 0.042 = 0.042$$

$$_{Alice}\hat{P}^{pp}_S = 0.5 \times 0.191 + 0.5 \times 0.191 = 0.191$$

$$_{Alice}\hat{REP}^{pp}_S = 0.4 \times 0.062 + 0.6 \times 0.078 = 0.072$$

$$_{\Psi_S}\hat{REC}^{pp}_S = 0.6 \times 0.098 + 0.4 \times 0.082 = 0.092$$

where $\Psi_S = \Psi_{S_1}$. *Consequently, Alice's trust on the combined Web service provider* S *is represented as* $[0.042, 0.191, 0.072, 0.092]$ *which has the value* $v(Alice \xrightarrow{pp} S)_t^N = 0.397$.

3.5.2 Trust Relationship between a Group of Trusters and a Single Trustee

Next, we address the situation where different trusters having different trust relationships with a particular entity in a context, form a group. After forming a group the trusters work as a single truster entity. We need to define a way to combine these different trust relationships to get the initial trust for the group. This initial trust gives the starting point of a trust relationship between two entities (a group and a single trustee). Thereafter, this trust evolves as before. But before grouping, different trusters have their own policy to evaluate the trustee for the same context. In other words, though trust context is same, there are different trust policies. Unless all the trusters agree to a common policy, as well as a common criteria for evaluation, there cannot be a single trust relationship. To achieve this, there should be a *consensus* among the members.

At this stage we need to discuss some issues. Let an entity A and B have trust vectors about an entity C in some context c at time t. Now A and B want to collaborate and work as a single truster. The initial trust for the group in the context c is derived from their individual relationship with C. Let A have higher interactions and properties values than B in terms of trust relationship with C. But B has stronger recommendations about C. Therefore, for initial group trust, for interactions and properties, A will play the major role determining those. The recommendation component of the initial trust of the group will have more influence of recommendation value of B's trust relationship with C. We assume that both the trusters have same value for reputation of C. So, for each component of the trust vector, the group of trusters has an ordering according to their individual contribution for the component. For each component, they have to assign weights to each individual truster in the group according to their relative ordering. How the weights would be assigned is determined by the consensus the group arrives at during the time of group formation. After that, the parameters are evaluated for the whole group as a single entity.

Definition 17 **[Trust Consensus]** *The* trust-consensus *of a group of trusters is defined as the agreement among all members to build a common basis for evaluating a combined trust relationship.*

Let $A_1, A_2, \ldots, A_m$ be m trusters trying to form a group say $\mathcal{G}$, to build a single trust relationship with a trustee B in some common context c. All these trusters have different trust relationships with B in context c at the present time t. So there are m existing trust relationships $(A_1 \xrightarrow{c} B)_t^N, (A_2 \xrightarrow{c} B)_t^N, \ldots, (A_m \xrightarrow{c} B)_t^N$ at t. The objective is to get a trust relationship $(\mathcal{G} \xrightarrow{c} B)_t^N$ where $\mathcal{G} = \{A_1, A_2, \ldots, A_m\}$.

The members need to agree to the following things before formation of the group:

1. For each component, a set of weights to assign relative importance of the members.
2. A common trust-parameter weight policy vector to assign weights to each parameter of combined trust relationship.
3. A common interval length to determine interactions, as well as trust.
4. A common policy to assign weights to trustee properties.
5. A common policy to evaluate reputation parameter.
6. A common set of recommenders whom the group consider for providing recommendation about the trustee, and
7. A common policy to evaluate trust relationship with a recommender.

For the 6th point above, let $\Psi_1, \ldots, \Psi_m$ be m group of recommenders who have provided recommendation for B in context c to the truster $A_1, \ldots, A_m$ respectively. Now the group $\mathcal{G}$ of trusters forms a new group Ψ' of recommenders, where $\Psi' = \bigcup_{i=1}^{m} \Psi_i$. Let $|\Psi'| = k$ (i.e., there are k distinct recommenders in the group Ψ). Each of the A_i's may not have a trust relationship with all of these k recommenders (when $\Psi_p \cap \Psi_q = \emptyset,\ p \neq q$ i.e., the two groups of recommenders are disjoint). In such cases the group $\mathcal{G}$ evaluates trust relationship according to their newly formed policy for establishing trust with a recommender.

Therefore, for the group's trust relationship with trustee B, we have $(\mathcal{G} \xrightarrow{c} B)_t^N = [_{\mathcal{G}}\hat{I}_B^c,\ _{\mathcal{G}}\hat{P}_B^c,\ _{\mathcal{G}}\hat{REP}_B^c,\ _{\Psi'}\hat{REC}_B^c]$ where,

$$_{\mathcal{G}}\hat{I}_B^c = \sum_{i=1}^{m} w_i^I \cdot {_{A_i}\hat{I}_B^c}, \qquad _{\mathcal{G}}\hat{REP}_B^c = \sum_{i=1}^{m} w_i^{REP} \cdot {_{A_i}\hat{REP}_B^c},$$
$$_{\mathcal{G}}\hat{P}_B^c = \sum_{i=1}^{m} w_i^P \cdot {_{A_i}\hat{P}_B^c}, \qquad _{\Psi'}\hat{REC}_B^c = \sum_{i=1}^{m} w_i^{REC} \cdot {_{A_i}\hat{REC}_B^c}$$

Here, $w_i^{comp} \in [0,1]$ and $\sum_{i=1}^{m} w_i^{comp} = 1, \forall comp \in \{I, P, REP, REC\}$. After arriving at a *trust-consensus*, group $\mathcal{G}$ works as a single entity to work further with the trustee B according to their trust-consensus.

Example 13 *In the example, let us assume that Alice consult her friend Bob before making any further interactions with a service provider, say S_1. That is, she wants herself and Bob to act as a single entity and wants to take decisions together. Here we assume that Bob has an existing trust relationship with S_1 in the context pp. This trust relationship was established during some previous online interactions made by Bob alone. Also Alice wants the cost of the air-ticket to be paid by Bob whereas she would pay for the hotel. In this scenario both of them need to disclose their personal sensitive information (e.g., credit card details). To achieve this they form a new customer identity where they together act as a single customer. Note, Alice and Bob could have availed the different services separately without establishing a combined truster entity. However, in that case they might end up with two conflicting decisions like Alice deciding to avail the hotel reservation service from S_1 and Bob, unaware of S_2, deciding not to make any purchase from S_1 considering S_1 not enough trustworthy in the context pp. To avoid this conflict they form a truster group where every decision has same effect on each of them. Let they form the entity* `AliceBob` *and make the consensus as follows: $w_{Alice}^I = 0.7$, $w_{Bob}^I = 0.3$; $w_{Alice}^P = 0.5 = w_{Bob}^P$; $w_{Alice}^{REP} = 0.45$, $w_{Bob}^{REP} = 0.55$, and $w_{Alice}^{REC} = 0.4$, $w_{Bob}^{REC} = 0.6$. Also $\Psi_{AliceBob} = \Psi_{Alice} \cup \Psi_{Bob}$ where $\Psi_{Alice} \cap \Psi_{Bob} = \emptyset$. Alice and Bob agree to use Alice's policy to assign weights to attributes of S_1. They adapt Bob's trust policy to evaluate trustworthiness of a recommender in the context 'providing recommendation'. We also assume that they used same interval length, which is 1 month, to evaluating interactions and trust. Finally their combined trust-*

parameter weight policy vector is $[0.4, 0.4, 0.1, 0.1]$. *Now, let Bob's trust vector at that time with* S_1 *in pp was* $[0.021, 0.184, 0.073, 0.121]$. *The initial values of the components are,*

$$
\begin{aligned}
{}_{AliceBob}I_{S_1}^{pp} &= 0.7 \times 0.042 + 0.3 \times 0.021 = 0.036 \\
{}_{AliceBob}P_{S_1}^{pp} &= 0.5 \times 0.191 + 0.5 \times 0.184 = 0.187 \\
{}_{AliceBob}REP_{S_1}^{pp} &= 0.45 \times 0.062 + 0.55 \times 0.073 = 0.068 \\
\Psi_{AliceBob}REC_{S_1}^{pp} &= 0.4 \times 0.098 + 0.6 \times 0.121 = 0.112
\end{aligned}
$$

and the initial normalized trust vector is given by $(AliceBob \xrightarrow{pp} S_1)_t^N = [0.014, 0.075, 0.007, 0.011]$. *Consequently, Alice and Bob together trust the web service provider* S_1 *to the degree of* 0.107. *Alice and Bob, as a single entity, starts interacting with* S_1 *with this initial trust value and updates this trust relationship in future according to their agreed upon common policy.*

3.5.3 Trust Relationship between a Group of Trusters and a Group of Trustees

We now explore the situation when a group of trusters $\mathcal{G}_r$ forms a trust relationship with a group of trustees $\mathcal{G}_e$ in some common context c. Though this is a complicated concept, we can formalize this by combining the above two cases. Combination can take place in different ways.

1. If the group of trustees $\mathcal{G}_e$ already exists, then each truster A_i must already have, or must build a trust relationship $(A_i \xrightarrow{c} \mathcal{G}_e)_t^N$ as described in Section 3.5.1. Then A_i's form the truster group $\mathcal{G}_r$ with $\mathcal{G}_e$, considering $\mathcal{G}_e$ as a single trustee, as described in Section 3.5.2.
2. If the truster group $\mathcal{G}_r$ already exists with m different trust relationships like $(\mathcal{G}_r \xrightarrow{c} B_i)_t^N$ for $i = 1, 2, \ldots, m$, then $(\mathcal{G}_r \xrightarrow{c} \mathcal{G}_e)_t^N$ can be formed as in section 3.5.1.
3. If neither group of trusters or trustees exist, either of the group has to be formed first and then the other group is formed as explained above.

We have defined combination operations for one truster-many trustees, many trusters-one trustee and many trusters-many trustees. The group is formed under a common trust policy. Next we examine the effect of reconfiguration of a group on the trust relationship.

3.5.4 Reconfiguration of a Group

After the group is formed, some member may leave, or some new member may join the group. This contraction (or, expansion) of the group can happen in steps or, in one instance. That is, old members can leave one by one or, together. Similarly, new members can join in subsequent time instances, or as a whole group. We now address the issue of reconfiguration of group of trusters (or, trustees) over time, and examine its effect on the existing trust relationship.

3.5.4.1 Reconfiguration of a Trustee Group

Let at time t_n, there be a trust relationship $(A \xrightarrow{c} \mathcal{G})_{t_n}^{N}$ between a truster A and a group of trustees $\mathcal{G}$, where $\mathcal{G} = \{B_1, B_2, \ldots, B_m\}$. Now, let at t_{n+1}, a new trustee B_{m+1} join the group $\mathcal{G}$. Then to build the new trust relationship (rather, we say to "reconfigure" the existing trust relationship) $(A \xrightarrow{c} \mathcal{G}')_{t_{n+1}}^{N}$ where $\mathcal{G}' = \mathcal{G} \cup \{B_{m+1}\}$, A reassigns the weights for each component for $\mathcal{G}$ and B_{m+1}, according to his trust policy without violating the existing conditions. A does not combine B_{m+1} with existing m trustees, rather he combines two entities $\mathcal{G}$ and B_{m+1}, treating $\mathcal{G}$ as a single entity. That is, in case of $(A \xrightarrow{c} \mathcal{G}')_{t_{n+1}}^{N}$, the weights $w_{\mathcal{G}}^{comp}, w_{B_{m+1}}^{comp} \in [0,1]$ and $w_{\mathcal{G}}^{comp} + w_{B_{m+1}}^{comp} = 1$ where, comp $\in \{I, P, REP, REC\}$. The truster A also needs to update the recommender list by adding the recommenders involved in the trust relationship $(A \xrightarrow{c} B_{m+1})_{t_{n+1}}^{N}$.

We now consider the situation where at t_{n+1} a trustee B_i leaves the group. That is, $\mathcal{G}' = \mathcal{G} \setminus \{B_i\}$ for some $i \in \{1, 2, \ldots, m\}$. Then the truster needs not to change the policy to evaluate the trust relationship. Because, after the trustee group is formed, truster considers the trustee group as a single entity. This trust relationship has evolved over time and removal of a trustee does not change the truster's policy of trust evaluation. The trust relationship will evolve further with the reduced trustee group. Impact of absence of a trustee on the trust relationship is noticed accordingly.

The above ideas can easily be extended if a group of new trustees (i.e., more than one new trustee) join (or, leave) the existing group of trustees at a time. If a new group of trusters $\mathcal{G}''$ joins, then $\mathcal{G}' = \mathcal{G} \cup \mathcal{G}''$ where $\mathcal{G}'' = \{B_{m+1}, \ldots, B_n\}$ and $n > m$. The recommender's list is also updated accordingly. If a subgroup $\mathcal{G}''$ of trustees leaves the group $\mathcal{G}$ ($\mathcal{G}'' \subset \mathcal{G}$), then $\mathcal{G}' = \mathcal{G} \setminus \mathcal{G}''$.

3.5.4.2 Reconfiguration of a Truster Group

Let at time t_n, there be a trust relationship $(\mathcal{G} \xrightarrow{c} B)_{t_n}^{N}$ between a group of trusters $\mathcal{G}$ and a trustee B, where $\mathcal{G} = \{A_1, A_2, \ldots, A_m\}$. Now, let at t_{n+1}, a new truster A_{m+1} joins the group $\mathcal{G}$. The new group $\mathcal{G}' = \mathcal{G} \cup \{A_{m+1}\}$ builds a new trust relationship $(\mathcal{G}' \xrightarrow{c} B)_{t_{n+1}}^{N}$ with B. Since $\mathcal{G}$ is a group that has been formed earlier, at t_{n+1} we no longer consider individual component values for all truster. The reason is, whenever a group is formed, at the time of formation, the members are ranked according to their relative importance for each component. After formation each member works in the same way as other with the trustee. So, after formation there is no discrimination among the existing group members. A trust-consensus is made between the two truster entities, $\mathcal{G}$ and A_{m+1}. The component values of the new truster A_{m+1} is, therefore, compared to the corresponding component values of the group $\mathcal{G}$. A_{m+1} may be a newcomer in the field (with less experience and knowledge) or may be senior enough to get more importance than the formed group $\mathcal{G}$, when he is about to join $\mathcal{G}$. In the latter case, in $(\mathcal{G}' \xrightarrow{c} B)_{t_{n+1}}^{N}$, A_{m+1} will have higher weights for the components in which he dominates. The agreement between the joining member and the group determines the relative importance of the two entities in that trust relationship.

This idea is easily extended to the situation where at t_{n+1} more than one new truster join $\mathcal{G}$. In that case, $\mathcal{G}' = \mathcal{G} \cup \mathcal{G}''$ where $\mathcal{G}'' = \{A_{m+1}, \ldots, A_n\}$ and $n > m$. In this case ordering is done for each of the component values of $\mathcal{G}$, and $A_{m+1}, \ldots, A_n$. The trust-consensus is made accordingly.

Removal of a truster from the group does not affect the group trust-consensus. The group continues its trust relationship with the trustee as earlier. The trust evolves over time as before. Absence of a member does not affect the trust relationship as long as the trustee group remains unaltered. It is also true if more than one member leave the group at a time. Changing of group policy, or arriving at a new trust-consensus depends totally on the group. If a significant number of members have left the consortium, the existing members may go for a revised agreement on how to evaluate the trust relationship thereafter. Suppose at time t_n we have a trust relationship $(\mathcal{G} \xrightarrow{c} B)_{t_n}^{N}$. Let us assume that at each subsequent time intervals one or more members leave the group. Then there is a sequence of time $\{t_k\}$ after the time t_n such that at t_{n+k} only one member from the group remains in $\mathcal{G}$. Then at t_{n+k}, the solitary member can go with the existing trust policy and scheme to evaluate trust, or he can establish a new one-to-one trust relationship with the trustee.

3.6 Summary

This chapter presents one of the major contributions of this work – the vector trust model. The model helps to express different states (trust, distrust, neutral, and unknown) of a trust relationship as quantitatively measurable objects. This also allows us to have multiple degrees (potentially infinite) of trust and distrust. One of the key advantages of the model, unlike existing trust models, is its extensibility. Trust is evaluated by numerically evaluating four independent parameters – *interaction, properties, reputation*, and *recommendation*. Independence of the parameters provides the option to extend the model in future. For example, if a fifth parameter is needed to be incorporated into the trust model, it can be done easily without changing any of the constructs and methods to evaluate the existing parameters, provided the fifth parameter is independent to the existing four parameters. This independence of parameters also helps to evaluate trust even when information regarding all the parameters are not available. None of the existing trust models allow these flexibilities. Another key feature of the model is the relative importance of the trust parameters. It helps to use the model in a security context where one (or more than one) of the parameters is (are) not applicable. In such scenarios the effect of the parameter(s) on the final trust value can be eliminated by assigning a 0 importance (weight) to the parameter(s). The other key features of the model are capturing dependence of trust on time as well as old trust, method of comparison and method of combination of trust relationships. No existing model has formalized dependence of trust on time as it is done here. Also the existing models do not explicitly formalized flexible methods of comparison and combination of trust. Therefore, this vector trust model provides features that are not available from any single trust model in literature.

Chapter 4

Reasoning about Trust Relationships in Different Contexts

The vector trust model developed so far can reason about trust relationships only with respect to a given context. In other words, it allows trust vectors to be compared only when there is an exact match on the context. For this to happen the contexts needs to be specified using exactly the same terms. This assumption is not realistic in most situations. It is extremely unlikely that different trusters will specify a given context in exactly the same manner. This prevents our model from being interoperable. However, it appears that a model providing such features will be useful. For example, let a user A (the truster) trust a software developer B (the trustee) to a degree T to produce excellent quality anti-virus software (the context). Assuming that expertise to develop an anti-virus software (a related context) is similar to the expertise needed to develop anti-spam software, it seems natural that the truster A will be able to determine how much to trust B for the different (but related) context. We introduce this feature in the model by formalizing a notion of context and the relationships that exist between different contexts.

We observe that we must define contexts in such a manner that makes our model interoperable. Different entities often use different words to describe the same context. Alternately, the same word can be used for describing different contexts. These are example of semantic conflicts in the use of terminology. To solve these problems we borrow some ideas from the work on ontologies [Gru93, UG96]. Our ontology consists of a set of contexts together with relationships defined among them. We have presented this context ontology in [RRC08]. We begin by giving a formal definition of context and later describe the relationships between contexts.

4.1 Context Ontology

Definition 18 **[Context]** *A* context *$\mathcal{C}$ is represented by a set of keywords that is denoted by $KeywordSet_{\mathcal{C}}$.*

Each keyword in $KeywordSet_{\mathcal{C}}$ is used to describe the context $\mathcal{C}$. The keywords in $KeywordSet_{\mathcal{C}}$ are semantically equivalent because they express the same context. For each context $\mathcal{C}$, we require that the $KeywordSet_{\mathcal{C}}$ should be non-empty and finite. For any two distinct contexts $\mathcal{C}$ and $\mathcal{C}'$, $KeywordSet_{\mathcal{C}} \cap KeywordSet_{\mathcal{C}'} = \emptyset$. In other words, any keyword belongs to exactly one context. An example will help illustrate the notion of contexts. The context *age* can be expressed by the keywords $\{age, yearOfBirth\}$.

Consider the two contexts *doing a job* and *doing a job well.* Modeling them as distinct concepts increases the total number of contexts that must be managed. To solve this problem, we specify *doing a job* as a context and associate a set of values with it. The values in this case will be $\{badly, neutral, well\}$. Using these values, we can specify different conditions on the context. Each of these conditions represent a *derived context.*

To obtain a derived context from the context $\mathcal{C}$, each keyword k, where $k \in KeywordSet_{\mathcal{C}}$, must be associated with a domain D_k that defines the set of values associated with the keyword. The formal definition of derived context appears below.

Definition 19 **[Derived Context]** *A* derived context *$\mathcal{DC}$ is one that is specified by a condition k op v defined over a context $\mathcal{C}$ where $k \in KeywordSet_{\mathcal{C}}$ and $v \in D_k$ and op is a logical operator compatible with the domain of D_k.*

To check whether two derived contexts specified using conditions on different keywords are equivalent, we need the notion of translation functions.

Definition 20 **[Translation Function]** *The* translation function *associated with a context $\mathcal{C}$, denoted as $TF_{\mathcal{C}}$, is a total function that takes as input a condition k op v*

($k \in KeywordSet_{\mathcal{C}}$) and a keyword k' ($k' \in KeywordSet_{\mathcal{C}}$) and produces an equivalent condition defined over keyword k'. This is formally expressed as follows. $TF_{\mathcal{C}} : Cond_{\mathcal{C}} \times KeywordSet_{\mathcal{C}} \rightarrow Cond_{\mathcal{C}}$ where $Cond_{\mathcal{C}}$ is the set of all valid conditions specified over the keywords in $KeywordSet_{\mathcal{C}}$.

Since the translation function is total, for every given valid condition and keyword there exists an equivalent condition defined on the given keyword. Several steps are involved in developing the translation function. To express k *op* v in terms of k', we need to first convert the value k to an equivalent value that is in the domain of k'. This step is performed by conversion functions which convert the value of one keyword to an equivalent value of another keyword. The second step is to convert the operator *op* into an equivalent operator *op'* that is suitable for the domain of k'. The definition of the conversion function together with the domain of the keyword can determine how the operator must be changed. Consider the two keywords *age* and *yearOfBirth*. Suppose we want to translate $age > 18$ to an equivalent condition defined over *yearOfBirth*. The first step is to convert $age = 18$ to an equivalent value defined over *yearOfBirth*. The function that converts *age* to $yearOfBirth$ will be specified as: $yearOfBirth = currentYear - age$. For $age = 18$, this function returns $yearOfBirth = 1987$. Since $yearOfBirth$ and *age* are inversely related, (that is, *age* increases as $yearOfBirth$ decreases) the operator $>$ is inverted to obtain $<$. The results obtained by the $TF_{\mathcal{C}}$ function in this case will be $yearOfBirth < 1987$.

4.1.1 Relationships between Contexts

We now describe two kinds of relations that may exist between distinct contexts. One is the generalization/specialization relationship existing between related contexts. The other is the composition relationship between possibly unrelated contexts.

4.1.1.1 Specialization Relation

Distinct contexts may be related by the specialization relationship. The specialization relation is anti-symmetric and transitive. We use the notation $\mathcal{C}_i \subset \mathcal{C}_j$ to indicate that the context $\mathcal{C}_i$ is a generalization of context $\mathcal{C}_j$. Alternately, context $\mathcal{C}_j$ is referred to as the specialization of context C_i. For instance, the contexts *makes decision* and *makes financial decisions* are related by the specialization relationship, that is, *makes decisions* $\subset$ *makes financial decisions.* Also, *makes financial decisions* $\subset$ *makes payment decisions.* By transitivity, *makes decisions* $\subset$ *makes payment decisions.*

Each specialization relationship is associated with a degree of specialization. This indicates the closeness of the two concepts. For instance, *makes payment decisions* is a specialization of *makes decision*, and *makes payment decisions* is also a specialization of *makes financial decisions.* However, the degree of specialization is different in the two cases. *makes payment decision* is closer to *makes financial decision* than *makes decision.* The *degree of specialization* captures this difference. Since two contexts related by specialization will not be exactly identical, the degree of specialization will be denoted as a fraction. The exact value of the fraction will be determined using domain knowledge.

The specialization relationship will be used in trust evaluation when information cannot be obtained for a particular context, and the values obtained from the generalized or specialized context will need to be extrapolated.

4.1.1.2 Composition Relation

Specialization captures the relationship between contexts that are related. Sometimes unrelated contexts can be linked together using the composition relation. We now describe this composition relation. A context in our model can either be an *elementary* context or a *composite* context. An elementary context is one which cannot be subdivided into other contexts. A composite context is one that is composed from other contexts using the logical 'and' operation. The individual contexts that form a composite context are referred to as the *component* contexts. A component context can either be composite or elementary.

We use the notation $C_i \ll C_j$ to indicate that the context C_i is a component of context C_j. In such cases, C_i is referred to as the component context and C_j is the composite context. For instance, we may have the component contexts *secure key generation* and *secure key distribution* that can be combined to form the composite context *secure key generation* *and distribution.* This is denoted as *secure key generation* $\ll$ *secure key generation and distribution.*

Sometimes a composite context $\mathcal{C}_i$ may be composed from the individual contexts $\mathcal{C}_j$, $\mathcal{C}_k$ and $\mathcal{C}_m$. All these contexts may not contribute equally to form $\mathcal{C}_i$. The *degree of composition* captures this idea. A degree of composition is associated with each composition relation. Since two contexts related by composition will not be exactly identical, the degree of composition is denoted as a fraction. The sum of all these fractions equals one if $\mathcal{C}_i$ is composed of $\mathcal{C}_j$, $\mathcal{C}_k$, and $\mathcal{C}_m$ only. If $\mathcal{C}_i$ is composed of $\mathcal{C}_j$, $\mathcal{C}_k$, $\mathcal{C}_m$ and also other component contexts, then the sum of fractions associated with $\mathcal{C}_j$, $\mathcal{C}_k$, and $\mathcal{C}_m$ must be equal to or less than one. The exact value of the fraction representing the degree of composition will be determined by domain knowledge.

The composition relationship is important. When trust information cannot be computed for the composite context because of missing parameter values, the related information obtained from the components can be used to compute the trust vector. Alternately, if we cannot calculate the trust vector for a component context, we can use the trust vector for the composite context and extrapolate it. Later, we show how we do this.

4.1.1.3 Context Graphs

The specialization and the composition relations can be described using one single graph which we refer to as the *context graph.* Each node n_i in this graph corresponds to a context. There are two kinds of weighted edges in this graph: composition edges and specialization edges. A composition edge (n_i, n_j), denoted by a solid arrow from node n_i to node n_j, indicates that the context represented by node n_i is a component of the context represented by node n_j. The weight on this edge indicates the degree of composition of the component context. A specialization edge (n_p, n_q), shown by a dashed arrow from node n_p to node n_q, indicates that the context represented by node n_p is a specialization of the context represented by node n_q. The weight on the edge indicates the degree of specialization of a context.

Unrelated contexts correspond to nodes in different context graphs. Each context corresponds to only one node in the set of context graphs. We denote the context graph associated with context $\mathcal{C}$ as $\mathcal{CG}_{\mathcal{C}}$. The formal definition of a context graph is as follows.

Definition 21 [Context Graph]
A context graph $\mathcal{CG} = \langle \mathcal{N}, \mathcal{E}_c \cup \mathcal{E}_s \rangle$ is a weighted directed acyclic graph satisfying the following conditions.

- *$\mathcal{N}$ is a set of nodes where each node n_i is associated with a context $\mathcal{C}_i$ and is labeled with $KeywordSet_{\mathcal{C}_i}$. $KeywordSet_{\mathcal{C}_i}$ is the set of keywords associated with the context $\mathcal{C}_i$.*

- *The set of edges in the graph can be partitioned into two sets $\mathcal{E}_c$ and $\mathcal{E}_s$. For each edge $(n_i, n_j) \in \mathcal{E}_c$, the context $\mathcal{C}_i$ corresponding to node n_i is a component of the concept $\mathcal{C}_j$ corresponding to node n_j. The weight of the edge (n_i, n_j), denoted by $w(n_i, n_j)$, indicates the degree of composition of component context that makes up the composite context. For each edge $(n_i, n_j) \in \mathcal{E}_s$, the context $\mathcal{C}_i$ corresponding to node n_i is a specialization of context $\mathcal{C}_j$ corresponding to node n_j. Here again the weight of the edge (n_i, n_j), denoted by $w(n_i, n_j)$, indicates the degree of specialization.*

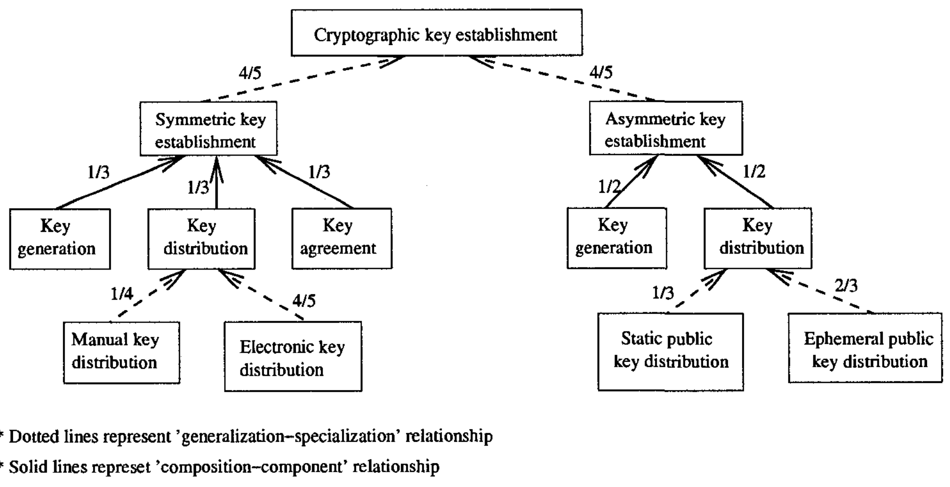

Cryptographic key establishment
4/5
4/5
Symmetric key establishment
Asymmetric key establishment
1/3
1/3
1/3
1/2
1/2
Key generation
Key distribution
Key agreement
Key generation
Key distribution
1/4
4/5
1/3
2/3
Manual key distribution
Electronic key distribution
Static public key distribution
Ephemeral public key distribution
* Dotted lines represent 'generalization–specialization' relationship
* Solid lines represet 'composition–component' relationship

Figure 4.1: Context graph showing specialization & composition relationships

Figure 4.1 gives an example of a context graph that is associated with the context *cryptographic key establishment.* Here, for simplicity of representation, we use simple phrases to label the nodes instead of keywordset. The solid arrows in this graph indicate composition relationships and the dashed arrows indicate generalization/specialization relationships. The context *cryptographic key establishment* can have two specializations, namely, *symmetric key establishment* and *asymmetric key establishment.* The weight on the edge connecting this *symmetric key establishment* with *cryptographic key establishment* indicates the degree of specialization. For instance, if symmetric key establishment is very closely related to key establishment, the degree of specialization may be labeled as $\frac{4}{5}$. Similarly, the edge connecting *asymmetric key establishment* to *key establishment* may be labeled as $\frac{4}{5}$. Each of these specific contexts is a composition of some component contexts. *Symmetric key establishment* has three components – *key generation, key distribution,* and *key agreement.* A weight of $\frac{1}{3}$ can be assigned to each of these components contexts. Similarly, *asymmetric key establishment* have components *key generation* and *key distribution* with weights $\frac{1}{2}$ each.

A component context can also be a generalization of some specialized contexts. In the above example the context *key distribution* has two categories – *manual key distribution* and *electronic key distribution.* Similarly *key distribution* in asymmetric keys can be thought of as generalization of *static public key distribution* and *ephemeral public key distribution.*

4.1.2 Computing the Degree of Specialization and Composition

4.1.2.1 Computing Degree of Specialization

Consider two contexts $\mathcal{C}_i$ and $\mathcal{C}_j$ where $\mathcal{C}_i \subset \mathcal{C}_j$, that is, $\mathcal{C}_j$ is a specialization of $\mathcal{C}_i$. The degree of specialization is computed as follows. Let n_i, n_j be the nodes corresponding to contexts $\mathcal{C}_i$ and $\mathcal{C}_j$ in the weighted graph. Let the path from n_i to n_j consisting of specialization edges be denoted as $(n_i, n_{i+1}, n_{i+2}, \ldots, n_{j-1}, n_j)$. The degree of specialization $= \Pi_{p=i}^{j-1} w(n_p, n_{p+1})$. This corresponds to our notion that the similarity decreases as the length of the path from the generalized node to the specialized node increases. Note that, in real world there may be multiple paths from $\mathcal{C}_i$ to $\mathcal{C}_j$. In such cases, it is important that the degree of specialization yield the same values when any of these paths are used for computation. An example will

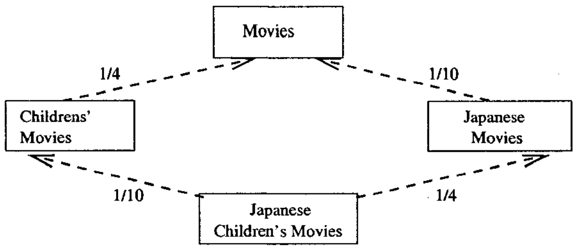


Figure 4.2: Computing the degree of specialization

help illustrate this point. Consider the following specialization relationships: (a) *Movies* $\subset$ *Japanese Movies* $\subset$ *Japanese Children's Movies* and (b) *Movies* $\subset$ *Children's Movies* $\subset$ *Japanese Children's Movies.* Suppose one is computing the degree of specialization that exists between *Movies* and *Japanese Children's Movies.* In this case, there are two paths consisting of specialization edges between the concepts *Movies* and *Japanese Children's Movies.* Computing the degree of specialization using any of these two paths should yield the same value. Calculating the degree of specialization that exists between *Movies* and *Japanese Children's Movies* using the values given in Figure 4.2 yields $\frac{1}{40}$.

4.1.2.2 Computing Degree of Composition

Consider two contexts C_i and C_j such that C_j is a component of C_i. Degree of composition captures what portion of C_i is made up of C_j. The degree of composition is computed as follows. Let n_i, n_j be the nodes corresponding to contexts C_i and C_j in the context graph. Let there be m paths consisting of composition edges from n_i to n_j. Let the qth path $(1 \leq q \leq m)$ from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \ldots, n_{j_q-1}, n_j)$. The degree of composition $= \Sigma_{q=1}^{m}(w(n_i, n_{i_q+1}) \times w(n_{j_q-1}, n_j) \times \Pi_{p=i_q+1}^{j_q-2} w(n_p, n_{p+1}))$. A *House* is composed of *Doors*, *Windows*, and *Walls.* A *Door* is composed of *Wood.* A *Window* is composed of *Wood* and *Glass.* Therefore, we have the following composition relationships: *Windows* $\ll$ *House*, *Doors* $\ll$ *House*, *Wood* $\ll$ *Windows*, and *Wood* $\ll$ *Doors.* The composition relationships are shown in Figure 4.3. Thus, to evaluate what part of house is made of wood, we have to consider all the paths. The degree of composition of *Wood* and *House* is $\frac{5}{24}$.

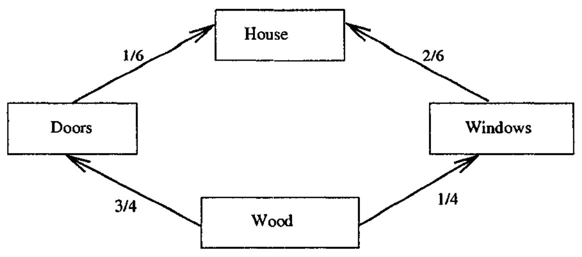


Figure 4.3: Computing the degree of composition

4.1.3 Closest Context

A context may be related to several other contexts through specialization and composition relationships. However, we need to find out which context or set of contexts is conceptually closest to the given context. The closest context is a singleton set if the context is a generalization or specialization of context C. It is also a singleton set if it is a composite context in which C is a component. However, if C is a composite context, then the closest context can also be a set that contains the components of C. The formal definition of closest context appears below.

Definition 22 [Closest Context] *Let C be a context. The set of contexts $S = \{C_1, C_2, \ldots, C_n\}$ is defined to be* closest *to C if the following relation holds:*

Case 1 – The elements in S are the components of C:

for all contexts n_i that are specializations of C

degree of specialization$(n_i, C) \leq \Sigma_{j=1}^{n}$ degree of composition(C_j, C)

for all contexts n_i that are generalizations of C

degree of specialization$(C, n_i) \leq \Sigma_{j=1}^{n}$ degree of composition(C_j, C)

for all composite contexts n_i in which c is a component

degree of composition$(C, n_i) \leq \Sigma_{j=1}^{n}$ degree of composition(C_j, C)

Case 2 – S is a singleton set containing C_1 and C is a component of C_1:

for all contexts n_i that are specializations of C

degree of specialization$(n_i, \mathcal{C}) \leq$ *degree of composition*$(\mathcal{C}, \mathcal{C}_1)$

for all contexts n_i *that are generalizations of* $\mathcal{C}$

degree of specialization$(\mathcal{C}, n_i) \leq$ *degree of composition*$(\mathcal{C}, \mathcal{C}_1)$

for all contexts n_1, n_2, ..., n_m *that are components of* $\mathcal{C}$

$\Sigma_{i=1}^{m}$ *degree of composition*$(n_i, \mathcal{C}) \leq$ *degree of composition*$(\mathcal{C}, \mathcal{C}_1)$

for all composite contexts n_i *in which* $\mathcal{C}$ *is a component*

degree of composition$(\mathcal{C}, n_i) \leq$ *degree of composition*$(\mathcal{C}, \mathcal{C}_1)$

Case 3 – S is a singleton set containing $\mathcal{C}_1$ *and* $\mathcal{C}$ *is a specialization of* $\mathcal{C}_1$*:*

for all component contexts n_1, n_2. ..., n_m *of* $\mathcal{C}$

$\Sigma_{i=1}^{m}$ *degree of composition*$(n_i, \mathcal{C}) \leq$ *degree of specialization*$(\mathcal{C}, \mathcal{C}_1)$

for all contexts n_i *that are specializations of* $\mathcal{C}$

degree of specialization$(n_i, \mathcal{C}) \leq$ *degree of specialization*$(\mathcal{C}, \mathcal{C}_1)$

for all contexts n_i *that are generalizations of* $\mathcal{C}$

degree of specialization$(\mathcal{C}, n_i) \leq$ *degree of specialization*$(\mathcal{C}, \mathcal{C}_1)$

for all composite contexts n_i *in which* $\mathcal{C}$ *is a component*

degree of composition$(\mathcal{C}, n_i) \leq$ *degree of specialization*$(\mathcal{C}, \mathcal{C}_1)$

Case 4 – S is a singleton set containing $\mathcal{C}_1$ *and* $\mathcal{C}$ *is a generalization of* $\mathcal{C}_1$*:*

for all component contexts n_1, n_2. ..., n_m *of* $\mathcal{C}$

$\Sigma_{i=1}^{m}$ *degree of composition*$(n_i, \mathcal{C}) \leq$ *degree of specialization*$(\mathcal{C}_1, \mathcal{C})$

for all composite contexts n_i *in which* $\mathcal{C}$ *is a component*

degree of composition$(\mathcal{C}, n_i) \leq$ *degree of specialization*$(\mathcal{C}_1, \mathcal{C})$

for all contexts n_i *that are specializations of* $\mathcal{C}$

degree of specialization$(n_i, \mathcal{C}) \leq$ *degree of specialization*$(\mathcal{C}_1, \mathcal{C})$

for all contexts n_i *that are generalizations of* $\mathcal{C}$

degree of specialization$(\mathcal{C}, n_i) \leq$ *degree of specialization*$(\mathcal{C}_1, \mathcal{C})$

4.1.4 Relationships between Context Graphs

Different information sources may use different context graphs. Comparing information or combining information that uses different context graphs may not give correct results. Before proceeding with the comparison of information obtained from different sources, the context graphs of these sources must be merged. Note that, sometimes context graphs cannot be merged because they contain conflicting information. To understand why this happens, we first need to elaborate on the relationships that can exist between a pair of context graphs. Two context graphs can be related by any of the following relationships: (i) equality, (ii) unrelated, (iii) subsumes, and (iv) incomparable.

Definition 23 [Equality of Context Graphs] *Two context graphs $\mathcal{CG}_1 =< \mathcal{N}_1, \mathcal{E}_{1c} \cup \mathcal{E}_{1s} >$ and $\mathcal{CG}_2 =< \mathcal{N}_2, \mathcal{E}_{2c} \cup \mathcal{E}_{2s} >$ are said to be equal if*

1. *$\mathcal{N}_1 = \mathcal{N}_2$, $\mathcal{E}_{1c} = \mathcal{E}_{2c}$ and $\mathcal{E}_{1s} = \mathcal{E}_{2s}$*

2. *for each $(n_i, n_j) \in (\mathcal{E}_{1c} \cup \mathcal{E}_{1s}) \cap (\mathcal{E}_{2c} \cup \mathcal{E}_{2s})$, $w_1(n_i, n_j) = w_2(n_i, n_j)$ where $w_1(n_i, n_j)$, $w_2(n_i, n_j)$ denote the weight of the edge (n_i, n_j) in graph $\mathcal{CG}_1$, $\mathcal{CG}_2$ respectively.*

Intuitively, two context graphs are equal if they have the same set of nodes, composition edges, and specialization edges. Moreover, each of these edges must have identical weights in the two graphs. The information obtained from identical context graphs can be compared.

Sometimes two context graphs are unrelated. They do not have any common context. It is conceivable that these graphs will be used for different situations.

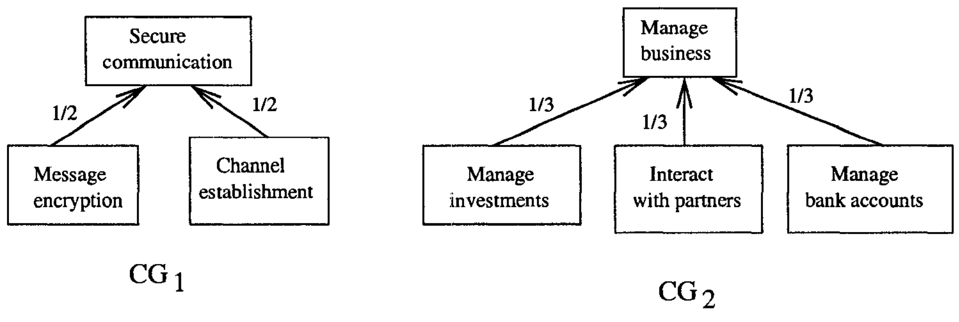


Figure 4.4: Unrelated context graphs

Definition 24 [Unrelated Context Graphs] *Two context graphs $\mathcal{CG}_1$ and $\mathcal{CG}_2$ are said to be unrelated if $KeywordSet_{\mathcal{C}_1} \cap KeywordSet_{\mathcal{C}_2} = \emptyset$ where $KeywordSet_{\mathcal{C}_1}$ and $KeywordSet_{\mathcal{C}_2}$ are the set of keywords associated with all the contexts in the context graphs $\mathcal{CG}_1$ and $\mathcal{CG}_2$ respectively.*

Often times two context graphs are comparable but one has more information than the other. In such cases, the context graphs are related by the subsumes relation. The intuition is that the context graph $\mathcal{CG}$ has more information than the one it subsumes. The formal definition is given below. If $\mathcal{CG}_1$ subsumes $\mathcal{CG}_2$, the first condition requires that the set of nodes in $\mathcal{CG}_1$ is greater than or equal to the set of nodes in $\mathcal{CG}_2$. The second condition requires that for every specialization edge (n_i, n_j) present in $\mathcal{E}_{2s}$, there exists a path from n_i to n_j in $\mathcal{CG}_1$ consisting of specialization edges such that the product of the weight of these edges equals the weight of (n_i, n_j) in $\mathcal{CG}_2$. The third condition imposes a similar requirement for the composition edges.

Definition 25 [Context Graphs related by the Subsumes Relation] *Consider two context graphs $\mathcal{CG}_1 =< \mathcal{N}_1, \mathcal{E}_{1c} \cup \mathcal{E}_{1s} >$ and $\mathcal{CG}_2 =< \mathcal{N}_2, \mathcal{E}_{2c} \cup \mathcal{E}_{2s} >$. Let $w_1(n_i, n_j)$, $w_2(n_i, n_j)$ represent the weight of edge (n_i, n_j) in graph $\mathcal{CG}_1$ and $\mathcal{CG}_2$ respectively. $\mathcal{CG}_1$ is said to subsume $\mathcal{CG}_2$ if it satisfies all the following conditions:*

1. *$\mathcal{N}_2 \subseteq \mathcal{N}_1$*

2. *for each specialization edge $(n_i, n_j) \in \mathcal{E}_{2s}$ there exists a path $\{n_i, n_{i+1}, n_{i+2}, \ldots, n_{j-1}, n_j\}$ in $\mathcal{CG}_1$ whose length ≥ 1 such that $\{(n_i, n_{i+1}), (n_{i+1}, n_{i+2}), (n_{i+2}, n_{i+3}), \ldots, (n_{j-1}, n_j)\} \subseteq \mathcal{E}_{1s}$ and $\prod_{p=i}^{j-1} w_1(n_p, n_{p+1}) = w_2(n_i, n_j)$*

3. *for each composition edge $(n_i, n_j) \in \mathcal{E}_{2c}$ there exists m paths consisting only of composition edges where $m > 1$ in $\mathcal{CG}_1$. Let the qth path $(1 \leq q \leq m)$ in $\mathcal{CG}_1$ from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \ldots, n_{j_q-1}, n_j)$ where $\{(n_i, n_{i_q+1}), (n_{i_q+1}, n_{i_q+2}), \ldots, (n_{j_q-1}, n_j)\} \subseteq \mathcal{E}_{1c}$. $w_2(n_i, n_j) = \Sigma_{q=1}^{m}(w_1(n_i, n_{i_q+1}) \times w_1(n_{j_q-1}, n_j) \times \Pi_{p=i_q+1}^{j_q-2} w_1(n_p, n_{p+1}))$*

Often times two context graphs, neither of which subsumes the other, may be comparable. Such graphs contain different but related information. Moreover, they never have any conflicting information. Such graphs can be merged without human intervention. Two context graphs are comparable if they satisfy a set of conditions. The first condition requires that the two graphs have one or more common nodes. The second condition requires that for any specialization edge (n_i, n_j) present in $\mathcal{E}_{1s}$, either there must be a path from n_i to n_j in $\mathcal{E}_{2s}$ consisting of specialization edges in $\mathcal{E}_{1s}$ and whose product equals $w_1(n_i, n_j)$ or no path exists between n_i and n_j in $\mathcal{E}_{2s}$. The third condition imposes a similar requirement for composition edges in $\mathcal{E}_{1c}$. This condition requires that either there are m paths $(m > 1)$ consisting of composition edges from n_i to n_j in $\mathcal{CG}_2$ or there are no paths. Computing the degree of composition along these paths gives the same result as $w_1(n_i, n_j)$. The fourth and the fifth requirements imposes similar requirements for edges in $\mathcal{E}_{2c}$. The formal definition appears below.

Definition 26 [Comparable Context Graphs] *Two context graphs $\mathcal{CG}_1 =< \mathcal{N}_1, \mathcal{E}_{1c} \cup \mathcal{E}_{1s} >$ and $\mathcal{CG}_2 =< \mathcal{N}_2, \mathcal{E}_{2c} \cup \mathcal{E}_{2s} >$ are said to be comparable if the following conditions hold.*

1. $\mathcal{N}_1 \cap \mathcal{N}_2 \neq \emptyset$

2. *for each $(n_i, n_j) \in \mathcal{E}_{1s}$ either of the following conditions hold:*

   (a) $\exists\ n_{i+1}, n_{i+2}, \ldots, n_{j-1} \in \mathcal{N}_2 - \mathcal{N}_1 \bullet ((\{(n_i, n_{i+1}), (n_{i+1}, n_{i+2}), \ldots, (n_{j-1}, n_j)\} \subseteq \mathcal{E}_{2s}) \wedge \prod_{t=i}^{j-1} w_2(n_t, n_{t+1}) = w_1(n_i, n_j))$

   (b) *there does not exist any path from n_i to n_j in $\mathcal{CG}_2$*

3. *for each $(n_i, n_j) \in \mathcal{E}_{1c}$ either of the following conditions hold:*

   (a) *there exists m paths $(m > 1)$ from n_i to n_j in $\mathcal{CG}_2$. Let the qth path $(1 \leq q \leq m)$ from n_i to n_j be denoted as $(n_i, n_{i_q+1}, n_{i_q+2}, \ldots, n_{j_q-1}, n_j)$ where $\{(n_i, n_{i_q+1}), (n_{i_q+1}, n_{i_q+2}), \ldots, (n_{j_q-2}, n_{j_q-1}), (n_{j_q-1}, n_j)\} \subseteq \mathcal{E}_{2c}$. In such a case, $w_1(n_i, n_j) = \Sigma_{q=1}^{m}(w_2(n_i, n_{i_q+1}) \times w_2(n_{j_q-1}, n_j) \times \Pi_{p=i_q+1}^{j_q-2} w_2(n_p, n_{p+1}))$*

   (b) *there does not exist any path from n_i to n_j in $\mathcal{CG}_2$*

4. *for each* $(n_i, n_j) \in \mathcal{E}_{2s}$ *either of the following conditions hold:*

   (a) $\exists\, n_{i+1}, n_{i+2}, \ldots, n_{j-1} \in \mathcal{N}_1 - \mathcal{N}_2 \bullet ((\{(n_i, n_{i+1}), (n_{i+1}, n_{i+2}), (n_{i+2}, n_{i+3}), \ldots, (n_{j-1}, n_j)\} \subseteq \mathcal{E}_{1s}) \wedge \prod_{t=i}^{j-1} w_1(n_t, n_{t+1}) = w_2(n_i, n_j))$

   (b) *there does not exist any path from* n_i *to* n_j *in* $\mathcal{E}_{1s}$

5. *for each* $(n_i, n_j) \in \mathcal{E}_{2c}$ *either of the following conditions hold:*

   (a) *there exists* m *paths* $(m > 1)$ *from* n_i *to* n_j *in* $\mathcal{CG}_1$*. Let the* q*th path* $(1 \leq q \leq m)$ *from* n_i *to* n_j *be denoted as* $(n_i, n_{i_q+1}, n_{i_q+2}, \ldots, n_{j_q-2}, n_{j_q-1}, n_j)$ *where* $\{(n_i, n_{i_q+1}), (n_{i_q+1}, n_{i_q+2}), \ldots, (n_{j_q-2}, n_{j_q-1}), (n_{j_q-1}, n_j)\} \subseteq \mathcal{E}_{1c}$*. In such a case,* $w_2(n_i, n_j) = \Sigma_{q=1}^{m}(w_1(n_i, n_{i_q+1}) \times w_1(n_{j_q-1}, n_j) \times \Pi_{p=i_q+1}^{j_q-2} w_1(n_p, n_{p+1}))$

   (b) *there does not exist any path from* n_i *to* n_j *in* $\mathcal{CG}_1$

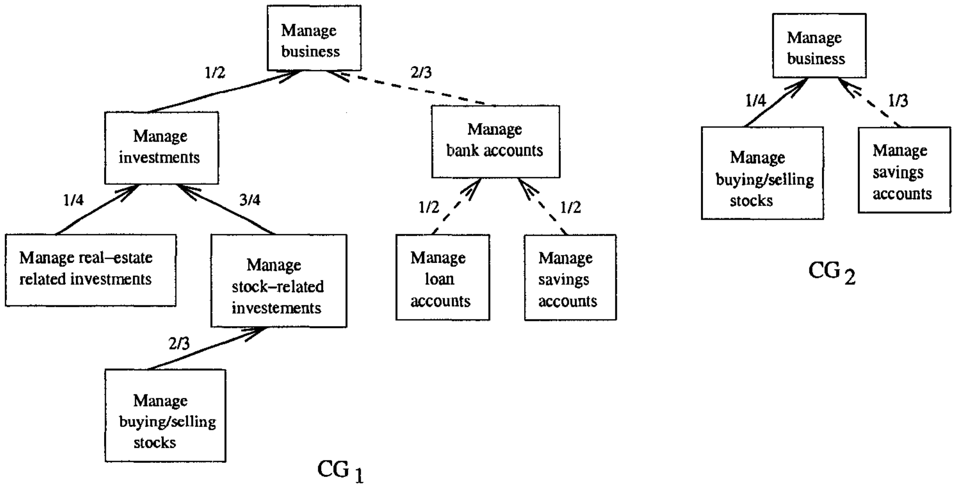


Figure 4.5: Context graphs having subsumes relation

Definition 27 [**Incomparable Context Graphs**] *Two context graphs that are not unrelated are incomparable if they are not comparable.*

Incomparable graphs occur when the underlying assumptions are different. For example, one system may think that the degree of specialization of the contexts *makes financial decisions* and *makes payment decision* is 0.5, another might think this degree is 0.3. In

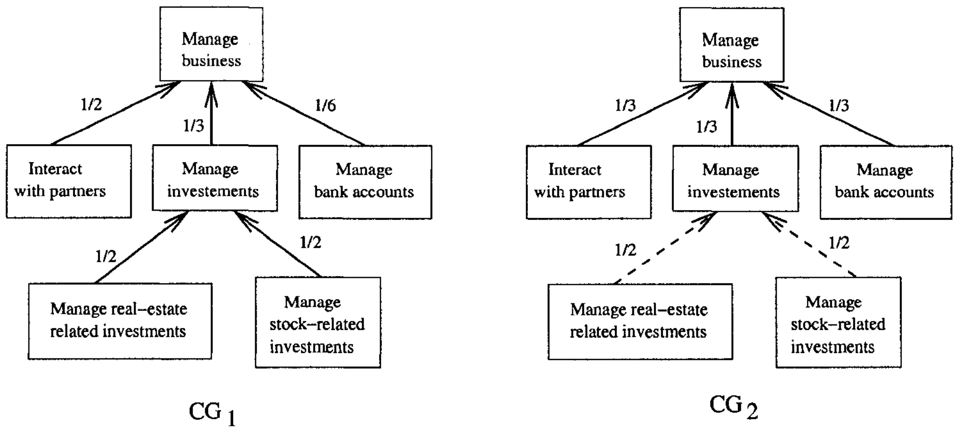


Figure 4.6: Incomparable context graphs

this case, the two systems will generate context graphs in which some edge present in both the graphs will have different weights. Alternately, one system may consider that *makes financial decisions* and *makes payment decisions* are related by generalization/specialization relationships whereas another system may consider them linked by a composition relationship. The systems will generate context graphs in which some pair of nodes present in both the graphs will be related by different types of edges. Since the conflicts are generated because of the differences in the underlying assumptions, they cannot be resolved without human intervention.

Next we give an algorithm for combining comparable graphs. The inputs to this algorithm are $\mathcal{CG}_1 =< \mathcal{N}_1, \mathcal{E}_1 >$ and $\mathcal{CG}_2 =< \mathcal{N}_2, \mathcal{E}_2 >$ – the two context graphs that must be merged. The output is $\mathcal{CG} =< \mathcal{N}, \mathcal{E} >$ which is the combined context graph. $\mathcal{N}$ is computed by performing a union of the nodes in $\mathcal{N}_1$ and $\mathcal{N}_2$. The edges common to $\mathcal{E}_1$ and $\mathcal{E}_2$ are inserted in $\mathcal{E}$. For each edge (n_i, n_j) that is present in $\mathcal{E}_1$ but not in $\mathcal{E}_2$, we check if there is a path from n_i to n_j in $\mathcal{E}_2$. If so, then these edges of $\mathcal{E}_2$ are added to $\mathcal{E}$. Otherwise, the edge (n_i, n_j) is added. The same process is followed for edges present in $\mathcal{E}_2$ but not in $\mathcal{E}_1$. The resulting graph so obtained subsumes the graphs $\mathcal{CG}_1$ and $\mathcal{CG}_2$.

Algorithm 2 Combining Comparable Graphs

```
**Input:** $\mathcal{CG}_1 =< \mathcal{N}_1, \mathcal{E}_1 >$ and $\mathcal{CG}_2 =< \mathcal{N}_2, \mathcal{E}_2 >$ – comparable context graphs.
**Output:** $\mathcal{CG} =< \mathcal{N}, \mathcal{E} >$ – the combined context graph.

**Procedure** $CombineContextGraphs(\mathcal{CG}_1, \mathcal{CG}_2)$
$\mathcal{E} = \{\}$ ; $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2$;
**for all** i, j such that edge $(n_i, n_j) \in \mathcal{E}_1 \cap \mathcal{E}_2$ **do**
  $\mathcal{E} = \mathcal{E} \cup \{(n_i, n_j)\}$;
  $w(n_i, n_j) = w_1(n_i, n_j)$;
**end for**
**for all** $i, j$ such that edge $(n_i, n_j) \in \mathcal{E}_1 - \mathcal{E}_2$ **do**
  **for all** $i, j$ such that path $(n_i, n_{i+1}, n_{i+2}, \ldots, n_j)$ between $n_i$ and $n_j$ in $\mathcal{CG}_2$ **do**
    **for all** $i, j$ such that $(n_i, n_{i+1})$ in the path **do**
      $\mathcal{E} = \mathcal{E} \cup \{(n_i, n_{i+1})\}$;
      $w(n_i, n_{i+1}) = w_2(n_i, n_{i+1})$;
    **end for**
  **end for**
**end for**
**if** there is no path between $n_i$ and $n_j$ in $\mathcal{CG}_2$ **then**
  $\mathcal{E} = \mathcal{E} \cup \{(n_i, n_j)\}$;
  $w(n_i, n_j) = w_1(n_i, n_j)$;
**end if**
**for all** $i, j$ such that edge $(n_i, n_j) \in \mathcal{E}_2 - \mathcal{E}_1$ **do**
  **for all** $i, j$ such that path $(n_i, n_{i+1}, n_{i+2}, \ldots, n_j)$ from $n_i$ to $n_j$ in $\mathcal{CG}_1$ **do**
    **for all** $i, j$ such that $(n_i, n_{i+1})$ in the path **do**
      $\mathcal{E} = \mathcal{E} \cup \{(n_i, n_{i+1})\}$;
      $w(n_i, n_{i+1}) = w_1(n_i, n_{i+1})$;
    **end for**
  **end for**
**end for**
**if** there is no path from $n_i$ to $n_j$ in $\mathcal{CG}_1$ **then**
  $\mathcal{E} = \mathcal{E} \cup \{(n_i, n_j)\}$;
  $w(n_i, n_j) = w_2(n_i, n_j)$;
**end if**
```

4.2 Evaluating Trust without Complete Information

The vector trust model presented so far describes how trust pertaining to some context is evaluated, how trust changes with time, and how different trust vectors can be compared. All this relies on the assumption that the trust vector can be determined in a given context. This may not be possible when complete information is not available. For instance, the trust evaluation policy vector may assign a weight of 0.5 to the recommendation component and it may not be possible to obtain any recommendation about the trustee under the given context. In that case the truster is totally uncertain about the recommendation (recommendation value is $\perp$ in this case). In the worst case, it may not be possible to obtain information about any of the components for a given context. That is, the values of the parameters are unknown ($\perp$) to the truster. This situation is not very uncommon – suppose a truster is trying to determine the trustworthiness of a new software product. In such a case, the truster may not have any interaction or knowledge about the properties pertaining to this product. Obtaining a reputation or a recommendation is also not possible. In such a case, the model that we have presented so far, cannot determine the trust vector of the software product. In this section, we present an approach using which a truster can obtain an approximate trust value of the given software product. Our approach exploits the relationships between contexts in order to extrapolate the values related to trust.

4.2.1 Extrapolating Trust Values from Related Contexts

When a truster A cannot determine the values related to his trust relationship with trustee B for a context $\mathcal{C}$, we show how the values can be obtained from one or more related contexts, say, $\mathcal{C}_i$. The first issue that we must resolve is what values should we use from the related context $\mathcal{C}_i$. There are two possibilities. We can use the trust value from $\mathcal{C}_i$ to extrapolate the trust value for $\mathcal{C}$. Alternately, we can use the values of the individual parameters interactions, properties, reputation, and recommendation from $\mathcal{C}_i$ and use these to compute the trust vector for $\mathcal{C}$. We adopt the second approach for two reasons. First, the trust-parameter weight vector can be different for contexts $\mathcal{C}$ and $\mathcal{C}_i$. Using the parameter values of $\mathcal{C}_i$ to extrapolate the parameter values for $\mathcal{C}$ will not be very meaningful. Second, we may not be missing all the parameters of $\mathcal{C}$. For instance, we may have values for recommendation and properties for context $\mathcal{C}$ but no value for interactions and reputation. In such a case, we would want to extrapolate only the interactions and reputation parameter from the context $\mathcal{C}_i$.

The second issue is that a context $\mathcal{C}$ may be related to many other contexts, say, $\mathcal{C}_i$, $\mathcal{C}_j$, and $\mathcal{C}_k$. Which contexts do we refer to in order to evaluate the trust vector for context $\mathcal{C}$? Many strategies are possible and different strategies may be needed in different real-world situations. In this dissertation, we propose a very simple strategy for choosing related contexts. The algorithm given below describes this strategy. This algorithm has three inputs. One is the context $\mathcal{C}$ whose closest context we are trying to determine. The other is the context graph $\mathcal{CG}$ in which $\mathcal{C}$ is a context. The third input is the set of contexts that we should not consider. We term this as the prohibited set of contexts. The algorithm uses a variable *total_weight* that is used for evaluating the closest context. The algorithm proceeds by checking the component contexts of $\mathcal{C}$. If these are not prohibited, then the total weight of all these component contexts makes up the variable *total_weight.* These component contexts are inserted into the set *closest* which contains the closest context. We then check each generalized parent and each specialized children whether the weight on the edge is greater than the *total_weight.* If so, *total_weight* is assigned this new weight and closest is initialized with this parent or children. The algorithm returns the set *closest* which gives the set of contexts closest to $\mathcal{C}$. Finally, we give an example that shows how the value of a parameter, say recommendation, about a trustee B can be obtained from the closest context. We describe the steps to be performed in the form of an algorithm. The steps show how to extrapolate the values of the parameter recommendation when it is $\perp$. It does this by getting the recommendation from its closest relation and multiplying it by the weight of the edge. Similar algorithms can be developed for extrapolating the value related to the other parameters.

Algorithm 3 Get the closest context

```
Input: (i) $\mathcal{C}$ – the context whose closest one needs to be determined. (ii) $\mathcal{CG}$ – the context graph in which $\mathcal{C}$ is a context. (iii) $S$ – set of contexts that should not be considered.
Output: $closest$ – set of contexts closest to $c$.

Procedure $ChooseClosestContext(\mathcal{C}, \mathcal{CG}, S)$
$closest = \emptyset$; $total\_weight = 0$;
Let $CC$ be the set of component contexts of $\mathcal{C}$;
for all $i$ such that $c_i \in CC(\mathcal{C})$ do
  if $c_i \notin S$ then
    $total\_weight = total\_weight + w(c_i, \mathcal{C})$;
    $closest = closest \cup \{c_i\}$;
  end if
end for
for all $i$ such that $p_i$ is a generalization or composite context of $\mathcal{C}$ do
  if $p_i \notin S$ and $total\_weight < w(\mathcal{C}, p_i)$ then
    $total\_weight = w(\mathcal{C}, p_i)$;
    $closest = \{p_i\}$;
  end if
end for
for all $i$ such that $r_i$ is a specialization of $\mathcal{C}$ do
  if $r_i \notin S$ and $total\_weight < w(r_i, \mathcal{C})$ then
    $total\_weight = w(r_i, \mathcal{C})$;
    $closest = \{r_i\}$;
  end if
end for
return $closest$;
```

Algorithm 4 Get Recommendation about Trustee B from the Closest Context

```
Input: (i) $\mathcal{C}$ – the context whose recommendation value needs to be determined. (ii) $\mathcal{CG}$
– the context graph in which $\mathcal{C}$ is a context.
Output: ${}_{\Psi}REC_B^{\mathcal{C}}$ – recommendation in context $\mathcal{C}$.

Procedure GetRecommendation($\mathcal{C}, \mathcal{CG}, S$)
if ${}_{\Psi}REC_B^{\mathcal{C}} \neq \perp$ then
  return ${}_{\Psi}REC_B^{\mathcal{C}}$;
else
  $S = \emptyset$; $closest = \{\mathcal{C}\}$;
  while ${}_{\Psi}REC_B^{\mathcal{C}} = \perp$ and $closest \neq \emptyset$ do
    $closest = ChooseClosestContext(\mathcal{C}, \mathcal{CG}, S)$;
    Case 1: $closest = \{\mathcal{C}_i\}$ and $\mathcal{C}$ is a component or specialization of $\mathcal{C}_i$.
    if ${}_{\Psi}REC_B^{\mathcal{C}_i} \neq \perp$ then
      ${}_{\Psi}REC_B^{\mathcal{C}} = w(\mathcal{C}, \mathcal{C}_i) \times_{\Psi} REC_B^{\mathcal{C}_i}$;
      return ${}_{\Psi}REC_B^{\mathcal{C}}$;
    else
      $S = S \cup \{\mathcal{C}_i\}$;
    end if
    Case 2: $closest = \{\mathcal{C}_i\}$ and $\mathcal{C}$ is a generalization of $\mathcal{C}_i$;
    if ${}_{\Psi}REC_B^{\mathcal{C}_i} \neq \perp$ then
      ${}_{\Psi}REC_B^{\mathcal{C}} = w(\mathcal{C}_i, \mathcal{C}) \times_{\Psi} REC_B^{\mathcal{C}_i}$;
      return ${}_{\Psi}REC_B^{\mathcal{C}}$;
    else
      $S = S \cup \{\mathcal{C}_i\}$;
    end if
    Case 3: $closest = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$ and $\mathcal{C}$ is composed of $\mathcal{C}_i$ $(i = 1, 2, \ldots, n)$;
    Set $j = 0$ and ${}_{\Psi}REC_B^{\mathcal{C}} = \perp$;
    for all $i$ such that $\mathcal{C}_i \in closest$ do
      if ${}_{\Psi}REC_B^{\mathcal{C}_i} \neq \perp$ then
        ${}_{\Psi}REC_B^{\mathcal{C}} = {}_{\Psi}REC_B^{\mathcal{C}} + w(\mathcal{C}_i, \mathcal{C}) \times {}_{\Psi}REC_B^{\mathcal{C}_i}$;
        $j = j + 1$;
      else
        $S = S \cup \{\mathcal{C}_i\}$;
      end if
      ${}_{\Psi}REC_B^{\mathcal{C}} = \frac{1}{j} \times {}_{\Psi}REC_B^{\mathcal{C}}$;
    end for
    return ${}_{\Psi}REC_B^{\mathcal{C}}$;
  end while
end if
```

4.3 Summary

This chapter presents another major contribution of this dissertation – formalism of trust context and relationship between different contexts. The context ontology includes definition of a context using set of similar meaning keywords. This helps us to use the model in an interoperable manner, where different entities may use different keywords from the set to specify the same context. Different contexts are related to each other with specialization/generalization and component/composite relations. These relationships, together with their corresponding degrees, are graphically represented using context graph. A context graph is used to find the set of closest context for given context. In particular, the concept of closest contexts and the algorithm to find it, are significant contributions as these help to estimate the trust value in a context when no information is available to evaluate trust in that context. The trust, in such scenarios, is extrapolated from the available trust values in the closest contexts. All the above features help to use the vector trust model in more interoperable and flexible manner and are otherwise unavailable in the existing trust literature.

Chapter 5

Validation

One of the ways to validate the trust model is to measure the relative trustworthiness of different entities in a security context, where a result is known from practice or other evaluation methods, and check how close the result obtained using the trust model can match the known result. For this purpose, we consider evaluating trust, using the vector trust model, about the security level of two mechanisms for protecting against Denial of Service (DoS) attacks. A particular type of DoS attacks (TCP SYN flooding), for a .NET e-commerce system is considered. The two security solutions that are considered for the validation purpose are *cookie solution* and *filtering mechanism.* Existing security evaluation methods indicate that cookie solution is relatively better than filtering mechanism to protect against DoS attacks. For the validation purpose, we adapt the vector trust model to evaluate the relative trustworthiness of these two security solutions and compare the result with the known result. Next we describe how to use the vector trust model to evaluate the trust about the security level of two mechanisms for protecting against Denial of Service (DoS) attacks[1].

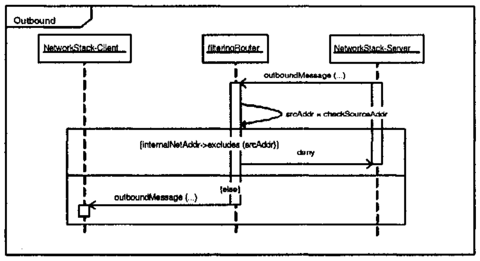
We illustrate our approach using the user authentication mechanism of an e-commerce platform, named ACTIVE. This was developed by the EU EP-27046-ACTIVE project [EU 01]. It is a standard .NET e-commerce system offering a set of services that end-

[1]This validation work is a part of the work presented in [HCRR08].

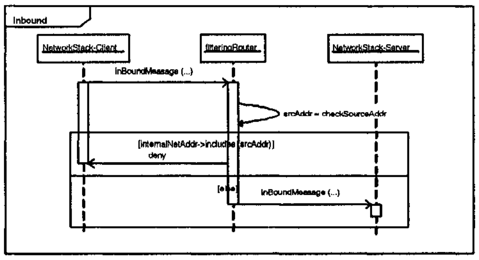
users can purchase online. To access any of the services in ACTIVE, users must either login as a registered user or as a visitor. The risk analysis of this login mechanism showed that it represents a security problem if the login actions are not properly protected [HGF$^+$05b, HGF$^+$05a]. The potential security problems are the different types of denial of service (DoS) attacks, such as TCP SYN flooding [Cen00] and IP spoofing [Cen97]. During such attacks, user names and passwords can be intercepted by an attacker and used later to impersonate as a valid user. These attacks have also been identified by the IST EU-project CORAS [CORa] when they performed assessments of the login mechanism of ACTIVE during the period 2000–2003.

The security attributes integrity and confidentiality are both compromised in these types of attacks. The solution to this problem are mechanisms that address integrity and confidentiality. Here we evaluate two such mechanisms – a cookie solution and a filtering mechanism. The cookie solution adds a patch to the network stack software that keeps track of sessions and their states. At first, a cookie is sent to the client and the pending connection is removed. If the client does not respond within a short period of time, the cookie expires and the client must re-start the request for a connection. If the client responds in time, the SYN-ACK message is sent and the connection is set up. Adding the cookie message makes it unlikely that an attacker can respond in time to continue setting up the connection. The cookie will expire on the server and the connection attempt will be closed. If the client address has been spoofed, the client will not respond in any event. The filtering mechanism works in a different way. The filtering mechanism has an outbound and an inbound part, shown in Figure 5.1(a) and 5.1(b) respectively, that checks the source address (srcAddr) against a set of accepted source IP addresses stored in internalNetAddr. Rather than adding control through the additional cookie, the filtering mechanism is implemented on the server side (usually on a firewall or an Internet router) and configured to block unauthorized connection attempts.

In our approach a truster A needs help to choose between the two security solutions. The truster maker uses the vector trust model to estimate the trust level of the security solutions.



(a) Outbound



(b) Inbound

Figure 5.1: Filtering mechanism

For this purpose A seeks help of information sources regarding anticipated number of DoS attacks for the two solutions. That is, A uses the *recommendation* about the security solutions from the information sources. A also adapts the vector trust model to evaluate the trust level of the information sources to scale their respective recommendation about the security solution. The next section presents the adapted model to evaluate information source trustworthiness and shows how this helps to estimate the trust level of security solutions.

5.1 Evaluating Trust Level of a Security Solution

We evaluate the trust level of a security solution by aggregating information from different information sources. Since many different types of sources with varying degrees of trustworthiness are used, the trustworthiness of a source must be factored in when using the information it provides. We adopt the vector trust model to evaluate the trustworthiness of information sources. In our approach, we assume that the trustworthiness of an information source depends on two parameters – *knowledge level* and *expertise level*. The knowledge level, which is assessed by a third party expert, is considered as a *recommendation* about the information source where the third party expert is the recommender. The expertise score can be considered as *properties* of the information source, where the information to evaluate this parameter is provided by the information source itself using a questionnaire. These are formally defined below.

Definition 28 **[Knowledge Level]** *Knowledge level of an information source is defined as a measure of awareness of the information source about the knowledge domains related to the security level of the security solutions. It is represented in terms of a number called* knowledge score.

Definition 29 **[Expertise Level]** *Expertise level of an information source is defined as a measure of degree of ability of the information source to assess the security level of a security solution. It is represented in terms of a number called* expertise score.

Level of trustworthiness is derived from *knowledge score* and *expertise score*. The following subsections describe in details how to evaluate these scores to determine the trustworthiness of the sources and how this, together with information provided by them, is used in estimation of trust levels of the security solutions.

5.1.1 Evaluating Knowledge Score of an Information Source

The knowledge score of an information source gives a measure of how closely the knowledge of that information source is related to the desired knowledge in the problem context. The knowledge score is calculated from two scores – *reference knowledge domain score* and *information source knowledge domain score*. These two scores are derived using two models – *reference knowledge domain model* and *information source knowledge domain model*. The reference knowledge domain model provides the relative importance of different knowledge domain regarding the problem context. The information source knowledge domain model gives an assessment, by a third party, of the relative importance of knowledge level of an information source corresponding to the knowledge domains identified in reference knowledge domain model. The following sections present the details of these models and show how the corresponding two scores are calculated.

5.1.1.1 Reference Knowledge Domain Model

Estimating security level of a security solution may involve several domains where not all domains are of interest for the estimation. We develop a reference knowledge domain model that captures the domains that are of interest and their relative importance with respect to the problem context. The relative importance of a domain is measured in terms of *importance weight* which is defined as

Definition 30 **[Importance Weight]** *Importance weight of a knowledge domain is defined to be the percentage of the whole reference knowledge domain covered by that particular knowledge domain.*

For example, suppose the context is assessing the security level of a new encryption algorithm over Internet Protocol (IP). In such a case, the knowledge domains network security and IP covers the context greater than the knowledge domain authentication mechanism. The percentage value of the above problem context covered by each of the knowledge domain network security, IP, and authentication mechanism indicates their respective importance weight with respect to the problem context. Figure 5.2 shows a reference knowledge domain model consisting of four domains – domain A, domain B, domain C and domain D. All these domains cover the whole knowledge domain equally (25%), and hence have equal importance. Thus the importance weight of each domain is 0.25.

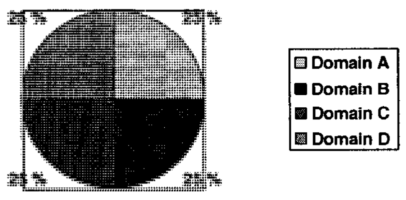


Figure 5.2: Reference knowledge domain model

Reference Knowledge domain score derives the relative importance for each knowledge domain in the reference knowledge domain model according to the problem context.

In the computation of reference knowledge domain score, we first find out the knowledge domains that are of interest for security level estimation. This can be done using various techniques. In Common Criteria evaluation, the evaluator may provide the set of knowledge domains based on his/her experience. Once we find out the knowledge domains that are of interest to the problem context, we need to find out the importance weight for each domain. Using these we calculate the reference knowledge domain score. Equations 5.1–5.5 constitute the reference knowledge domain model for evaluating reference knowledge domain score.

$$\begin{aligned}
W_{Kimp}(x) &= [w_{Kimp}(x(j))]_{j=1}^{m} && (5.1)\\
W_{allKimp}(X) &= [W_{Kimp}(x)]_{x=1}^{q} && (5.2)\\
W_{aggregatedKimp}(X) &= f_{aggregation1}(W_{allKimp}(X)) \\
&= [w_{aggregatedKimp}(X(j))]_{j=1}^{m} && (5.3)\\
f_{refKnorm} &= \frac{1}{\sum_{j=1}^{m} w_{aggregatedKimp}(X(j))} && (5.4)\\
W_{refKnowledgeDomainScore}(X) &= f_{refKnorm} \times W_{aggregatedKimp}(X) \\
&= [w_{refKnowledgedomainScore}(X(j))]_{j=1}^{m} && (5.5)
\end{aligned}$$

As already mentioned, each knowledge domain in the reference model has a particular importance weight associated to it. Note that multiple stakeholders are often involved in formalizing the context. Consequently, different stakeholders can assign different weights for importance. Suppose the stakeholders are denoted by the set X and the cardinality of the set is q. We use x to denote an individual stakeholder. Suppose m is the number of knowledge domains in the problem context. The importance of knowledge domains, from the point of view of a stakeholder x, are represented as an m-element vector. This vector is denoted by $W_{Kimp}(x)$ where $W_{Kimp}(x) = [w_{Kimp}(x(j))]_{j=1}^{m}$. For each j, $w_{Kimp}(x(j))$ denotes the importance weight of the j^{th} knowledge domain assigned by stakeholder x. This is shown by Equation 5.1 in the reference knowledge domain model. We obtain such vector for each of the stakeholders in the set X. The importance of the m different domains given by q stakeholders is presented in a $q \times m$ matrix denoted by $W_{allKimp}(X)$. Equation 5.2 gives the formula for $W_{allKimp}(X)$.

The next step is to aggregate the information obtained from q stakeholders. The aggregation can be done using different aggregation techniques. The idea is to apply an aggregation function, denoted by $f_{aggregation1}$, on the $q \times m$ matrix $W_{allKimp}(X)$ to merge the rows, resulting in a vector of size m. Equation 5.3 indicates the result of this aggregation. In Section 5.2 we aggregate them, by taking the arithmetic average for each m elements from all q vectors, into a single vector (for X) $W_{aggregatedKimp}(X)$ which is given by $[w_{aggregatedKimp}(X(j))]_{j=1}^{m}$. The arithmetic average is the simplest type of expert opinion aggregation, and thus does not capture the differences in the ability of each x to evaluate the relations between knowledge domains in a security level estimation. The reader is referred to Cooke [Coo91] and similar sources for examples of other aggregation techniques. To normalize this vector, the normalization factor is obtained using Equation 5.4. Finally, the weight of each domain in the problem context is obtained by normalizing each element in the vector $W_{aggregatedKimp}$ by the above normalization factor to obtain the vector $W_{refKnowledgeDomainScore}$. This is shown in Equation 5.5.

5.1.1.2 Information Source Knowledge Domain Model

An information source may not have knowledge in all the desired knowledge domain regarding the problem context. The information source knowledge domain model provides the relative importance of knowledge level of the information source corresponding to the knowledge domains in reference knowledge domain model. This relative importance is assessed by a third party or an expert.

Consider the reference knowledge domain example from Section 5.1.1.1. Now, for an information source, say I, a third party assessor assesses the relative importance of knowledge level of I on the identified knowledge domains as 30% on domain A, 30% on domain B, and 40% on domain D. Thus, the relative importance of I's knowledge level on the domains, as assessed by a third party, is $[0.3, 0.3, 0.0, 0.4]$.

Information source knowledge domain score derives the relative importance of knowledge level of an information source for each knowledge domain, in the reference knowledge domain. This relative importance is assessed by a third party.

Suppose we have n information sources, denoted by $b_1, b_2, \ldots, b_n$, in a security level estimation. Suppose Y is the set of third parties assessing the relative importance of knowledge domains for these n information sources. Suppose an individual third party in the set Y is denoted by y and the cardinality of Y is z. The following Equations 5.6–5.11 constitute the information source knowledge domain model for evaluating information source knowledge domain score for an information source b_i.

$$\begin{aligned}
W_{Kis}(y(b_i)) &= [w_{Kis}(y(b_i(j)))]_{j=1}^{m} && (5.6)\\
W_{allKis}(Y(b_i)) &= [W_{Kis}(y(b_i))]_{y=1}^{z} && (5.7)\\
W_{aggregatedKis}(Y(b_i)) &= f_{aggregation2}(W_{allKis}(Y(b_i)))\\
&= [w_{aggregatedKis}(Y(b_i(j)))]_{j=1}^{m} && (5.8)\\
& && (5.9)\\
f_{isKnorm} &= \frac{1}{\sum_{j=1}^{m} w_{aggregatedKis}(Y(b_i(j)))} && (5.10)\\
W_{isKnowledgeDomainScore}(Y(b_i)) &= f_{isKnorm} \times W_{aggregatedKis}(Y(b_i))\\
&= [w_{isKnowledgeDomainScore}(Y(b_i(j)))]_{j=1}^{m} && (5.11)
\end{aligned}$$

Each third party y provides a vector, denoted by $W_{Kis}(y(b_i))$, of m-elements. Each element represents the assessed importance of knowledge level of the information source b_i corresponding to the domain represented by that element. Equation 5.6 describes this. This step is then repeated for each y in the set Y and results in z such vectors. To aggregate information from all y for the information source b_i, these z vectors are first combined in a $z \times m$ matrix in Equation 5.7 and then aggregated using an aggregation function in Equation 5.9. The aggregation function is denoted as $f_{aggregation2}$ in the equation. As for the reference knowledge domain score model, the aggregation technique used is arithmetic average. We normalize this vector using the normalization factor obtained in Equation 5.10. Finally, the importance of knowledge level of each domain in the problem context is obtained by normalizing each element in the vector $W_{aggregatedKis}$ by the above normalization factor to obtain the vector $W_{isKnowledgeDomainScore}$. This is shown in Equation 5.11.

The result gives one vector for the set Y holding the information source knowledge domain scores for the information source b_i. All these steps are then repeated n times (as we have n number of information sources in the security level estimation).

5.1.1.3 Calculating Knowledge Score

The 'knowledge score' gives a measure of the knowledge level of an information source in the problem context. The reference knowledge domain score and the information source knowledge domain score are used to derive the knowledge score for each information source b_i. This is done using Equation 5.12 in the knowledge score evaluation.

Knowledge score combines the scores from the reference knowledge domain model with the scores obtained in the information source knowledge domain model to compute the measure of knowledge level of an information source in the problem context. For an information source b_i, this score is denoted by $K_{score}(b_i)$.

$$K_{score}(b_i) = \sum_{j=1}^{m}\{w_{refKnowledgedomainScore}(X(j)) \times w_{isKnowledgeDomainScore}(Y(b_i(j)))\} \tag{5.12}$$

The result from the knowledge score is a real number derived by component-wise multiplication of the two vectors $W_{refKnowledgeDomainScore}(X)$ and $W_{isKnowledgeDomainScore}(Y(b_i))$ and then adding all the product values. Here we assume that the external sources in X and Y are completely trustworthy.

5.1.2 Evaluating Expertise Score of an Information Source

The *expertise level,* captures the level of expertise of an information source. The level of expertise is represented by a score, known as the *expertise score* and is evaluated using questionnaires.

Each questionnaire consists of a set of *calibration variables* which are further divided into *categories.* Table 5.1 provides an example questionnaire with an example set of calibration variables. Each calibration variable that is pertinent to the problem context is associated with an importance value. The sum of importance values is in the range $(0, 1]$. The importance value of all the calibration variables used in a problem context equals 1. Each category in a calibration variable is also associated with a value. The importance value for each calibration variable and the value associated with each category is determined by a third party, such as an expert. There are also other ways of obtaining this information, such as experience repositories. We do not provide any further details about this. Interested readers are referred to Cooke [Coo91] and Goossens et al.[GHKM00] for an overview of the general challenges and benefits related to expert judgments.

Let $E_{score}(b_i)$ represent the expertise score of information source b_i. This score is computed as follows. Each information source is required to fill the questionnaire. Let there be p calibration variables denoted by $l_1, l_2, \ldots, l_p$ and $W_{l_1}, W_{l_2}, \ldots, W_{l_p}$ be their relative importance value. Therefore, $\sum_{t=1}^{p} W_{l_t} = 1$. Let the calibration variable l_j have q categories denoted by $l_{j1}, l_{j2}, \ldots, l_{jq}$. Let $w_{cat}(l_{jm}) \in [0,1]$ represent the weight acquired by b_i in the category m of calibration variable l_j. Equation 5.13 gives the value obtained by b_i for calibration variable l_j. The sum of the values of all these calibration variables gives the expertise score of b as demonstrated by Equation 5.14.

$$W_{calib}(l_j) = W_{l_j} \times \sum_{m=1}^{q} w_{cat}(l_{jm}) \tag{5.13}$$

$$E_{score}(b_i) = \sum_{t=1}^{p} W_{calib}(l_t) \tag{5.14}$$

5.1.3 Computing Information Source Trustworthiness

For an information source b_i, the knowledge score and the expertise score basically provide measures for *recommendation* and *properties.* In our approach we have not used the parameters *interactions* and *reputation,* though use of interaction in evaluation of trust on information source is not ruled out. Therefore, in trust-parameter weight policy vector we include only two weight values corresponding to the above parameters. Suppose we denote these two by k and e respectively, where $0 \leq k, e \leq 1$ and $k + e = 1$. Hence, trustworthiness

Variables	Categories
level of expertise	low, medium and high
age	under 20, [20-25), [25-30), [30-40), [40-50), over 50
years of relevant education	1 year, 2 years, Bsc, Msc, PhD, other
years of education others	1 year, 2 years, Bsc, Msc, PhD, other
years of experience from industry	[1-3) years, [3-5) years, [5-10) years, [10-15) years, over 15 years
years of experience from academia	[1-3) years, [3-5) years, [5-10) years, [10-15) years, over 15 years
role experience	database, network management, developer, designer, security management and decision maker

Table 5.1: Example of calibration variables for determining the expertise level for an information source

of an information source b_i is computed as

$$\mathrm{v}(A \xrightarrow{c} b_i)_t^N = k \times K_{score}(b_i) + e \times E_{score}(b_i) \tag{5.15}$$

For simplicity reason, we do not involve trust dynamics in our evaluation.

5.1.4 Computing Trust Level of a Security Solution

The trust level of an information source is used to compute the trust level of the security solutions. The information obtained from each source b_i, denoted by $b_i(I)$ is considered to be a *recommendation* about the security solution and is scaled with the trust value of the recommender (information source). This gives the trust level for the security solution, say s_c. That is, we use only *recommendation* (from information source) to evaluate trust level of the security solutions. However, *interactions* and *properties* can also be used. The characteristic attributes of the security solutions (e.g., cookie solution is an OS level solution) can be considered as properties whereas their performance as a security solution can be used to measure *interactions*. However, for our case the trust level is computed as

$$\mathrm{v}(A \xrightarrow{c'} s_c)_t^N = \frac{\sum_{j=1}^{n}(\mathrm{v}(A \xrightarrow{c} b_i)_t^N) \times b_i(I)}{\sum_{j=1}^{n}(\mathrm{v}(A \xrightarrow{c} b_i)_t^N)} \tag{5.16}$$

5.2 Evaluation

In our evaluation, we have five information sources; one honeypot [Øst03] and four domain experts from a pool of 18 domain experts (undergraduate students at Norwegian University of Science and Technology). The four chosen domain experts are denoted as b_4, b_6, b_{15}, b_{18} and the honeypot is denoted by $b_{honeypot}$. These five information sources provide information on the anticipated number of DoS attacks for the two involved solutions to A^2.

The information source honeypot was set and logging is done for three different configurations: (1) system without any security solutions, (2) system with the patch to the network stack software (the cookie solution), and (3) system with the filtering mechanism. For each configuration logging is done for 24 hours and only the connection attempts to TCP port 80 (intended for the web server (IIS 4.0)) is considered. For configuration where the system does not have any security solution, Snort detected 470 different IP-addresses trying to make a connections to port 80. This gives 470/24 = 19.8 attack tries per hour (This was done by penetrating the network sending SYN requests to ranges of IP-addresses. In this case we are interested in the attack attempts where outsiders sent several SYN requests to the same source.). 140 out of 470 IP-addresses have sent series of SYN requests to the same source within 24 hours. This shows that there are 140/24 = 5.8 SYN flooding attack tries per hour. It is found that two of the attack tries were successful. Consequently, the mean time to misuse, MTTM (a measure for how long in clock time, on average, takes between attacks), is 12 hours. The mean attack time (effort) for the successful attacks, denoted by mean effort to misuse (METM, a measure of how much time, on average, it takes an attacker to perform the attack), was found to be 0.2 hours. In this case we are consider successful attacks and not attack tries[3]. Average monthly (assuming a 30 day month) successful attacks is computed as 60. For the cookie solution, the average monthly successful attacks observed was 1.5 and For the filtering mechanism, the average monthly successful attacks observed was 4.0. Since the honeypot simulates these three configurations, no update was done between the attack tries and successful attacks. However, this is usually done in a real

[2]For our evaluation, the truster A is one of the authors of the work [HCRR08].

[3]More information on MTTM and METM is given in Houmb et al. [HGF$^+$05b].

systems, otherwise the system would just keep on being attacked until it stops functioning properly.

The result of the logging are then used as the information provided by the information source honeypot ($b_{honeypot}$) when evaluating the two DoS security solutions. Honeypot observes actual events and in our case, it is under direct control of the truster A. Therefore, A has complete trust in the abilities of honeypot to provide accurate and correct information on the potential number of successful DoS attacks. In other words, $\mathrm{v}(A \xrightarrow{c} b_{honeypot}) = 1$. This means that we do not need to calculate the knowledge and expertise score for honeypot and that we assign the trust value directly.

Elicitation of expert judgments are done using a combined knowledge level and expertise level questionnaire. The information provided on each expert for all variables in the questionnaire is then used to derive the knowledge score and expertise score, as described in Section 5.1.1 and Section 5.1.2 respectively. Table 5.2 shows the variables and the values provided for the combined questionnaire. As described in section 5.1.1, the knowledge

Expert number	**Calibration variable**	**Information provided**
4	level of expertise	medium
	years of relevant of education	Bsc
	years of experience from industry	0
	role experience	database and security management
6	level of expertise	low
	years of relevant of education	Bsc
	years of experience from industry	0
	role experience	database
15	level of expertise	high
	years of relevant of education	Bsc
	years of experience from industry	0
	role experience	designer, developer and security management
18	level of expertise	low
	years of relevant of education	Bsc
	years of experience from industry	0.5
	role experience	developer

Table 5.2: The combined knowledge and expertise level questionnaire and the information provided

score of an information source is determined by comparing the information source knowledge domain model with the reference knowledge domain model. The reference knowledge domain model is created with the importance weights for the aggregated set of knowledge domains provided by the set of external sources X (using the reference knowledge domain score model). Here the relevant knowledge domains are *security management, design, network manager, database*, and *developer.* These domain are identified by a domain expert, in our case the truster A, based on her prior experience and knowledge in secure system development. A also works as an x in X and a y in Y as mentioned in Section 5.1.1. Figure 5.3 shows the five knowledge domains and their corresponding importance weights. For demonstration purpose the focus is put on the result of the identification, rather than discussing techniques that can be used to identify these knowledge domains. However, as described earlier the knowledge domains are derived from the problem definition, the system configuration, and the system environment.

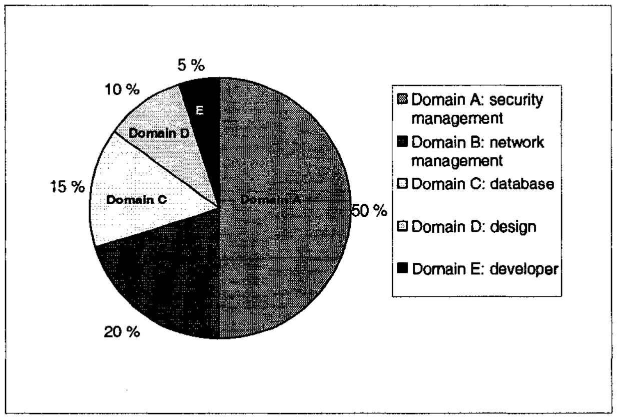


Figure 5.3: The reference knowledge domain model

As can be seen in the figure, the importance weights for different knowledge domains in the reference knowledge domain model are, 50% for security management, 20% for network management, 15% for database, 10% for design and 5% for developer. Thus, the importance vector is modeled using Equation 5.1 in the knowledge domain score model as $W_{Kimp}(x) = [0.5, 0.2, 0.15, 0.1, 0.05]$. Since we have only one external source x, we obtain, $W_{aggrgatedKimp}(X) = W_{allKimp}(X) = W_{Kimp}(x)$. As can be seen in the figure, the knowledge domains are already normalized with each other and hence we do not need to normalize the elements in the vector $W_{aggrgatedKimp}(X)$. Hence, $W_{refKnowledgeDomainScore}(X) = W_{aggrgatedKimp}(X) = [0.5, 0.2, 0.15, 0.1, 0.05]$.

Figure 5.4 shows the information source knowledge domain models for the four experts (information sources). The importance weight that each of the experts has for the knowledge domains are: for expert 4, 85% on security management and 15% on database; for expert 6, 100% on database; for expert 15, 60% on design, 30% on developer, and 10% on security management; for expert 18, 100% on developer. Equation 5.6 in the the information source knowledge domain score model gives these information source knowledge domain vectors as,

- Expert-4 (b_4): $W_{Kis}(y(b_4)) = [0.85, 0.0, 0.15, 0.0, 0.0]$
- Expert-6 (b_6): $W_{Kis}(y(b_6)) = [0.0, 0.0, 1.0, 0.0, 0.0]$
- Expert-15 (b_{15}): $W_{Kis}(y(b_{15})) = [0.1, 0.0, 0.0, 0.6, 0.3]$
- Expert-18 (b_{18}): $W_{Kis}(y(b_{18})) = [0.0, 0.0, 0.0, 0.0, 1.0]$

Since there is only one external source y in the set Y of external sources providing information on the information sources, we have $W_{isKnowledgeDomainScore}(y(b_i)) = W_{aggregatedKis}(y(b_i)) = W_{Kis}(y(b_i))$, $\forall i = 4, 6, 15, 18$.

The knowledge score for each of the information source are then derived using equation 5.12 in the knowledge score model as follows:

- Expert-4 (b_4): $K_{score}(y(b_4)) = 0.85*0.5+0*0.2+0.15*0.15+0*0.1+0*0.05 \approx 0.45$
- Expert-6 (b_6): $K_{score}(y(b_6)) = 0*0.5+0*0.2+1*0.15+0*0.1+0*0.05 = 0.15$
- Expert-15 (b_{15}): $K_{score}(y(b_{15})) = 0.1*0.5+0*0.2+0*0.15+0.6*0.1+0.3*0.05 \approx 0.13$
- Expert-18 (b_{18}): $K_{score}(y(b_{18})) = 0*0.5+0*0.2+0*0.15+0*0.1+1.0*0.05 = 0.05$

The level of expertise of an information source is derived using the calibration variables described in Table 5.2. First, we obtain the importance values for all calibration variables and the weights of categories for each calibration variable from the external expert A. We assume that a set of aggregated values is obtained for category weights as well as calibration

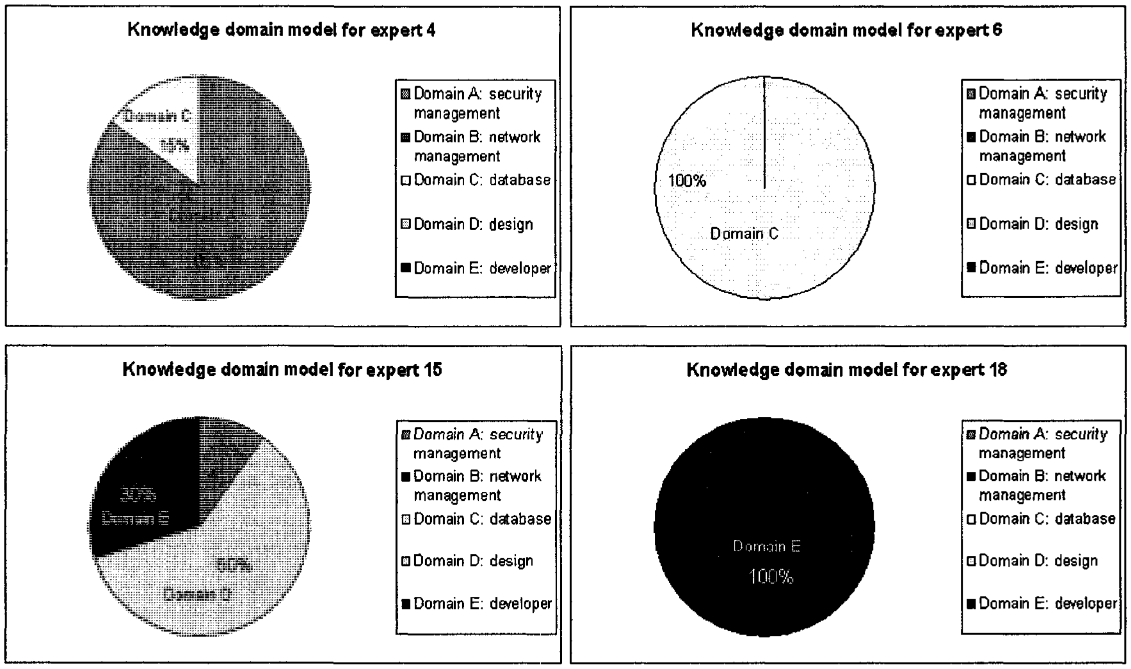


Figure 5.4: Information source knowledge domain model for expert 4, 6, 15, and 18

variable importance values from the external expert. Here we use three calibration variables to determine level of expertise – *level of experience* denoted by l_1, *years of relevant education* denoted by l_2, and *years of experience from industry* denoted by l_3. Since we use subset of the variables, we only include the categories that are actually used in the experiment (not all categories and calibration variables from Table 5.1 are relevant because the experts are undergraduate students). This gives the following vectors of categories for the three calibration variables: (i) $l_1 = [low, medium, high]$, (ii) $l_2 = [Bsc]$, (iii) $l_3 = [no\_of\_year]$

The next thing to look into is the importance weight for each of the categories for all calibration variables. The expert assigns the following values: $w_{cat}(l_1(low)) = 0.2$, $w_{cat}(l_1(medium)) = 0.5$, $w_{cat}(l_1(high)) = 1.0$, $w_{cat}(l_2(Bsc)) = 0.2$ and $w_{cat}(l_3(no\_of\_year)) = 0.2$ for each year of industrial experience. Therefore, $w_{cat}(l_1) = [0.2, 0.5, 1.0]$, $w_{cat}(l_2) = [0.2]$, and $w_{cat}(l_3) = [0.2]$.

Suppose the importance value given to the calibration variables by the external expert A are 0.3 for *level of experience*, 0.2 for *years of relevant education*, and 0.5 for *years of experience from industry*. Therefore, $W_{l_1} = 0.3$, $W_{l_2} = 0.2$ and $W_{l_3} = 0.5$.

We then look at the information about categories of calibration variables provided by the information sources b_4, b_6, b_{15}, b_{18} in the questionnaire. Suppose the information is as follows:

- Expert-4 (b_4): $w_{cat}(l_1) = [medium]$, $w_{cat}(l_2) = [Bsc]$, $w_{cat}(l_3) = [0]$.
- Expert-6 (b_6): $w_{cat}(l_1) = [low]$, $w_{cat}(l_2) = [Bsc]$, $w_{cat}(l_3) = [0]$.
- Expert-15 (b_{15}): $w_{cat}(l_1) = [high]$, $w_{cat}(l_2) = [Bsc]$, $w_{cat}(l_3) = [0]$.
- Expert-18 (b_{18}): $w_{cat}(l_1) = [high]$, $w_{cat}(l_2) = [Bsc]$, $w_{cat}(l_3) = [0.5]$.

Using the above information and the weights for categories and calibration variables, the truster calculates the expertise score of the information sources as (using Equations 5.13 and 5.14),

- $E_{score}(b_4) = 0.3 * 0.5 + 0.2 * 0.2 + 0.5 * 0 = 0.15 + 0.04 + 0 = 0.19$
- $E_{score}(b_6) = 0.3 * 0.2 + 0.2 * 0.2 + 0.5 * 0 = 0.06 + 0.04 + 0 = 0.10$
- $E_{score}(b_{15}) = 0.3 * 1.0 + 0.2 * 0.2 + 0.5 * 0 = 0.3 + 0.04 + 0 = 0.34$
- $E_{score}(b_{18}) = 0.3 * 1.0 + 0.2 * 0.2 + 0.5 * (0.5 * 0.2) = 0.3 + 0.04 + 0.5 = 0.84$

We have now derived the knowledge and expertise score for all five information sources. These two scores are then used to evaluate trust of the information sources using Equation 5.15, described in Section 5.1.3. For this purpose, suppose the truster A has the trust-parameter weight policy vector as $(0.6, 0.4)$ that is, $k = 0.6$ and $e = 0.4$. Recall that the trust value for the information source honeypot was set to 1. Thus, the trustworthiness score for the experts $b_{honeypot}$, b_4, b_6, b_{15}, b_{18} are computed as,

- $\mathrm{v}(A \xrightarrow{c} b_{honeypot})_t^N = 1.0$.
- $\mathrm{v}(A \xrightarrow{c} b_4)_t^N = 0.6 * 0.45 + 0.4 * 0.19 = 0.27 + 0.076 = 0.346$.
- $\mathrm{v}(A \xrightarrow{c} b_6)_t^N = 0.6 * 0.15 + 0.4 * 0.1 = 0.09 + 0.04 = 0.130$.

- $\mathrm{v}(A \xrightarrow{c} b_{15})_t^N = 0.6 * 0.13 + 0.4 * 0.34 = 0.078 + 0.136 = 0.214.$
- $\mathrm{v}(A \xrightarrow{c} b_{18})_t^N = 0.6 * 0.05 + 0.4 * 0.84 = 0.03 + 0.336 = 0.366.$

The above values show the trust level the truster A has on the information sources (recommenders). These sources provide information (recommendation) about the security solutions. The honeypot provides number of average monthly successful attacks for two solutions. The experts (information sources) provide their judgment about the security solutions using terms *low*, *medium*, or *high*. In order to calculate the trust level of the security solutions from these pieces of information, the information must be at the same level of abstraction and comparable. The truster interprets and transforms the information from the honeypot as well as from the experts according to her policies. The honeypot reports less number of average monthly successful attack for cookie solution than filter mechanism. This shows that according to the information source $b_{honeypot}$, the cookie solution s_c has higher security level. She transforms the average monthly successful attack inversely and the reciprocal of this average value is used as recommendation value from $b_{honeypot}$. This gives: $b_{honeypot}(s_c) = 1/1.5 = 0.667$ and $b_{honeypot}(s_f) = 1/4.0 = 0.25$. Recommendation from the other information sources (experts) are summarized in the following table: The

Expert	Reco. about s_c	Reco. about s_f
b_4	*medium*	*low*
b_6	*medium*	*medium*
b_{15}	*medium*	*low*
b_{18}	*high*	*low*

Table 5.3: Recommendation about the security solutions by the experts

decision maker translates the judgment according to her policies as follows: *low* = 0.2, *medium* = 0.5, and *high* = 1.0. Hence the trust level of the security solutions s_c and s_f are computed using Equation 5.16 as

$$\begin{aligned}
\mathrm{v}(A \xrightarrow{c'} s_c)_t^N &= \frac{0.667 * 1.0 + 0.5 * 0.346 + 0.5 * 0.130 + 0.5 * 0.214 + 1.0 * 0.366}{1.0 + 0.346 + 0.130 + 0.214 + 0.366} \approx 0.67 \\
\mathrm{v}(A \xrightarrow{c'} s_f)_t^N &= \frac{0.25 * 1.0 + 0.2 * 0.346 + 0.5 * 0.130 + 0.2 * 0.214 + 0.2 * 0.366}{1.0 + 0.346 + 0.130 + 0.214 + 0.366} \approx 0.24
\end{aligned}$$

This trust level indicates the truster's level of assurance on the security level of a security solution. This is just a measure of assurance that the decision maker has on the security solution and should not be considered as the actual security level of a security solution. There are several reason for this, one being the interpretation and transformation that is done. Furthermore, the actual security level is a future event that depends on many factors, such as the security environment, relevant operational procedures, maintenance strategy, the resources available etc. What we can infer from our trust-based evaluation of the two security solutions is that the cookie solution is a more trusted choice than the filtering mechanism when it comes to preventing denial of service attacks in our testbed ACTIVE (the .NET e-commerce system).

5.2.1 Discussions on the Evaluation

DoS attacks are becoming more sophisticated and hence increasingly difficult to detect and protect against. Such attacks are often performed using legitimate or expected protocols and services as the main attack vehicle. Hence the malicious requests differ from legitimate requests only by intent and not by content. Because it is intriguingly hard to measure intent, many of the existing DoS solutions do not offer a proper defense. Many solutions are deployed on the network device level, such as the filtering mechanism. However, filtering on the network device level has been demonstrated as being infeasible to deploy in an effective manner [LcC06]. In fact, filtering against a defined legitimate or expect type of traffic may even contribute in completing the attacker's task by causing legitimate services to be denied [LcC06].

In [KL01] Karig and Lee gives an overview of common DoS attacks and potential countermeasures for DoS attacks. In this context, the filtering mechanism is categorized as a network device level countermeasure while the cookie solution is categorized as an OS level countermeasure. A network device level DoS solution provides measures to protect against potential misuse of a communication protocol. Thus, the protection is often on the IP or transport layer and hence there are possible ways around the mechanism, such as those discussed in [KL01]. The main shortage of filtering mechanisms are their inability to filter out spoofed packets [KL01]. There are, however, more efficient filtering mechanisms available, such as the one discussed in [BLG+00].

The other DoS solution considered here, the cookie solution, operates on the OS level. An OS level DoS solution integrates protection into the way a protocol is implemented in a particular operating system. Thus, the measure is deployed on the source (target) and refers to a host-based protection solution. Hence, the cookie solution represent a more defense-in-depth DoS solution than the filtering mechanism. Furthermore, the cookie solution discussed here is a SYN cookie, which has been well tested and is well understood. SYN cookies have also been incorporated as a standard part of Linux and Free BSD and are recognized as one of the most effective DoS mechanisms [Ber06].

The above discussion implies that cookie solution is a better security solution than filtering mechanism to protect against denial of service attacks. This fact is corroborated by the findings in our trust-based evaluation of the security solutions.

We also compare the results obtained by considering the information obtained from the experts only and the result obtained by the honeypot. Rationale is, the honeypot observes actual events and evaluates the trustworthiness of the security solutions based on these events. In our evaluation, we really do not use the trust-based scheme to scale the information from the honeypot as we have assigned absolute trust on the honeypot. Alternately, the information obtained from the other information sources (experts) are scaled with their respective trustworthiness measured using their properties and recommendations. Hence, it will be interesting to see whether a trust-based evaluation of the security solutions based on the information obtained from domain experts can match the actual finding of the honeypot. Using the honeypot information we have a value 0.667 for the security solution s_c and a value 0.25 for the security solution s_f. In relative terms s_c is $\frac{0.667}{0.25} \approx 2.67$ times better than s_f. Considering only the domain experts in trust-based evaluation, we have

$$\begin{aligned}
\mathbf{v}(A \xrightarrow{c'} s_c)_t^N &= \frac{0.5 * 0.346 + 0.5 * 0.130 + 0.5 * 0.214 + 1.0 * 0.366}{0.346 + 0.130 + 0.214 + 0.366} \approx 0.673 \\
\mathbf{v}(A \xrightarrow{c'} s_f)_t^N &= \frac{0.2 * 0.346 + 0.5 * 0.130 + 0.2 * 0.214 + 0.2 * 0.366}{0.346 + 0.130 + 0.214 + 0.366} \approx 0.237
\end{aligned}$$

According to the above results, in relative measure, s_c is $\frac{0.673}{0.237} \approx 2.84$ times better than s_f. This again shows that trustworthiness evaluation using the vector trust model matches with that of based on actual findings.

Chapter 6

Application Scenarios

The primary objective of developing the vector trust model is to use it for making reasoned trust-based decisions in different security contexts. Next we discuss three application scenarios where the proposed trust model is used as a decision-aid in the corresponding security contexts.

6.1 Trust-based Access Control Mechanism

Conventional access control models like role based access control are suitable for regulating access to resources by known users. However, these models have often found to be inadequate for open and decentralized multi-centric systems where the user population is dynamic and the identity of all users are not known in advance. For such systems, credential based access control has been proposed. Credential based systems achieve access control by implementing a binary notion of trust. If a user is trusted by virtue of successful evaluation of its credentials it is allowed access, otherwise not. However, such credential based models have also been found to be lacking because of certain inherent drawbacks with the notion of credentials. In [CR06], we propose a trust based access control model called TrustBAC. It extends the conventional role based access control model with the notion of trust levels. Users are assigned to trust levels instead of roles based on a number of factors like user credentials, user behavior history, user recommendation etc. using the vector trust model. Trust levels are assigned to roles which are assigned to permissions as in role based access control. The TrustBAC model thus incorporates the advantages of both the role based access control model and credential based access control models.

6.1.1 Motivation

Proper access control to resources is one of the major concerns for any organization. Different models of access control have been proposed over the years, for example, discretionary and mandatory access control models, Clark-Wilson model, Task based models and Role Based Access Control model. Among these, role based access control (RBAC) [FSG+01] is gradually emerging as the standard for access control. The main advantage of RBAC over other access control models is the ease of security administration. In the RBAC model access permissions are not assigned directly to the users but to abstractions known as "roles". Roles correspond to different job descriptions within an organization. Users are assigned to different roles and, thus, indirectly receive the relevant permissions. Thus, with RBAC, security is managed at a level corresponding to an organization's human resource structure.

Notwithstanding the success of the RBAC model, researchers have often found the model to be inadequate for open and decentralized multi-centric systems where the user population is dynamic and the identity of all users are not known in advance. Examples of such systems are service providers operating over open systems like the Internet. It is almost impossible to know beforehand all the users that will request services in these systems. Assigning appropriate roles to these users thus becomes an irrational and ad-hoc exercise. To overcome the shortcomings of RBAC for such systems, researchers have proposed credential-based access control models [BFL96, BFIK99, LM03a]. Credentials implement a notion of binary trust. Here the user has to produce a predetermined set of credentials (for example, credit card numbers or proof of membership to certain groups etc.) to gain specific access privileges. The credential provides information about the rights, qualifications, responsibilities and other characteristics attributable to its bearer by one or more trusted authorities. In addition, it provides trust information about the authorities themselves. Researchers have also integrated credential based access control with role-based access control to facilitate security administration [LMW02, COB03, LM03b, LWM03].

Although credential based models solve the problem of access control in open systems to a great extent, it still has a number of shortcomings. A credential, strictly speaking, does not bind a user to its purported behavior or actions. It does not guarantee that its bearer really satisfies the claims in the credential. It does not convey any information about the behavior of the bearer between the time the credential was issued and its use. A credential does not reveal whether it was obtained via devious means. In real life some or all such information may play crucial parts in access control decisions. Additionally, credential based access control does not keep track of the user's behavior history. Access permission is given on the basis of the credential presented for a particular session. Either the user's credentials are accepted and required privileges are allowed, or the credentials are rejected and the user does not get the access rights. Thus, good behavior by the user cannot be rewarded with enhanced privileges nor bad behavior be punished.

The above observations motivate us to revisit the problem of access control in decentralized and multi-centric open systems. We believe credential based access control is a step in the right direction. However, we would like to enhance the binary trust paradigm in these models with our much richer multi-level trust model. In this new trust-based access control model, trust levels of the users can be determined not only by using the credentials (*properties*) presented by the user but also from the results of past interactions (*interactions*) with the user, from recommendations about the user and/or knowledge about other characteristics of the user (*properties*). A user is mapped to different trust levels based on these information. Trust levels (and not users, unlike in conventional RBAC) are then mapped to roles of RBAC. Thus our access control model is an enhanced RBAC (TrustBAC). Changes in the trust level of user changes the roles that the user has in the system and thus the user's privileges. Since our vector trust model provides multiple levels of trust, the system can define as many trust levels as it wants and can assign each level to a specific set of resources tied with a specific set of access privileges. The system just needs to monitor the trust level of the user and the regulation of access is automatically achieved.

6.1.1.1 Background

Role-based Access Control (RBAC) was first introduced by Ferraiolo and Kuhn in [FK92] to address the limitations of discretionary access control model (DAC). Sandhu et al. [SCFY96] introduce four reference models to provide systematic approach to understand RBAC model. Their framework separates administration of RBAC from its use for controlling access. They also categorize the implementation of RBAC in different systems. Finally, after a series of modifications, the NIST standard for RBAC is proposed in [FSG$^+$01]. The standard specifies RBAC reference model which defines the scope of features that comprise the standard and provides terminologies to support specification. The standard also specifies system and administrative functional specification which defines functional requirements for administrative operations and system level functionalities. We have followed the approach of the standard to formalize our TrustBAC model.

A main feature of the TrustBAC model is the use of user behavior history in determining access privileges. Similar concept of using execution history in access control can be found in literature. In [EAC98] the authors present a history-based access control mechanism which is suitable for controlling access from mobile code. The scheme maintains a selective history of access requests made by individual program and use this history to measure the degree of safeness of a request. Another history based access control for codes is presented in [AF03]. In this scheme the access privileges of a code is determined in runtime by examining the attributes of the pieces of code that have run before. The pieces of code that have run includes the codes on stack as well as the codes that have been called and returned. Also, ours is not the only work which uses the concept of trust in access control. Sandhu et. al in a recent paper [SZ05] present a trusted computing architecture to enforce access control policies in peer-to-peer environment.

In [Tho97], Thomas introduces the notion of TeaM-based Access Control (TMAC) as an approach to applying role-based access control in collaborative environments. The "team" is an abstract container that encapsulates a set of users with specific roles. A team is formed with the objective of accomplishing a specific task. One advantage of TMAC model is it allows role-based permissions across object types as well as fine-grained, identity-based control on individual users in certain roles and to individual object instances. Georgiadis et al. [GMPT01] extend TMAC model by integrating it with RBAC and using it for general contextual information like time of access, location from which access is requested, location of the object for which access is requested etc. Somewhat similar idea is presented in the YGuard access control model [vdASC01]. YGuard employs a set-based access control list where a group of subjects is authorized to access sets of objects. That is, for those objects, every member of the group has same access privileges. Another similar approach is coalition-based access control (CBAC) model [CTWS02] where a coalition (group of members) shares data with their partners while ensuring that their resources are safe from inappropriate access. They define the protection state of a system, which provides the semantics of CBAC-based access policies. Researches are still continuing on issues and applications of RBAC model. Some recent works includes [WL04] which presents a multi-layered, distributed and location-dependent approach to RBAC; Park and Hwang [PH03b] present a scheme in P2P environment where RBAC mechanisms are dynamically supported based on each peer's current context; in GEO-RBAC [BCD05], Bertino et al. extend the RBAC model to deal with spatial and location-based information.

Digital library is one of the major application areas of access control in open environments. There are quite a few work on authorization issues in digital library domain. In one of the early work on access control in digital libraries, H.M. Gladney proposes a scheme called DACM (Document Access Control Methods) in [Gla97]. The basic idea is biased towards discretionary access control with some extensions to handle mandatory access control. Winslett et al. [WCJS97] propose a mechanism to assure security and privacy for digital library transactions. This is basically a credential-based system where both client and server can specify their own policies regarding credential disclosure and security. Client uses a personal security assistant (PSA) module and the server uses server security assistant (SSA) to manage credentials and credential acceptance policies. In [AABF02], Adam et al. propose a content-based authorization model for digital library environments. Authorization is specified based on positive and negative qualifications and characteristics of the user.

Credentials are associated with each user and represent qualifications and characteristics of the user.

There are a number of works that specifically address access control on the Internet. Bonatti and Samarati [BS00] propose a uniform formal framework to regulate service access and information disclosure on the Internet. The regulation is based on credentials. The framework includes mechanism to treat information which are not in the form of a certificate and is needed for required access. It has comprises a language for expressing access and release policies. The framework also has a policy filtering mechanism which helps the entities involved in the communication to exchange their requirements in a concise and privacy preserving way. In [BSF02] Bauer et al. describe the design, implementation, and performance of a system to control access on web. The system is based on proof-carrying authorization (PCA). The access control model provides locating and using pieces of the security policy that have been distributed across hosts and keeping the policies hidden from unauthorized clients. It also provides iterative authorization by which a server can require a browser to prove a series of challenges. In [HJ03] the authors address the problems of access control in large open systems where the authenticated identity of an entity does not provide any information regarding the likely behavior of that entity. Their scheme, called cryptographic access control, is based on cryptography to guarantee confidentiality and integrity of objects stored in potentially untrusted servers in the system.

In [BJBG03], the authors present X-RBAC, an XML-based RBAC policy specification framework to deal with access control issues in dynamic XML-based web services. It is based on a policy specification language which addresses security issues, specifically in web services. They extend X-RBAC to a trust-enhanced version in [BBG04] where the role assignment is based on trust. They define 'trust' as "the level of confidence associated with a user based on certain certified attributes". Unlike our model, this trust level is not quantitatively measured. Instead, the authors use the trust management approach of trusted third parties (e.g., public key encryption certification authority) and use the certificate provided by the third party to assign roles to the users. They also argue that traditional access control schemes following identity or capability-based approach for authorization do not scale well to the distributed web services architecture. To overcome this limitation they describe a mechanism to configure X-GTRBAC to provide context-aware trust-based access control in Web services. X-GTRBAC is a framework based on Generalized Temporal Role Based Access Control – GTRBAC [JBLG05] It provides a generalized mechanism to express a wider range of temporal constraints including periodic as well as duration constraints on roles, user-role assignments, and role-permission assignments. The X-GTRBAC extends GTRBAC with XML to allow policy enforcement in heterogeneous and distributed environment.

In a current survey [DVS05], the present state and future trends in the access control are discussed. The survey shows that the new trend in access control are part of future communication infrastructure supporting mobile computing.

6.1.2 TrustBAC model

The TrustBAC model is defined in terms of a set of elements and relations among those elements. The elements are of the following types: *user, user_properties, session_instance, session_type, session, session_history, trust_level, role, object, action, permissions* and *constraint.* The corresponding sets are USERS, USER_PROPERTIES, SESSION_INSTANCES, SESSION_TYPES, SESSIONS, SESSION_HISTORY, TRUST_LEVELS, ROLES, OBJECTS, ACTIONS, PERMISSIONS and CONSTRAINTS. The TrustBAC model is illustrated in Figure 6.1 (we use one-directional arrows to represent one-to-many relationships, two directional arrows to denote many-to-many relationships and plain lines to denote one-to-one relationships). We define the different elements as follows.

user A user $\in$ USERS is defined as a human being. The notion of user can be extended to include systems, or intelligent agents, but for simplicity we choose to limit a *user* to a human entity.

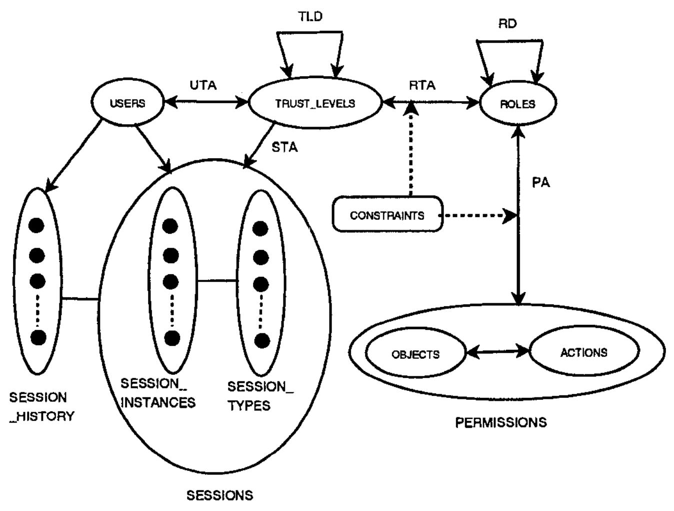


Figure 6.1: TrustBAC model

user_properties Each user u has certain set of properties $\mathscr{P}_u$, called *user_properties*. The set USER_PROPERTIES = $\bigcup_{u \in USERS} \mathscr{P}_u$. A user can manifest any subset P of $\mathscr{P}_u$ (i.e., $P \in 2^{\mathscr{P}_u}$) at a particular session.

session_instance A session_instance $\in$ SESSION_INSTANCES is a 'login' instance of an user. A user can instantiate multiple login thereby initiating multiple session_instances at the same time. A session_instance is uniquely identified by a system generated id.

session-type A session_instance is identified with a type which is determined by the set of properties manifested in that session_instance by the user invoking that session_instance. For a session_instance s invoked by a user u with P ($P \subseteq \mathscr{P}_u$) properties, has the session_type P. Formally, the set SESSION_TYPES = $2^{USER\_PROPERTIES}$.

session A session $\in$ SESSIONS is identified by a session_instance with a session_type. A session with session_instance s of type P is denoted by the symbol s^P. Formally, $SESSIONS = SESSION\_INSTANCES \times SESSION\_TYPES$.

session_history A session_history $\in$ SESSION_HISTORY is a set of information regarding the user's behavior and trust level in a previous use of a session of that type.

trust_level A trust_level is a set of real number between -1 and +1. A user, at some instant of time with a particular session has a trust_level. The set TRUST_LEVELS is the set of possible subsets of [-1, 1]. That is, TRUST_LEVELS = $\{S \mid S \subseteq [-1, 1]\}$. Thus TRUST_LEVELS becomes an infinite set where each member S can be either discrete or continuous.

role The concept of role is same as in the RBAC model. A role $\in$ ROLES is a job function with some associated semantics regarding the responsibilities conferred to a user assigned to the role.

object An object $\in$ OBJECTS is a data resource as well as a system resource. It can be thought of as a *container* that contains information.

action An action $\in$ ACTIONS is an executable image of a program. 'read', 'write', 'execute' are examples of a typical action.

permission A permission $\in$ PERMISSIONS is an authorization to perform certain task within the system. It is defined as a subset of OBJECTS $\times$ ACTIONS that is, $PERMISSIONS = 2^{(OBJECTS \times ACTIONS)}$. Therefore, a permission $= \{(o, a) \mid o \in OBJECTS, a \in ACTIONS\}$. Permissions are assigned to a role. The type of a permission depends on the nature of the system. The model does not dictate anything about the type.

constraint We borrow the concept of constraint from RBAC model. Therefore, a constraint $\in$ CONSTRAINTS is defined as a predicate which applied to a relation between two TrustBAC elements returns a value of "acceptable" or "not-acceptable". Constraints can be viewed as conditions imposed on the relationships and assignments.

Association between any two of the above elements are specified by mathematical relations. TrustBAC has the following relations.

1. *sua* : USERS $\times$ SESSION_INSTANCES $\times$ SESSION_TYPES $\rightarrow$ SESSIONS defines the *user-session* assignment relation. $sua(u, s, P) = s^P$ for $u \in$ USERS, $s \in$ SES-

SION_INSTANCES, $P \in$ SESSION_TYPES, and $s^P \in$ SESSIONS shows that a single session s^P of type P is associated with a single user u with certain properties P. A user can invoke multiple sessions of different types simultaneously.

2. UTA $\subseteq$ USERS $\times$ TRUST_LEVELS defines the *user-trust_level* assignment relation. It is a many-to-many relation where a user can have multiple trust levels. Since a user can invoke many sessions at a time, she can have different trust levels, one for each invoked session. A single trust_level can be assigned to many users. The restriction on a member $(u, L) \in$ UTA is L must be a singleton member of TRUST_LEVELS i.e., $L = \{l\},\ l \in [-1, 1]$.

3. STA $\subseteq$ SESSIONS $\times$ TRUST_LEVELS defines the *session-trust_level* assignment. It is a one-to-many relation where a session can have only one trust value. That is, the trust_level L corresponding to that session is a singleton member of TRUST_LEVELS. But many sessions can have the same trust_level.

4. RTA $\subseteq$ ROLES $\times$ TRUST_LEVELS defines the *role-trust_level* assignment relation. It is also a many-to-many relationship where a trust_level can be associated with many roles and same role can be performed with different trust_levels.

5. The function *ush*: USERS $\times$ SESSIONS_TYPES $\rightarrow$ SESSION_HISTORY defines a three-way relation between a user, a session_type and the trust history of the user in an earlier use of a session of that type. $ush(u, P) = {}_u h^P$, where $u \in$ USERS and $P \in$ SESSION_TYPES. A session_history ${}_u h^P$ is associated with a single user u and any session s^P of type P. A user can have many session_histories as a user can invoke many sessions of different types.

6. PA $\subseteq$ PERMISSIONS $\times$ ROLES is a many-to-many permission to role assignment relation. An element in PA is of type (p, r) where $p \in$ PERMISSIONS and $r \in$ ROLES.

7. The function $Assigned\_Roles : TRUST\_LEVELS \rightarrow 2^{ROLES}$ specifies the mapping of a trust_level $L(\subseteq [-1, 1])$ onto a set of roles. Formally, $Assigned\_Roles(L) =$

$\{r \in ROLES \mid (r, L) \in RTA\}$. It implies, for any $l \in L, Assigned\_Roles(\{l\}) = Assigned\_Roles(L)$.

8. The function $Assigned\_Permission : ROLES \rightarrow 2^{PERMISSIONS}$ specifies the mapping of a role r onto a set of permissions. Formally, $Assigned\_Permission(r) = \{p \in PERMISSIONS \mid (p, r) \in PA\}$. This function is same as *assigned_permissions* function of RBAC model.

The constraints are applied on the above assignment functions depending on the access control policies of the system. Constraints on *Assigned_Roles* are similar to the constraints on user-role assignment in RBAC model. It specifies which roles are 'permitted' to be assigned to a certain trust_level. Constraints on *Assigned_Permissions* determines the assignment of permissions to a specific role. RBAC model suggests different constraints like *mutually exclusive role, prerequisite roles, cardinality constraints, static separation of duty, dynamic separation of duty* etc. But we prefer not to specify any particular constraint on these functions. Rather we leave it as general to give finer control in defining access control policies depending on the requirements of a system.

We also introduce a concept of **role dominance** among roles in our model. *Role dominance* is similar to the concept of *role hierarchies* in RBAC model. A role dominance relation, denoted by RD, defines a dominance relation between two roles. The dominance is described in terms of permissions. We define role dominance as,

Definition 31 **[Role Dominance]** *Role dominance* $RD \subseteq ROLES \times ROLES$ *is a partial order on ROLES where the partial order is called a* Dominance *relation, denoted by* $\preceq$. *For any* $(r_1, r_2) \in RD$*, we say* r_2 *'dominates'* r_1 *only if all permissions assigned to* r_1 *are also permissions of* r_2*. Formally,* $(r_1, r_2) \in RD \Rightarrow r_1 \preceq r_2$ *and* $r_1 \preceq r_2 \Rightarrow Assigned\_Permissions(r_1) \subseteq Assigned\_Permission(r_2)$.

The above definition implies that any user u having a role r_2 can have all the privileges of a user with role r_1.

The relation RD induces a similar relation called **trust_level dominance** among trust_levels in our model. Whenever there is a role dominance between two roles, there is a trust_level dominance between the corresponding trust_levels. Trust_level dominance, denoted by TLD is defined as follows:

Definition 32 [Trust_level Dominance] *Trust_level dominance,* $TLD \subseteq TRUST\_LEVELS \times TRUST\_LEVELS$ *is a partial order relation on* $TRUST\_LEVELS$ *and is denoted by* $\leq'$. *For any* $(L_1, L_2) \in TLD$, *we say* L_2 *'dominates'* L_1 *only if* $L_1 \subseteq L_2$. *If* L_2 *is a singleton set* $\{l_2\}$, *then dominance is defined as,* $sup\{L_1\} \leq l_2$ *that is,* l_2 *is greater than or equal to the maximum element of* L_1. *If both* $L_1 = \{l_1\}$ *and* $L_2 = \{l_2\}$ *are singletons then* $L_1 \leq' L_2 \Rightarrow l_1 \leq l_2$ *(the* $\leq$ *is the usual 'less equal to' relation of number theory).*

The relation TLD is induced by RD. That is, for any $(r_1, r_2) \in RD, \exists(L_1, L_2) \in TLD$ such that $r_1 \in Assigned\_Roles(L_1)$ and $r_2 \in Assigned\_Roles(L_2)$. That is, the trust degree of a user with role r_2 is greater than that of a user with role r_1.

6.1.3 Access Control using TrustBAC

Basic purpose of an access control mechanism is to protect system resources by restricting the user's activities on them. A user's authorization to perform certain tasks on specific resources is specified by the access control policy of the system. When using TrustBAC for access control, a *user* invokes a *session_instance* of a particular type at an instant of time. During this session the user has a *trust_level* which allows her to use the *roles* associated with that trust_level. That is, a user can be a member of a role. Also a single role can be exercised by many users. For each of these roles, the user has a set of *permissions.* Therefore, the user is restricted to perform a set operations on a particular set of resources as specified by the set of permissions obtained as a member of those roles.

A first time user u registers with the system and logs in which instantiates a session_instance s of the user. Depending on the set of disclosed properties P, the system invokes the function sua with arguments u, s, and P to start a session s^P. The system initiates a trust relationship $(SYS \xrightarrow{P} u)_t^N$ with the user in that session. The underlying context of this trust relationship is identified by the session_type P. This relationship does not change, but gets updated for any other session of same type P invoked by the same user u. If the user invokes another session_instance of type P' at time t (where $P \neq P'$), then the system creates another trust relationship $(SYS \xrightarrow{P'} u)_t^N$. The value of the trust relationship $(SYS \xrightarrow{P} u)_t^N$ is evaluated for the session s^P. Let $\mathrm{v}(SYS \xrightarrow{P} u)_t^N = l,\ l \in [-1, 1]$. The system invokes the function *Assigned_Roles* to determine the roles that the user u can execute. Let $Assigned\_Roles(\{l\}) = \{r_1, r_2, \ldots, r_n\}$. u can choose to execute more than one of these n roles. With each r_i, u has a set of p_js where $\forall j, (p_j, r_i) \in PA$. Therefore, in a session s^P, the user u has the set of permissions given by, $\bigcup_i Assigned\_Permissions(r_i) = \bigcup_{1 \leq i \leq n} \{p_{ji} \mid (p_j, r_i) \in PA\}$. Hence, the user u is restricted to perform actions A on a set of objects O where $\exists$ a $p_{ji} \in \bigcup_{1 \leq i \leq n} \{p_{ji} \mid (p_j, r_i) \in PA\}$, such that, for any $(o, a) \in O \times A$, $(o, a) \in p_{ji}$. The user executes these actions on the allowed objects and each activity during that session s^P is stored as the session_history ${}_u h^P$ for that session. Whenever the trust_level is re-evaluated (within s^P or, at the start of next instance s' of a session of type P), the *events* in ${}_u h^P$ are evaluated. The evaluated trust_level, say l' overwrites l in ${}_u h^P$. The subsequent events also overwrite the previous event log.

We assume that for a registered user u in a session s^P, the trust relationship $(SYS \xrightarrow{P} u)$ is managed by a diligent system-administrator who is outside the scope of this access control framework. We denote this system-admin by the symbol SYS. We also assume that the system has two predefined policies – an access control policy $\mathcal{A}_{SYS}$ and a trust evaluation policy $\mathcal{T}_{SYS}$ which are not independent. $\mathcal{A}_{SYS}$ defines the functions *Assigned_Roles* and *Assigned_Permissions* together with the 'constraints' on them.

In TrustBAC we use our vector trust model to evaluate trust levels of a user in a session. The parameters, for this purpose, are evaluated as

6.1.3.1 Computing Properties

The user u initiates the session s^P by disclosing a set of properties P, which includes information (e.g., name, address, affiliation, etc.) as well as some credentials. Credentials are in the form of typical digital certificates. The system assign a value within $[-1, 1]$ as weights to the information and the credentials. The assignment is done as specified by $\mathcal{T}_{SYS}$ and ${}_{SYS}P_u^P$ is computed according to the Equation 3.5. The next instance of a session of type P, the values assign to members of P may change due to change in values in P. For example, the user disclose the same type of certificate, but with a different certifying authority.

6.1.3.2 Computing Interactions

Interactions is computed from the *events* occurred during some intervals. In TrustBAC, we do not dictate about the length of an interval. It depends on implementation – the system may choose to identify a whole session as an interval. Independent of the length of an interval, any *action* performed by the user is identified as an 'event'. This record is kept in session_history ${}_{u}h^{P}$ till the next instant of trust evaluation. Formally, let l be the trust_level of u in a session s^P. Let $Assigned\_Roles(l) = \{r_1, r_2, \ldots, r_n\}$ of which u activate r_1, r_2, r_3. These are the *active roles* of u in session s^P. The events are the set of actions $\mathbb{A}$ where for any $a \in \mathbb{A}$, $\exists\, p \in \bigcup_{1 \leq i \leq 3} Assigned\_Permissions(r_i)$. The weight to the result of a particular action is assigned according to $\mathcal{T}_{SYS}$ and the 'interactions' ${}_{SYS}I_u^P$ is computed as specified in Section 3.3.1.1.

6.1.3.3 Computing Recommendation

The system may take *role-specific* and *role-independent* input from other users about u in a session. These information constitute u's recommendation and the component ${}_{\Psi}REC_u^P$ is calculated using the Equation 3.7. Ψ is the set of other users who provide recommendation for u to SYS. However, we choose not to specify how these information are collected.

6.1.3.4 Regarding Reputation

For this application we choose to disregard the parameter *reputation.* Rationale is, all the third party sources providing information about the user u are users of the system SYS, that is they are attributable sources. Hence the information obtained from these sources are considered to be *recommendation* rather than *reputation.* We assign a value $\perp$ to the reputation component in this application.

6.1.4 Computing User's Trust

After computing the components, the system calculates the normalized trust by combining $(SYS \xrightarrow{P} u)_t$ and the normalization policy of $\mathcal{T}_{SYS}$. Then the previous trust_level is fetched from ${}_uh^P$ and final $(SYS \xrightarrow{P} u)_t^N$ is calculated. The corresponding value $\mathbf{v}(SYS \xrightarrow{P} u)_t^N$ is calculated as specified in the Section 3.3.4. This value denotes the current trust_level of u in a session of type P and gets stored in corresponding session_history ${}_uh^P$.

Note, TrustBAC does not verify the credentials disclosed by the user. The TrustBAC module includes a trust evaluation module which computes and stores all relevant trust information including session_history. It interacts with two policy specifier modules which stores $\mathcal{A}_{SYS}$ and $\mathcal{T}_{SYS}$. Authenticity and veracity of credentials are checked by a suitable module outside the framework. It passes the necessary information to trust evaluation module. A compliance checker and access controller module can be implemented to enforce the access privileges according to the trust decisions. It interacts with the access specifier and the trust evaluation module to enforce the proper access privileges. These modules are outside TrustBAC framework and a part of the application which work in conjunction with TrustBAC.

6.1.5 Architecture of Trust-based Access Control System

A high level system architecture of the trust-based access control system consists of the components as shown in Figure 6.2. The two main components are *authorization controller* and *trust engine*. The authorization controller interacts with the *content-server* and the trust engine.

Access specification module This module defines the roles, groups and classes of resource objects, and the permissions, that is types access privileges that are to be tied to each resource object or resource class. It also defines role dominance and class hierarchy, if any, of classes of resources. This module is also responsible for specifying any special constraint (other than trust level) or an exception that has to be satisfied to allow access to a resource object or resource class.

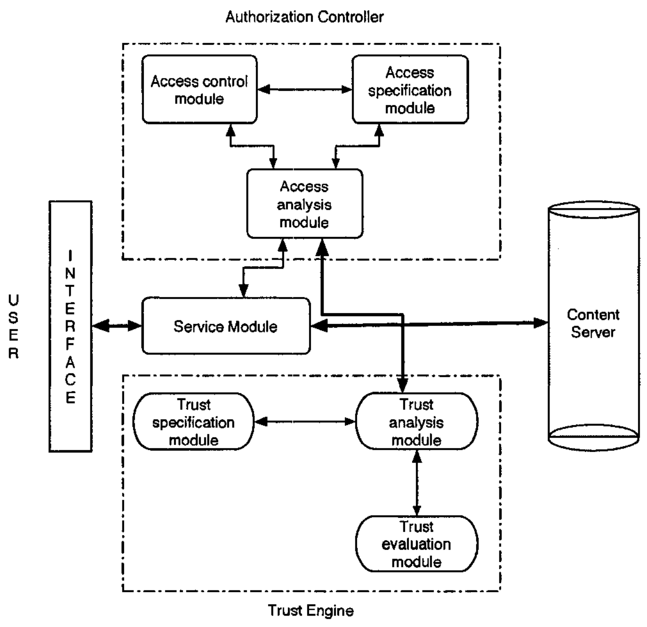


Figure 6.2: Architecture of trust-based access control system

Access control module This module is responsible to define trust_levels. It also defines the function *Assigned_Roles* that is, what trust_levels are associated to which roles.

Access analysis module This module has a user database. It receives the user's information and user's request through a *Service module.* It passes user information to trust engine and receives trust related result from it. Consulting with the access specification module and access policy module, it takes the decision about the specific request of the user and pass it to the service module. It also verifies user information and checks for special constraints and exceptions.

Service module The service module is an independent module outside the authorization controller as well as trust engine. Its job is to interact with the user through an interface. It collects user input and sends it to access analysis module of authorization controller. According to the decision it receives from access analysis module about the request it interacts with the content-server and provides the requested service to the user.

Trust specification module It is responsible for defining and managing trust relationships. It creates database entries corresponding to a specific user when a new trust relationship is established. It codifies general trust evaluation policies (for example policy for trust dynamics). The specification module conveys this information to the analysis module and the evaluation module as and when needed.

Trust analysis module The analysis module processes trust queries from access analysis module of authorization controller. It obtains trust vectors from the evaluation module.

Trust Evaluation module This module retrieves information about interactions, properties, and recommendation from the database and also other pertinent information from the trust specification module to compute the trust vector. It also stores back resulting values in the database kept in trust specification module.

We have proposed a trust-based access control scheme (TrustBAC) by extending the role-based access control mechanism (RBAC) using the vector trust model. Instead of users being assigned to roles as in traditional RBAC, users are assigned to trust levels. Trust levels are assigned to roles according to organizational policies. Roles are assigned to permissions as in the traditional RBAC model. The TrustBAC model being an extension of the RBAC model has all the latter's advantages. In addition, it borrows from credential based access control models in the sense that TrustBAC relies on evaluation of user's trustworthiness for access control. The model does not preclude use of credentials for such evaluation of trustworthiness. Thus the model is well suited for open systems like the Internet.

6.2 Trust-based Routing in Pervasive Computing

This section presents another application of vector trust model in finding the most 'reliable' path to send data from a source to a destination in pervasive computing environment.

The event-condition-action (ECA) paradigm holds enormous potential in pervasive computing environments. However, the problem of reliable delivery of event data, generated by low capability sensor devices, to more capable processing points and vice versa, needs to be addressed for the success of the ECA paradigm in this environment. The problem becomes interesting because strong cryptographic techniques for achieving integrity impose unacceptable overhead in many pervasive computing environments. We address this problem by sending the data over the path from the sensor node to the processing point that provides the best opportunity of reliable delivery among competing paths. This allows using much weaker cryptographic techniques for achieving security. The problem is modeled as a problem of determining the most reliable path – similar to routing problems in networks. In [CPR07], we propose a trust-based metric for measuring reliability of paths. The higher the trust value of a path the more reliable it is considered. We adopt the vector trust model for estimating the trust levels of paths and propose a new algorithm for identifying the desired path.

6.2.1 Motivation

Pervasive computing technology has the potential to impact numerous applications that benefit society. Examples of such applications are emergency response, automated monitoring of health data for assisted living, environmental disaster mitigation and supply chain management. Pervasive computing uses numerous, casually accessible, often invisible, computing and sensor devices. These devices are frequently mobile and/or embedded in an environment that is mobile. Most of the time they are inter-connected with each other, with wireless or wired technology. Being embedded in the environment and interconnected allow pervasive computing devices to exploit knowledge about the operating environment in a net-centric manner. This enables pervasive computing applications to provide a rich new set of services and functionalities that are not otherwise possible through conventional means. Pervasive computing applications frequently rely on event-triggered obligation policies to operate in a dynamic environment. An obligation policy is associated with events, conditions, subjects, objects and actions. When the event of interest occurs and the associated conditions evaluate to true, the subjects perform the specified actions on the objects. Events are typically identified and captured by embedded sensing devices and actions are actuated by similar embedded devices. Processing of captured event data for evaluation of

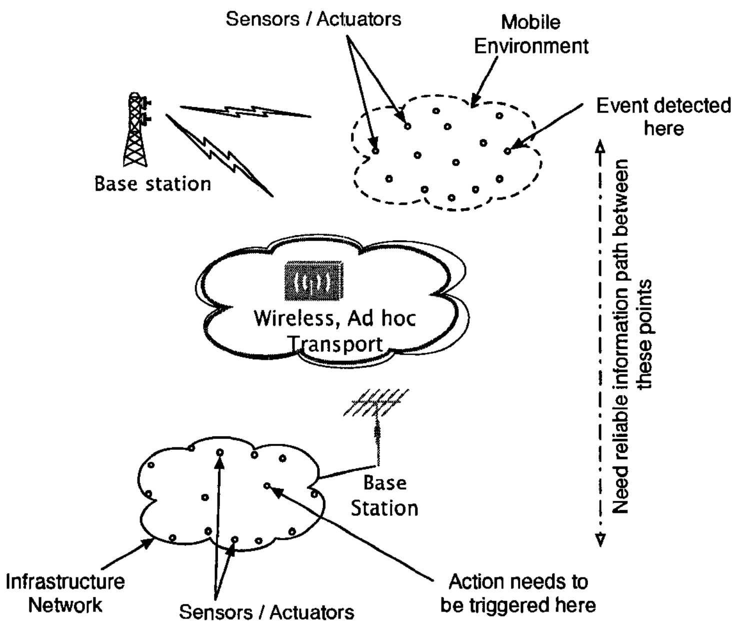


Figure 6.3: Pervasive computing environment involving remote event detection and action triggering

conditions are, on the other hand, mostly performed at remote processing nodes or base stations. This is because the sensing and actuating devices embedded in the environment are frequently of very low performance capabilities including low computing, low storage and low power. Thus a major challenge in a pervasive computing environment is to provide a path for propagating sensor data to processing nodes and action data to actuating nodes. Reliability of the paths is important. The data should be delivered with the minimum possible error and in as timely a manner as possible.

The reliable transmission path requirement imposes significant challenges in pervasive computing environments. A pervasive computing application can seldom assume a reliable network infrastructure for communication. In a conventional setting, a node that generates a message forwards it to a neighboring reliable node. The receiving node in turn forwards the message to another fixed node that is known a priori. This procedure is followed till the message reaches the destination. Every node in this process knows at least one other reliable node in the path towards the destination to which the message can be handed over. Frequently a node will know about more than one other node and thus have a choice of a better node. The nodes are static, that is they do not change their location and consequently the links between the nodes are stable. This and the proper use of strong cryptographic techniques, easily facilitate reliable delivery of messages in conventional settings. In a pervasive computing environments, on the other hand, mobility of nodes (sensing, processing or actuating) is frequently considered an asset. Figure 6.3 depicts the scope of the problem. Nodes are not locationally stable; instead they continuously change their coordinates. Thus, a node that needs a message delivered cannot rely on another fixed node to forward the message but has to make use of one or more nodes that happen to be within reachable distance at that particular moment. In addition, since a majority of these nodes are low capability devices (in the sense of low computational capabilities, low storage and low power provisions), use of strong cryptographic techniques needs to be ruled out. Moreover, in hostile environments these nodes get easily compromised. Under such circumstances it will enormously benefit a pervasive computing application if the path that provides most opportunity of reliable delivery of messages is presented to it. Determining an appropriate path within a network is the problem of routing and we revisit this problem in the context of pervasive computing environments.

6.2.1.1 Background

The problem of routing in mobile ad hoc networks have been addressed before [BSSW02, CWLG97, CE95, GB81, GT95, HPJ02, PH03a, PB94, Toh96, ZH99, ZMHT05]. Among these [BSSW02, HPJ02, PH03a, ZH99] study cryptographic techniques for securing the routing protocol. Some use public key cryptography to encrypt the end-to-end transmission of routing messages. Others use digital signature techniques to authenticate routing messages at the peer-to-peer level connection. However, these cryptographic techniques incur high computation and storage overhead which limit their use in sensor devices. Use of secret key techniques instead of public keys alleviate this problem to some extent although at the expense of added complexity. Moreover, key distribution and management is a big problem in secret key based systems. It is difficult to establish a key distribution or certification authority in mobile ad hoc environments. Ensuring the availability of a key distribution center or a certifying authority is almost impossible given the unstable nature of the network.

The Hermes protocol developed by Zouridaki et al. [ZMHT05] proposes using trustworthiness of its neighbors for routing. The trust values are computed under the assumption that they follow the beta probability distribution. The parameters of the beta distribution come from the empirical observation of the forwarding behavior. Thus, nodes that maintain a good and steady forwarding history have more trust and confidence on them. The route is established for the most trusted path. However, the major problem of this work is its complete reliance on forwarding history for measuring trust. A malicious node can easily fake this history thus presenting itself as a trusted node. Other similar works include [AHNRR02, DRWT97, YNK01]. Among these [DRWT97] proposes a signal stability-based adaptive routing (SSR) where the routes are selected based on signal strength. This work looks promising; however it does not discuss how to measure this signal strength quantitatively. In [AHNRR02] the authors propose an on demand secure routing protocol where the metric is based on past history. Yi et al. [YNK01] present a security-aware ad hoc routing protocol (SAR) in which a route is selected on the basis of degree of 'security guarantee' that the route provides. If two routes have same guarantee then the shorter path is chosen. The security metric can be specified by standard security properties like timeliness, ordering, authenticity etc. However, the work does not discuss how we can measure these properties quantitatively.

6.2.1.2 Our Proposition

We propose a trust-based routing protocol for pervasive computing environments. Our protocol determines the most reliable path under currently determinable properties of the system to forward a packet from source to destination. A node in the pervasive computing environment is any entity that is able to forward a packet. It can be a sensor node, a mobile device like a PDA or a cellular phone, a powerful computing device or even an actuator like a switch. Reliability of a node is measured in terms of a *trust* value for the node. Each node determines its neighbor's trust based on physical properties of the neighbor that can be directly observed *properties*, the neighboring node's behavior history (i.e., results of past interactions), recommendation and reputation about the neighbor from other neighbors. The resulting trust value is used to generate the 'cost of forwarding', or simply *cost.* The cost metric is inversely related to the trust metric, that is, higher the trust on the node, lower is the cost. This cost is associated with links in the network. We modify the widely used distance vector routing protocol using these costs between the links to find the path with minimum average cost. The chosen path then becomes the most 'reliable' path.

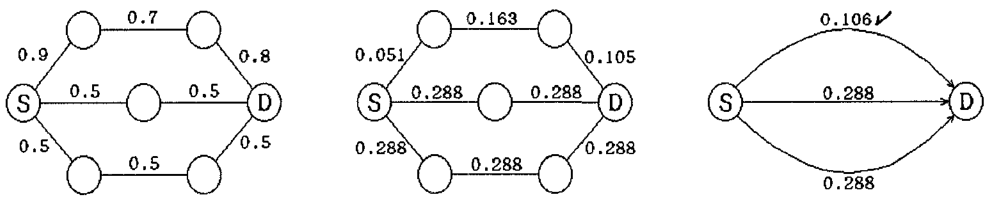
6.2.2 Overview of Trust-based Routing Protocol

We assume that the pervasive computing environment supporting the application has a very dynamic topology. Nodes join or depart the environment at random. Each node in the network maintains a table *RT* consisting of tuples of the form $\langle Dest, Win\#, NH, Cost_{avg}, \sum Cost^2, \#Hops\rangle$. In this table the node stores on a per-destination (*Dest*) basis, the identity of the next neighbor (*NH*), to which the message needs to be forwarded. Together with the next neighbor information, the node also stores the minimum average cost ($Cost_{avg}$) and the number of hops ($\#Hops$) to reach the particular destination. This information is generated periodically. Thus each tuple bears a time stamp in the form of a current time window ($Win\#$). The routing algorithm that we propose is used to update the next neighbor entry in the *RT* table for a particular destination.

A source node initiates the routing protocol if it has a packet to be sent to a destination for which it does not have any next hop (*NH*) entry. The source node can also execute the routing protocol when the path to the destination has expired (when the value under $Win\#$ is less than the current window number). The source sends out a route discovery request. We assume that each node that participates in the pervasive computing environment has a *trust relationship* with its neighbor (that is a node at 1 hop distance). A trust-aware node periodically sends a *beacon* message to its neighbors. The beacon message is something like an "I am alive" message and carries information necessary to prove the node's existence. Beacon messages are broadcast in nature. They carry rudimentary checksums to provide weak protection against integrity violations. Once in a while a node can also send out a beacon message on demand. In our protocol, a node may request a *recommendation* from a second node about the target node. In such cases the feedback is carried on a beacon message. Trust relationships are periodically refreshed locally. At some point after system initiation we assume that every node in the system will have a trust relationship with each of its neighboring nodes. We do not assume that trust relationships are symmetric or transitive.

We adapt our vector trust model for this purpose. We express the trust relationships between nodes as $N_r \longrightarrow N_e$ where N_r is the *truster* node and N_e is the *trustee* node. The underlying context is 'reliability in delivering a data towards destination' and is same for all node in the network. Therefore, we do not use the context any further for this application. We also choose to disregard the dependence of trust on time in this application., as a trust relation can be short-lived in pervasive computing environment. Consequently, it will not be much meaningful to capture the dynamics of trust over time. The trust value $\mathbf{v}(N_r \longrightarrow N_e)$, corresponding to $N_r \longrightarrow N_e$, is referred to as the trust value for node N_r on node N_e along the edge $(N_r,\ N_e)$. We convert this trust value to a cost on the link $(N_r,\ N_e)$. The higher the trust value the lesser is the cost to transfer messages on the link. The path having the least average cost from the source node to the destination node is considered the most reliable among the available paths and is chosen by the source node to forward the data.

Figure 6.4(a) describes pictorially the main idea of our protocol. We assume that if a node N_r has a distrust value (that is value less than 0) on another node N_e the cost on that link is infinite and that next hop is discarded. When a node receives a route discovery



(a) Trust relations in trust-aware pervasive computing environment

(b) Forwarding cost in trust-aware pervasive computing environment

(c) Path having least average cost is most reliable

Figure 6.4: Trust relation between nodes and the corresponding cost on the link

request from a source, it checks its routing table RT. If a route to the destination is present in RT which has not expired, it sends the '$\#Hops$' and cost related information to the source. The source then evaluates the cost of the link between the neighbor and itself and using the $\#Hops$ it computes the average cost of forwarding the packet to the destination. The source may get multiple such responses. It then chooses the next hop for which the average cost over the path is minimum. If the node that receives a route discovery request from the source, does not itself have the next hop information in its RT for that destination, it initiates a route discovery process as a source. This process can go on till the node which is 1 hop behind the destination initiates a route discovery request.

6.2.2.1 Cost Function

Each node tries to find the path to a given destination which has the minimum average forwarding cost. The *cost* of forwarding a packet from the node N_r to N_e is a function of the trust value $\mathbf{v}(N_r \longrightarrow N_e)$. These two are related as follows: higher the trust, lesser is the cost and the cost increases as the distrust increases (i.e., the trust decreases). Rationale is, the cost (in terms of integrity violation and other malicious activities) of forwarding a packet through a more trustworthy node is less than that through a less trustworthy node. The cost is minimum (not zero though) when N_r has absolute trust ($\mathbf{v}(N_r \longrightarrow N_e) = 1$) on N_e. This minimum cost (Min_{cost}) is a small positive cost incurred due to forwarding overhead. It is uniform over the whole pervasive computing environment and set at the bootstrapping of the system. We assume that the decay in cost with increased trustworthiness is logarithmic with the following conditions: at $\mathbf{v}(N_r \longrightarrow N_e) = 1$, cost $= Min_{cost}$ and at $\mathbf{v}(N_r \longrightarrow N_e) = -1$, cost $= \infty$. The function is defined as,

$$cost(N_r,\ N_e) = Min_{cost} - \ln(\frac{1 + \mathbf{v}(N_r \longrightarrow N_e)}{2}) \qquad (6.1)$$

The maximum allowable cost for N_r is incurred when $\mathbf{v}(N_r \longrightarrow N_e) = 0$. This corresponds to the situation when N_r is *neutral* about trustworthiness of N_e. This cost, denoted by $MaxAllowed_{cost(N_r,\ N_e)}$, is $Min_{cost} - \ln(\frac{1}{2})$, that is, $MaxAllowed_{cost(N_r,\ N_e)} = Min_{cost} + 0.69$.

Computing this cost value has some overhead but is only linear in the number of nodes in the pervasive computing environment. The cost value is stored in the device for a predetermined time or until a new beacon message has arrived. The absence of a beacon message from a particular node in a particular window of time represents a broken link during that time period. It can happen for various reasons including that the node is compromised. The node in such a case may either discard the broken link from the list of current neighbors or mark it as unused. Routing information is advertised by broadcasting the route setup packets periodically or on demand depending on the protocol used. These packets indicate which mobile nodes are accessible from which others and the average cost associated with the path towards a destination. When a node receives a data packet, it chooses the path which has the lowest average forwarding cost and forwards the packet to the neighbor on this path to be further forwarded towards the destination. During this process, a node also evaluates the packet forwarding performance of the neighbor node. By measuring this the node essentially evaluates an interaction score for the neighbor. The details of this process and other routing processes is explained in Section 6.2.4.

6.2.3 Trust Metric

As mentioned earlier, we adapt our vector trust model to evaluate trustworthiness of nodes. For this application, we evaluate the parameters as follows:

6.2.3.1 Computing Properties

In our approach trust is used as a reliability metric of a neighbor node for proper handling and forwarding the packet to the destination. A node N_i is a neighbor of node N_j if N_i is within the range of a beacon message from N_j. A node becomes more reliable when it has relatively more resources (in terms of signal strength, signal stability, less propensity to corrupt data etc.). Higher values of these attributes show that it is more capable of handling and forwarding a packet in a reliable manner. This motivates us to measure the node properties quantitatively and include that measure as a factor to evaluate trustworthiness of a node. We focus on two properties of a node – *signal strength*, and *stability factor*. A node maintains a property table, $PT = \langle Node\_id, SS_{avg}, SF \rangle$ for each neighbor node where the properties of the neighbor is kept along with the corresponding id. The table is updated after each time-window *win*.

6.2.3.1.1 Measuring Signal Strength In each time-window *win*, a node periodically sends a link layer beacon message to its neighbors. When the neighbor node receives such a beacon message, the extended device driver interface of the receiving node measures the signal strength at which the beacon was received. In our approach we use the *receive signal strength indicator* (RSSI) unit to measure the signal strength. RSSI is the IEEE 802.11 standard for measuring radio frequency (RF) energy sent by the circuitry on a wireless network interface card (W-NIC). It is a numeric integer value with an allowable range of 0 to 255. However, for the sake of our model we give a transformation to this recorded signal strength value by dividing it by 255. This scales the received signal strength value within the range $[0, 1]$. We require this transformation as the final value of the component 'properties' lies within $[-1, 1]$. At the end of each time-window, we take the average of these values. This transformed average signal strength value is then stored under the column SS_{avg} in the table PT corresponding to the neighbor. All these signal strength values within the time-window *win* is kept in a separate temporary property table, $PT_{tmp}^{Node\_id} = \langle SS, SB \rangle$ where SB is the *stability bit* explained next.

6.2.3.1.2 Measuring Stability Factor The stability factor indicates the stability of a node. Higher the stability more reliable is the node to forward a packet. We derive the stability factor using signal strength. The reason is as follows: if a node is locationally unstable, it will have a varying signal strength. Alternatively, if the average signal strength of a node is fairly constant over few time-windows, the link with the node can be considered as stable. Therefore, after storing the strength of received signal, say $SS_{current}$, in $PT_{tmp}^{Node\_id}$, it is compared with the value present in SS_{avg} of PT. If $SS_{current} < SS_{avg}$ then the SB is set to 0, otherwise the default value 1 is kept. At the end of time-window, the stability factor SF is calculated as,

$$SF = \frac{\text{number of bits set to 1 under SB}}{\text{Total number of bits under SB}}$$

At the beginning of each time window the temporary property table $PT_{tmp}^{Node\_id}$ is set to its default values. The default value for SS is 0 and for SB is 1.

6.2.3.1.3 Measuring Properties

The *properties* component of the node N_e is then computed as

$$_{N_r}P_{N_e} = \alpha * SS_{avg} + (1 - \alpha) * SF \tag{6.2}$$

where $\alpha \in (0, 1)$ is a fraction used as the relative importance weight to the signal strength property.

6.2.3.2 Computing Recommendation

Each trust-aware node agrees to provide a 'recommendation' about its neighbors upon receiving a *recommendation request* from a source node. Let N_r request N_k for a recommendation about a node N_e. The source node N_r sends this recommendation request by sending a special message REC_REQ containing the node_id of the target node (in this case N_e). If N_k has a trust relationship with N_e, then N_k replies by sending a message REC_RESPONSE containing the pair $\langle node\_id, V\rangle$ where $V = \mathbf{v}(N_k \longrightarrow N_e)$. The node N_r then evaluates the recommendation parameter using the Equation 3.7.

The truster N_r maintains a list, called *recommendation list*, $RL = (node\_id, list)$ for each trustee where structure of each item in the list is (node_id, recommendation_value).

6.2.3.3 Computing Reputation

In our approach, we assume that each node also agrees to forward, to the source node (truster), the rating of the target node by a third node. For example, a node N_j may have obtained a recommendation about N_e by N_k. N_j agrees to pass this information to the source N_r together with its own recommendation about N_e. All such ratings can be bundled together and piggybacked with the message REC_REQ. N_r considers this rating of N_e by N_k, received from N_j, as a reputation information about N_e, if N_k is not a neighbor of N_r. This information is considered to be a reputation rather than recommendation as N_r does not have any trust relationship with N_k (since, N_k is not a neighbor of N_r). N_r then uses the Equation 3.6 to evaluate the the reputation parameter.

6.2.3.4 Computing Interactions

Interaction is modeled as cumulative effect of events encountered by a truster node N_r regarding N_e. We classify interaction in two categories – *packet forwarding interaction* – when the truster considers the behavior of the trustee as a packet forwarder, and *rating interaction* – when the truster considers the behavior of the trustee as a recommender. Every event in each of these categories has binary outcome; either the truster N_r has trust-positive event or a trust-negative event depending whether the event contributes toward a trust-positive interaction or a trust-negative interaction.

6.2.3.4.1 Evaluating Packet Forwarding Interaction To evaluate packet forwarding interaction, the truster N_r checks the outcome of each packet forwarded to N_e within the specific time window *win*. Each packet forwarded correctly towards the destination is considered as a trust-positive event. Each dropped packet gives rise to a trust-negative event. The node N_r measures the number of forwarded packet by N_e as follows: N_r forwards packets to N_e and with every such packet N_r sends an ECHO message with a *time-to-live* (TTL) = 2. Each reply received by N_r denotes correct forwarding of the packet by N_e to the next member N_k in the path. N_r keeps this information in a table $IT = \langle Node\_id, PFC_p, PFC_n, RC_p, RC_n \rangle$ where PFC_p denotes the counter for trust-positive packet forwarding interaction within the window and PFC_n counts the trust-negative packet forwarding interactions. RC_p and RC_n are rating counters used for counting the results of rating interactions. All these fields has default value 0. Whenever a packet is dropped the counter PFC_n is increased by 1 and for each received reply the counter PFC_p is increased. Formally, *packet forwarding interaction*, denoted by I_{pf} of N_r about N_e within the window *win* is defined as the ratio $\frac{PFC_p - PFC_n}{PFC_p + PFC_n}$. This is a simplified form of the

Equation 3.1 as we choose to disregard the time factor. Consequently, we ignore relative importance of past events and current events.

6.2.3.4.2 Evaluating Rating Interaction The *rating interaction* is evaluated in a similar manner. We assume that each node agrees to provide a 'trust recommendation' about its neighbors upon receiving a *recommendation request* from a source node. We also assume that for each neighbor, the truster keeps a list of node_ids of the nodes who have provided recommendation for that neighbor. Whenever the truster has a 'packet forwarding interaction' with the neighbor node and the result of that interaction matches with the recommendation, that is the truster has positive (negative) experience and the recommendation is also positive (negative), it increases the RC_p in IT of all such recommenders by 1. If there is a mismatch between the outcome and the recommendation, the truster increases the RC_n counter by 1 for those recommenders. For example, let the trustee N_e has provided "positive" recommendations for the nodes N_i, N_j, N_k to the truster N_r. Therefore, in the recommender list RL, N_e appears in the list against each of these nodes. Let N_r have trust-positive packet forwarding interaction with N_i, N_j and trust-negative packet forwarding interaction N_k. Then in the interaction table IT, for the node N_e, the counter RC_p is increased twice and RC_n once. At the end of time window *win* the *rating interaction*, denoted by I_r of N_r about N_e is defined as the ratio $\frac{RC_p - RC_n}{RC_p + RC_n}$.

6.2.3.4.3 Evaluating Interaction The *interaction* component of the node N_e is evaluated as

$$_{N_r}I_{N_e} = \beta * I_{pf} + (1 - \beta) * I_r \tag{6.3}$$

where $\beta \in (0, 1)$ is a fraction used as the relative importance weight to the packet forwarding interaction. While evaluating rating interactions, N_r penalizes N_e whenever there is a mismatch between N_e's rating and N_r's experience about a third node N_k. This is done to reduce the effect of 'badmouthing' or 'false campaigning' of N_k by N_e. However, there is a chance of wrongly penalizing N_e when the mismatch occurs due to no fault of N_e. To reduce the effect of this β should be chosen high so that the weight of I_r that is, $1-\beta$ is low.

6.2.3.5 Computation of Final Trust Value

After computing values of the parameters, we evaluate $\mathbf{v}(N_r \longrightarrow N_e)$, that is the trust value for the trustee N_e by the truster N_r. For normalization, we choose to keep the component-weight vector uniform over the whole environment that is, the vector is same for all nodes. Therefore, W is chosen as (1/4, 1/4, 1/4, 1/4). Formally, the trust value is computed as

$$\mathbf{v}(N_r \longrightarrow N_e) = \frac{{}_{N_r}I_{N_e} +{}_{N_r} P_{N_e} +{}_{N_r} REP_{N_e} +{}_{N_r} REC_{N_e}}{4} \tag{6.4}$$

These information are kept in a trust table, $TT = \langle$node_id, properties, recommendation, interaction, trust_value, cost$\rangle$. After each window *win* this table is updated with new values which are kept and used in the next time window. All other tables are set to their corresponding default values. Next section describes the modified distance vector routing algorithm which finds the path with minimum average cost for forwarding a packet to the destination.

6.2.4 Data Path Discovery

To select the most trustworthy path, each node evaluates and dynamically updates the trust components between itself and current neighbors. It then calculates the trust value of the neighbors by the process described in Section 6.2.3. These values are used to calculate the forwarding cost between two neighbors, using the Equation 6.1. The path with the minimum average forwarding cost is preferred and the adjacent node on this path is trusted to forward the packets toward the destination.

Our algorithm is based on a "rumor" about paths from neighbors. This is incomplete information. We thus choose to use the average and standard deviation of the running sum of cost in our route discovery protocol. This formula does not require the complete path information yet can correctly evaluate the path's reliability like the one with the complete path information. The average and standard deviation of running sum are computed using the following equations:

$$AVG = (x + \sum_{i=1}^{n} x_i)/(n+1) \tag{6.5}$$

$$SD = \sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2/n(n-1)} \tag{6.6}$$

where x_i's are values taken by a random variable X.

When a node receives a route information message from a neighbor node N_k, it updates the forwarding cost on the path towards node N_j (where node N_k is chosen as the next hop) by adding the current cost between itself and N_k and calculate the updated average cost using Equation 6.5. It then re-evaluates the path to choose the optimal route to the destination. This process compares among all possible candidate routes and chooses the path that has the minimal average cost. If more than one candidate paths have same minimum average cost, the routing algorithm selects the path that has the least standard deviation as an optimal path. The standard deviation is calculated using the Equation 6.6. Algorithm 5 gives the protocol used in generating the routing table. It consists of two phases: table initialization and iteration. The table initialization phase establishes paths to all immediate neighbors known to the source S. For each neighbor N_k, node S keeps track of hop count, average cost (calculated from Equation 6.4), running sum of cost ($DT(S,\ N_k)$), and running sum of square of cost ($DT^2(S,\ N_k)$). These cost parameters are used for calculating the average cost and standard deviation according to the Equation 6.5 and Equation 6.6. The iteration phase is only triggered upon receiving the routing packets or upon changing of an immediate link with its neighbor.

In the first case, if the destination node N_j in the received packet is not known by node S, it will add N_j to the routing table and compute the routing cost to N_j by adding its trust between itself and its neighbor node who has sent the routing information of N_j to S. The routing cost to the destination N_j is computed as:

$$D^S(N_j,\ N_k) = \frac{\mathbf{v}(S \longrightarrow N_k) + DT(N_k,\ N_j)}{hop\_cnt(N_k,\ N_j) + 1} \tag{6.7}$$

where Node N_k is the sender of the routing information and $DT(N_k, N_j)$ is the forwarding cost from N_k to N_j. If S already knows the destination node N_j, it recomputes the routing

Algorithm 5 Route Discovery in Pervasive Computing

```
**Description: Route Discovery procedure simplifies the modified Distance Vector algorithm.**
**Input:** destination $N_j$, reachable from node $N_k$
**Output:** : The routing table of a given source node S

**Initialization:**
Initialize cost to all nodes $N_j$ known to S to $\infty$
Calculate the trust between S and its immediate adjacent node $N_k$ in S's neighbor list
Add all immediate adjacent nodes to the routing table
for all node $N_k$ in the neighbor list do
  compute trust between S and $N_k$ (Equation 6.4)
  compute average cost $D^S(N_k,\ N_k)$ (Equation 6.5)
  compute running sum $DT(S,\ N_k)$
  compute running sum of squares $DT^2(S,\ N_k)$
end for
**Iteration:**
Wait until S detects change from its immediate link $N_k$ or receives a routing packet from
its neighbor /* This packet contains the information about the destination node $N_j$ */
if S detects change in its immediate link then
  Update the cost and propagate the change to all neighbors
else
  if $N_j$ is a destination that S has never seen before then
    Compute routing cost to $N_j$
    Compute running sum, running sum of squares, and hop count to $N_j$
    Add $N_j$ and its routing parameters into the routing table
  end if
  if $N_j$ is already in the routing table then
    Compute the routing cost to $N_j$
    Update the routing table if new cost is better than the current cost in the routing
    table
    Announce the new routing table to neighbors
  end if
end if
```

cost to N_j and compares this value with the existing value. If the new cost is less or more stable[1] than the current cost, the cost to the destination N_j is updated. If the trust value between node S and its immediate neighbor N_k has changed, S has to recompute the routing cost to all destinations N_j where N_k is the next hop. The above Equation 6.7 is used to recomputing the new routing cost. Then S compares the new cost to the current cost that S has in its routing table. If the new cost is less or more stable than the current cost, the cost to the destination N_j is updated.

We use the vector trust model to address the problem of reliable delivery of event data in pervasive computing environments to appropriate action points in order to support obligation policies. The problem is modeled as a routing problem. We present a trust-based approach to routing and used vector trust model to evaluate trustworthiness of nodes to proper handling and forwarding data towards the destination.

6.3 Allowing Finer Control over Privacy using Trust

This section presents the third example of application of vector trust model. Here the trust model is used to facilitate privacy related decisions in different privacy contexts on the Internet by evaluating trust in recipients of private data. Reasoning about trustworthiness of recipients of private data is essential as every time a user uses the Internet, a wealth of personal information is revealed, either voluntarily or involuntarily to some recipients. This often causes privacy breaches, specially if the information is misused. Ideally, a user would like to make a reasoned decision about who to release her information to and what to release. For this purpose, in [RC08], we propose using the level of trust that a user has on the recipient regarding not to misuse her private data. To measure this trust level, we adapt the vector trust model. We formalize a notion of privacy context and show how a

[1]This is used in the case when the new cost is equal to the the current cost or has a slight difference. The path that has less standard deviation is said to be more stable path.

privacy context ontology can be used to determine trust values for previously unencountered situations. For this purpose we adapt our context ontology presented in Chapter 4.

6.3.1 Background and Motivation

Researchers are getting increasingly concerned about protecting the user's privacy in an electronic world. Unfortunately, most of us would find it difficult to provide a concrete definition of privacy with enough information to be able to apply it to our real lives. As individuals, each of us have unique needs and views of what constitute personal and private data [ACR99]. The task is considerably more difficult when we have to define what privacy means to us as we use the Internet. Efforts to define and develop technologies that support the specification of consumer privacy requirements as well as help protect them, are evolving at a considerably slower pace. Efforts like the Platform for Privacy Preferences (P3P) Project of the World Wide Web consortium [$C^{+}04$] and the related Privacy Bird project [Corb], and works based on the *k-anonymity* and *ℓ-diversity* models, provide solutions to some facets of electronic consumer privacy. For example, the P3P project attempts to provide a framework for service providers to express their privacy policies to the user with the goal that a user can form a reasoned opinion about the state of her privacy at the service provider. The related work on Privacy Bird [Corb] provides a user-friendly mechanism by which a user can determine if a service provider's P3P policies match the user's privacy preferences. The understanding is that such compliance will enhance the user's trust in the service provider. However, P3P is only able to provide a technical mechanism by which service providers can describe their use of personal information. P3P does not provide mechanisms by which policies are enforced. Nor can policies be used to verify or prove that the services accurately reflect the original policies. The *k-anonymity* model [SS98, Swe02], *ℓ-diversity* model [MKGV07] and similar works like [Sam01], on statistical databases [DF02], and deductive databases [BKS95] address the problem of releasing personal information so that the subjects of the data cannot be identified uniquely. Proponents argue that these efforts enable the users to act on what they see and thereby help protect their private information. However, often privacy is breached by factors that the users cannot see or control – for example, misjudged trust and misuse of information. Thus, these models and technologies solve only parts of the problem of protecting user privacy.

The last observation indicates that trust plays an important role in the problem of preserving privacy. Ultimately, the user needs to trust the recipient with her private information. A number of researchers have previously explored the idea of modeling privacy using a trust centric approach [GM02, SDB03, NM02]. Goecks and Mynatt treat reputation and trust as separate independent entities and propose an approach to combine trust networks with reputation to provide privacy [GM02]. Shand et al. [SDB03] rely on recommendation to direct the sharing of private information. Nguyen and Mynatt [NM02] address the problem of trust in pervasive computing environment. Their goal is to make the user more aware of privacy issues. The goal of enhancing consumer confidence in privacy practices of service providers has been explored by privacy seal programs such as TrustE (http://www.truste.org). However, it relies heavily on policy statements similar to P3P statement.

We believe that trust based approach to preserving privacy is promising. The problem with this group of work is that the trust models used are not very expressive. Moreover, none of these works discuss how to evaluate trust for the purpose of privacy preservation. Therefore, for this application, we adapt and extend the vector trust model to help the user decide how much to trust the recipient of private data to preserve her privacy. We specify the user's different Internet activities like browsing a website, downloading content, purchasing, etc., as privacy contexts. The user is likely to have different privacy preferences for different contexts and may switch context anytime during an online session. Sometimes the user may not have enough information to calculate trust about a trustee in a new context. Or, the user may have no predefined preference rules in that context. We show how, in the above scenarios, the user can extrapolate a trust or a privacy preference rule-set using trust and preference rules for existing contexts. For this purpose, we adapt our context ontology to define an ontology of privacy contexts containing a similarity relationship between different contexts. This similarity relationship is represented by a context similarity graph. Using the degree of similarity between contexts, the user can extrapolate trust or can set up privacy preferences in the context for which she does not have any a priori information.

6.3.2 Preserving Privacy Using The Trust Model

We look into the privacy preservation scheme from a user's perspective. That is, we investigate how a user can have a reasonable control over her privacy while interacting with a server. We first identify following activities that a user can perform:

1. *Downloading* – The user downloads some resources from the server. This requires the user to specify (in active or passive manner) the download destination.
2. *Purchasing* – The user acquires some product, service, or access to a resource via a purchase. This requires the user to exchange funds and reveal a destination for whatever she is purchasing. In the case of acquiring access to some resource, that 'destination' is an identity to which that access is related.
3. *Sending/Receiving email* – The user exchanges electronic messages with other individuals to pass along digital information.
4. *Negotiating* – A series of proposal-response messages are passed between the user and the server, until either both parties reach an agreement with each others proposals, or one or both parties terminate the activity without an agreement. A certain level of trust is typically assumed in negotiation, and the user may have to reveal various characteristics about her to engender that trust and complete the negotiation.
5. *Filling out web forms* – A user fills out a form presented by a website to willingly share information.

During any of these activities there are many different ways that the user's privacy can be violated. We categorize the violations as follows

1. *Confidentiality breach* – when private and personal information of the user is intercepted and collected by an entity to whom the user is not intended to disclose that piece of information.

2. *Integrity breach* – when private and personal information of a user is modified without the knowledge or consent of that user. This can occur even if the modification is done by a legitimate receiver, but who is not authorized to do so.
3. *Information exploitation* – when private and personal information about the user, collected with her consent, is misused or allowed to be exploited. This would include, personal data of the user is made available for sale, use of the data by the receiver for profiling when the user has not so consented, use of the data that was not agreed to by the user prior its collection, and allowing unauthorized access to the data by other entities.
4. *Personal space violation* – when an entity other than the user places data of any kind on the computing system of that user without the knowledge or expressed consent of the user.
5. *Pretexting/Identity theft* – when private or personal data of the user is used by someone other than the user without her consent to do so to gain access to resources, products, or services intended for the user only.
6. *Anonymity violation* – when the identity of the user is disclosed despite the user's effort to remain anonymous.
7. *Linkability* – when personal or private data about the user, collected under the condition of anonymity of that user, is maintained/used/distributed in such a manner as to link that data to the identity of that user, or contribute to the linking of the identity of that user to that data.

Some of the above listed violations can lead to other violations. For example, a breach in confidentiality can lead to integrity violation, information exploitation, or identity theft.

Before each transaction, a user evaluates the trustworthiness of the server using the vector trust model. To evaluate this trust the user uses information about characteristics of the server (*properties*), her personal experience with the server (*interactions*), and information that she gathers from other members in the community (*reputation* and *recommendation*). In the following sections, we describe how these parameters are evaluated.

6.3.2.1 Computing Properties

To quantify the 'properties' component of the trust relationship, the user A (truster) needs to gather information about the attributes of the server B (trustee) and classify them. We give some examples of classes and subclasses of attributes of a trustee that a truster may define to evaluate *properties* component.

1. *Communication protocol* – Presence of a secure communication protocol like SSL helps preventing confidentiality breach, integrity breach, identity theft and thereby can prevent other indirect violations of privacy. In this class the truster may have the following subclasses:
   (a) Encryption algorithm – which encryption algorithm (e.g., AES or DES or RSA) is being used,
   (b) Key-type – what type of key (e.g., symmetric key or asymmetric key) is used,
   (c) key-size – what is the key size (e.g., 56-bit or 128-bit or 512-bit),
   (d) Message digest algorithm – what type of message digest algorithm is used (e.g., MD5 or SHA),
   (e) Authentication – what authentication mode is used (e.g., authentication of both ends, or only B's authentication or, it is totally anonymous),
   (f) Key exchange – which key exchange algorithm is used (e.g., RSA or Diffie-Hellman).

2. *Credential* – Presence of a certificate from a well-known certifying authority (e.g., Verisign) about policies, methods and tools applied and used by B in a particular transaction. The truster A can have following subclasses:
   (a) Certifying authority – who the certifying authority is (i.e., how well-known the certifying authority is),
   (b) Validation period – how long the certificate is valid. For example if it is an old certificate and is still valid for sufficiently long, then that would create a positive impression about B.

3. *Policy* – Presence of an explanation of policies adopted by the server for a transaction. In particular, the user looks for the following policies in the 'policy document'
    (a) *Data collection policy* – explaining how the server is going to collect private and personal data from the user,
    (b) *Data storage policy* – explaining how the server is going to store the private data of the user so that it remains secure from the privacy violating threats,
    (c) *Data handling policy* – explaining how the server is going to use the data,
    (d) *Data disclosure policy* – discussing whether the server is going to disclose the data to third parties. If so, to whom it will be disclosed.
    (e) *Data retention policy* – explaining how long the server is going to keep the private information about the user in the storage,
    (f) *Applicability & Validity* – applicability shows which entities are going to follow this policies (or, a part of the policies). Validity explains for how long the server (or other entities) is going to stick to this policy. The lifetime of a policy tells the user how long she can rely on the claims made in the policy, or whether there is any exception in these policies,
    (g) *Cookie policy* – a cookie policy must cover any data that is stored in that cookie or linked via that cookie. It must also reference all purposes associated with data stored in that cookie or enabled by that cookie. In addition, any data/purpose stored or linked via a cookie must also be put in the cookie policy. It must clearly specify the path of the cookie (this would give the idea about the parties that are going to get the data),
    (h) *Dispute handling policy* – explaining how the server is going to resolve dispute issues, or if the user lodges a complain about her privacy being violated, what compensation the server is offering.

Once some or all of these information are available, A evaluates 'properties', according to her own policies, as described in Chapter 3.

6.3.2.2 Computing Interactions

Most of the information that goes toward forming the properties of the trustee B in a particular privacy context by itself does not necessarily enhance/diminish the truster's trust on B. This is because majority of the above criteria are examples of self-assertions. There is no guarantee that B conforms to these self-assertions. B's behavior as an entity (it includes behavior as a service provider, recommender or just as a community member) in a transaction manifests in the form of *events*. If there are events that conform to the properties that A has gathered then these events will be termed positive. If the events are contrary to the properties then they are negative. A false or misleading recommendation is also a negative event. Otherwise the events are neutral.

Categorizing an event to positive or negative depends on the truster A's policy, specific activities and violations. Interactions is computed by counting how many times (i.e., in how many events) B has deviated from or conformed to self-assertions or provided wrong information. During a specific period of time, number of deviations from the stated self-assertions give number of negative events in that period. The events where B adhered to the self-assertions or provided correct feedback generate positive events.

6.3.2.3 Computing Recommendation

Evaluation of recommendation involves measuring the feedback provided by other members in the community. Note, however, a group of malicious members can send false good/bad reviews about the server (trustee) to influence the trust decision of the user (truster). The server may or may not be a member of that malicious group. To diminish the effect of such collusion while computing the recommendation, that is, to select 'attributable sources' from the whole community, we propose the concept of 'trusted neighbors'. Note, we do not use the term 'neighbor' to mean the physical distance (in terms of length or hop) of a member from the user. We intend to measure how 'close' the member is with the user in terms of trust relationship. We now discuss how a truster builds this set.

6.3.2.3.1 Trusted Neighbors Let there be m members in the community M. To choose the trusted neighbor set, a truster A sets up a neighbor_trust threshold τ_A^{nbr}. Then A broadcasts a ('neighbor_invitation') message to each of the members with whom A has a trust relationship and the value of the trust relationship is $\geq \tau_A^{nbr}$. A considers all members as her trusted neighbor from whom she gets back acceptance message. Therefore 'trusted neighbors' can be defined as

Definition 33 [**Trusted Neighbors**] *The trusted neighbors of a truster A is the set $TNBR_A$ of all members j where the trust value of j as evaluated by A is greater than or equal to the neighbor_trust threshold set by A and A receives an acceptance of neighbor_invitation from j. Formally, we can write, $TNBR_A = \{j \in M| \ v(A \xrightarrow{c} j)^N \geq \tau_A^{nbr} \wedge A$ receives acceptance of neighbor_invitation$\}$.*

The next algorithm formally describes the process of creating trusted neighbor set.

Algorithm 6 Get the trusted neighbors

```
Input: (i) $M$ – the community of members, (ii) $A$ – the truster whose neighbor set is to
be determined, (iii) $\tau_A^{nbr} (> 0)$ – neighbor_trust threshold set by $A$;
Output: $TNBR_A$ – set of trusted neighbors of $A$;

Procedure FindTrustedNeighbors($M$, $A$, $\tau_A^{nbr}$)
$TNBR_A = \emptyset$;
for all $j \in M$ do
  if v$(A \xrightarrow{c} j)^N \geq \tau_A^{nbr}$ then
    Send 'neighbor_invitation' message to $j$;
    if $A$ receives an acceptance of neighbor_invitation from $j$ then
      $TNBR_A = TNBR_A \cup \{j\}$;
    end if
  end if
end for
return $TNBR_A$;
```

6.3.2.4 Computing Reputation

We assume that the user requests only the trusted neighbors for recommendation or considers their feedback as recommendation. Information obtained from other members are used to compute reputation. The reason is as follows: the user will have a low trust or no trust for the members other than trusted neighbors. Therefore, these sources are almost non-attributable to the user. The user can gather information about the server's *reputation* from the following:

1. general description of the server's activities and performance – this can be available from the other members in the community,
2. report of other members about the server – this report can contain evaluation of the server and comments by those members. The report can have two categories: (a) general – general remark about the server by the other member, (b) specific – action specific remark about the server. For example, how the server has performed to handle private data, how it has collected and stored sensitive data etc.

After collecting these data, the user computes the reputation parameter as discussed in Section 3.3.1.3.

After computing the trust, the truster checks the privacy policy of the trustee. If it conforms with her privacy preferences in that context, then she controls the disclosure of her information based on the evaluated level of trust.

6.3.3 Privacy Context

As mentioned earlier, a trust relationship between A and B is never absolute. In privacy platform, a user's trust on another user (service provider or recommender) will depend how the other user is capable of keeping A's privacy in a specific context. For example, A (truster) can trust the entity B (trustee) to protect her private information collected during a registration procedure. However, that does not necessarily mean that A also trusts B to protect the private information collected while A is making a purchase from B. This leads us to associate a notion of *privacy context* with a trust relationship.

A user typically performs different activities during an online session. These activities can be categorized by their type. We denote each type as a 'context' of user activity. For example, a user may 'search' for some document, and when found, she may 'download' the corresponding file. The above involves two different contexts of activities, 'searching' and 'downloading'. Some examples of context are *Browse, Download, Purchase, Register, Login* etc. We assume the universe of contexts is finite. We observe that context should be defined such that the model is interoperable. Different entities often use different words to describe the same context. Alternately, the same word can be used for describing different contexts. These are example of semantic conflicts in the use of terminology. To solve these problems we adapt some ideas from our context ontology presented in Chapter 4. The next section presents our privacy context ontology.

6.3.3.1 Privacy Context Ontology

Our ontology consists of a set of contexts together with relationships defined among them. First, we formally define the privacy context and later define the relationships between them.

Definition 34 [Privacy Context] *A privacy context $\mathcal{C}$ is represented by a set of semantically equivalent keywords, denoted by $keywords(\mathcal{C})$.*

Each keyword in $keywords(\mathcal{C})$ is used to describe the privacy context $\mathcal{C}$. The keywords in $keywords(\mathcal{C})$ are semantically equivalent because they express the same context. For each context $\mathcal{C}$ we assume that the set $keywords(\mathcal{C})$ is non-empty and finite. Also, for any two semantically distinct privacy contexts $\mathcal{C}_1$ and $\mathcal{C}_2$, we require $keywords(\mathcal{C}_1) \cap keywords(\mathcal{C}_2) = \emptyset$. That is, any keyword belongs to exactly one context. We give an example to illustrate the notion of privacy context. Consider the usual registration process in an Web-service. Some sites call it 'registration', some call it 'register', and some sites specify it as 'sign-up'. All these different terminologies describe the same process. Therefore we specify any of these the privacy contexts by the keyword set {register, registration, sign-up}. Using this notion, we define equality of two contexts as

Definition 35 [Equality of Privacy Contexts] *Two privacy contexts $\mathcal{C}$ and $\mathcal{C}'$ are said to be equal, denoted by $\mathcal{C} = \mathcal{C}'$, if and only if $keywords(\mathcal{C}) = keywords(\mathcal{C}')$.*

In the above example, the privacy contexts *register* and *sign-up* are equal.

6.3.3.1.1 Relationship Between Privacy Contexts We define a relation called 'similarity' between distinct privacy contexts. This relation is defined for every pair of contexts in the privacy context set. The similarity relation is reflexive, symmetric, but not transitive. Each similarity relationship is associated with a *degree of similarity.* For two contexts $\mathcal{C}$ and $\mathcal{C}'$, we denote the degree of similarity by the symbol $sim(\mathcal{C},\ \mathcal{C}')$. This indicates the closeness of the two contexts. Since two distinct privacy contexts related by similarity will not be exactly identical, the degree of similarity will be denoted as a fraction. The exact value of the fraction will be determined by the truster using her domain knowledge. Therefore, for two privacy contexts $\mathcal{C}$ and $\mathcal{C}'$ we have,

$$sim(\mathcal{C},\ \mathcal{C}') = \begin{cases} 1, & \text{if } \mathcal{C} = \mathcal{C}' \\ 0, & \text{if } \mathcal{C} \text{ and } \mathcal{C}' \text{ are unrelated} \\ d \in (0,1), & \text{otherwise} \end{cases}$$

The similarity relationship will be used in setting up privacy preference rule-set when there is no such preference is available in the privacy preference repository for a new context. It will also be used to calculate the initial trust about the trustee on that new context. The degree of similarity together with the trust on the entity in known privacy context will be used to extrapolate the trust on the new privacy context.

6.3.3.2 Privacy Context Similarity Graph

The privacy contexts and the similarity relationships between them is represented using a single graph which we refer to as the *privacy context similarity graph.* Each node n_i in the graph corresponds to a context $\mathcal{C}$ and is labeled by the set $keywords(\mathcal{C})$. We draw a weighted, undirected edge between two nodes n_i and n_j if degree of similarity between the corresponding contexts is between $(0, 1)$. The weight on the edge indicates the degree of similarity between the nodes n_i and n_j. We formally define privacy context similarity graph as

Definition 36 [Privacy Context Similarity Graph] *A privacy context similarity graph $PCSG = \langle \mathcal{N}, \mathcal{E} \rangle$ is a weighted undirected graph satisfying the following conditions*

1. *$\mathcal{N}$ is a set of nodes where each node n_i is associated with a privacy context C_i and is labeled with keywords(C_i), which is the set of keywords associated with the privacy context C_i.*

2. *For each edge $(n_i, n_j) \in \mathcal{E}$, the weight on the edge (n_i, n_j), denoted by $w(n_i, n_j)$ is in $(0, 1)$ and equals to sim(C_i, C_j), where C_i and C_j are represented by n_i and n_j respectively.*

Figure 6.5 illustrates an example of privacy context similarity graph. The four privacy contexts *browse, search, register, login* and the similarity degree between them are shown in the example. The weights are assigned by the truster according to her domain knowledge about these four contexts. We do not discuss further the actual method of assigning the weights.

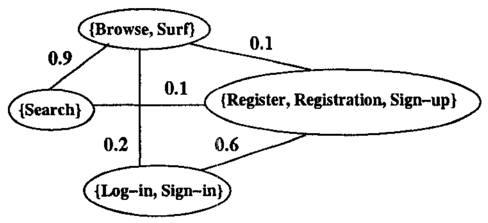


Figure 6.5: Example of a privacy context similarity graph

6.3.4 Reasoning about Privacy Preferences and Trust in Different Privacy Contexts

A user (truster) is likely to have different privacy preferences for different privacy contexts, that is user's privacy preferences depend on underlying contexts. In other words, the user (truster) will perform certain actions, during a communication with a trustee, in one context and other actions in a different privacy context. For example, a user may disclose her address information while making a 'purchase', but not when she is just 'searching' or 'downloading' something on the Web. Such actions, that are to be performed during a communication in a particular privacy context, are specified by the user, according to her own policies, as a set of rules. We call this rule-set for a particular context as *privacy preference rule-set* and formally define it as

Definition 37 **[Privacy Preference Rule-set]** *A user's privacy preference rule-set for a privacy context $\mathcal{C}$, denoted by $\mathcal{R}_{\mathcal{C}}$ is a set of rules or guidelines regarding the actions or steps to be performed by the user (truster) when interacting with another entity (trustee) in the privacy context $\mathcal{C}$.*

A user (truster) can add/delete/modify these rule-sets according to her own policies. The privacy context keeps switching as a user continues her online activities, thereby continuously changing the user's privacy preferences. A user maintains a *privacy preference repository* where she keeps her privacy preference rule-sets for entities (trustees) in specific privacy contexts. However, a user may have a privacy preference rule-set for an entity in some context $\mathcal{C}$, but may not have any preference rule-set in context $\mathcal{C}'$ in the repository. In this scenario the user, while interacting with that entity, needs to make decision about using some existing privacy preference rule-set.

A user also maintains a trust repository where she keeps the trust about entities in particular privacy contexts. For a particular trustee, the user will not have a trust in a new privacy context, irrespective of whether she has or does not have a privacy preference rule-set for that context. After setting up a rule-set the user needs to initiate a trust relationship with the trustee in the new privacy context. This initial trust is calculated using the trust on the trustee in some existing privacy context.

Using an existing privacy preference rule-set from the repository when encountering a new privacy context or an existing privacy context, for which no rule-set is available, is reasonable only when the new context is 'similar' to the context for which a privacy preference rule-set is available in the repository. This is also true when extrapolating the initial trust in the new privacy context. The initial trust should be calculated from the trust on the trustee in some 'similar' privacy context available in trust repository. A privacy context may be related to several other privacy contexts through 'similarity' relationship. Nonetheless, we need to find out which context or set of contexts is conceptually closest to the given context. In other words, we need to find the privacy context or set of privacy contexts that has the highest similarity degree with the given privacy context. For this, we first define the concept of *closest privacy context.*

Definition 38 [**Closest Privacy Contexts**] *Let C be a privacy context. The set of privacy contexts $\mathscr{C}(C) = \{C_1, \ldots, C_m\}$ is defined to be* closest *to C if the following conditions hold:*

1. *for all $i \neq j, 1 \leq i, j \leq m$, $sim(C, C_i) = sim(C, C_j)$.*

2. *for all $i = 1, \ldots, m$, $sim(C, C_i) = max(sim(C, C'))$, where C' is any privacy context that is related to the privacy context C.*

Note, the set $\mathscr{C}(C)$ can be a singleton set. The following Algorithm 7 describes the method for finding out the closest privacy context(s) of a given privacy context. We illustrates the

Algorithm 7 Get the closest privacy context

```
Input: (i) $C$ – the privacy context whose closest one needs to be determined, (ii) $PCSG$
– the privacy context similarity graph in which $C$ is a privacy context.
Output: $\mathscr{C}(C)$ – set of privacy contexts closest to $C$

Procedure $FindClosestContext(C, PCSG)$
$\mathscr{C}(C) = \emptyset$; $relatedContext(C) = \emptyset$;
for all $i$ such that $C_i \in PCSG$ do
  if there is an edge between nodes corresponding to $C$ and $C_i$ in $PCSG$ then
    $relatedContext(C) = relatedContext(C) \cup \{C_i\}$;
  end if
end for
for all $i$ such that $C_j \in relatedContext(C)$ do
  if $sim(C, C_j) = max(sim(C, C_k))$ where $C_k \in relatedContext(C)$ then
    $\mathscr{C}(C) = \mathscr{C}(C) \cup \{C_j\}$;
  end if
end for
return $\mathscr{C}(C)$;
```

above algorithm with an example.

Example 14 *Consider the privacy context similarity graph shown in Figure 6.5. Suppose a user Alice wants to find the closest contexts of the privacy context {Login, Sign-in}. The relatedContext set for this privacy context is {{Browse, Surf}, {Register, Registration, Sign-up}}. The graph shows that sim({Login, Sign-in}, {Browse, Surf}) = 0.2 and* *sim({Login, Sign-in}, {Register, Registration, Sign-up}) = 0.6. Therefore, $\mathscr{C}$({Login, Sign-in}) is found to be the context {Register, Registration, Sign-up} as it has highest similarity degree with the privacy context {Login, Sign-in}.*

If the privacy context similarity graph $PCSG$ has n nodes, then the node corresponding to the privacy context $\mathcal{C}$ can be related to at most $n-1$ nodes in the graph. Therefore, at most $n-1$ edges can be in the set $relatedContext(\mathcal{C})$, from which the closest privacy contexts are determined. Thus, the algorithm has complexity $O(n)$, where n is the number of nodes in the privacy context similarity graph $PCSG$. However, note, if $\mathcal{C}$ was not present in $PCSG$, then the truster needs to update the existing $PCSG$ by including a node corresponding to $\mathcal{C}$ and determining the weighted edges between the new node and the existing nodes. This updated $PCSG$ is then used to find $\mathscr{C}(\mathcal{C})$.

6.3.5 Extrapolating Privacy Preferences from Similar Privacy Contexts

When a user A does not have privacy preferences in a particular privacy context $\mathcal{C}$, we show how she can select one such preference rule-set using one or more similar privacy contexts. Suppose the user A encounters a privacy context $\mathcal{C}$ with an entity B and A does not have a privacy preference rule-set in the repository for $\mathcal{C}$. A finds the set of closest privacy context $\mathscr{C}(\mathcal{C})$ using the Algorithm 7. If $\mathscr{C}(\mathcal{C})$ is a singleton set, say $\{\mathcal{C}'\}$, then the preference rule-set corresponding to $\mathcal{C}'$ is retrieved from the repository and set for context $\mathcal{C}$. Now, suppose $\mathscr{C}(\mathcal{C}) = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ that is, $\mathscr{C}(\mathcal{C})$ is not a singleton set. Suppose for all $i = 1, \ldots, k$, $\mathcal{R}_{\mathcal{C}_i}$ is the privacy preference rule-set corresponding to privacy context $\mathcal{C}_i$. The user A has two choices in this case to set the rule-set for $\mathcal{C}$. She can choose an $\mathcal{R}_{\mathcal{C}_i}$ arbitrarily from the available $\mathcal{R}_{\mathcal{C}_i}$s. Alternately, she constructs the rule-set by taking union of all available $\mathcal{R}_{\mathcal{C}_i}$s. Algorithm 8 describes the method.

6.3.6 Extrapolating Trust from Similar Privacy Contexts

As mentioned earlier, if a truster A does not have a trust about a trustee B in a privacy context $\mathcal{C}$ in her trust repository, then she calculates the initial trust about B in $\mathcal{C}$ using the similar privacy contexts of $\mathcal{C}$. For this evaluation, we discuss two scenarios:

Algorithm 8 Extrapolate privacy preference rule-set for a new privacy context

```
Input: (i) $\mathcal{C}$ – the privacy context for which preference rule-set needs to be set, (ii) $PCSG$ – the privacy context similarity graph in which $\mathcal{C}$ is a context, (iii) The privacy preference rule set repository $\mathscr{R}$.
Output: $\mathcal{R}_{\mathcal{C}}$ – privacy preference rule-set for privacy context $\mathcal{C}$.

Procedure $ConstructPreferenceRules(\mathcal{C}, PCSG, \mathscr{R})$
$\mathcal{R}_{\mathcal{C}} = \emptyset$;
$\mathscr{C}(\mathcal{C}) = FindClosestContext(\mathcal{C}, PCSG)$;
if $\mathscr{C}(\mathcal{C}) = \{\mathcal{C}'\}$ then
  if $\mathcal{R}_{\mathcal{C}'} \in \mathscr{R}$ then
    $\mathcal{R}_{\mathcal{C}} = \mathcal{R}_{\mathcal{C}'}$;
  else
    exit;
  end if
end if
if $\mathscr{C}(\mathcal{C}) = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ then
  Case 1: $\mathcal{R}_{\mathcal{C}} = \mathcal{R}_{\mathcal{C}_j}$ for an arbitrary $j$ such that $1 \leq j \leq k$ and $\mathcal{R}_{\mathcal{C}_j} \in \mathscr{R}$;
  Case 2: $\mathcal{R}_{\mathcal{C}} = \bigcup \mathcal{R}_{\mathcal{C}_j}$ for all $1 \leq j \leq k$ such that $\mathcal{R}_{\mathcal{C}_j} \in \mathscr{R}$;
  return $\mathcal{R}_{\mathcal{C}}$;
end if
```

Scenario 1: $\mathscr{C}(\mathcal{C}) = \mathcal{C}'$ i.e., the closest privacy context set is a singleton set.

In this case A retrieves the normalized trust vector $(A \xrightarrow{\mathcal{C}'} B)_t^N$ of B in privacy context $\mathcal{C}'$ and assigns the value $sim(\mathcal{C}, \mathcal{C}') \times \mathbf{v}(A \xrightarrow{\mathcal{C}'} B)_t^N$ as the initial value for the trust relationship $(A \xrightarrow{\mathcal{C}} B)_t^N$. If $\mathbf{v}(A \xrightarrow{\mathcal{C}'} B)_t^N = \perp$ that is, A has no information about trust on B in privacy context $\mathcal{C}'$, then she needs to extrapolate $(A \xrightarrow{\mathcal{C}'} B)_t^N$. Therefore, this extrapolation can be a recursive process.

Scenario 2: $\mathscr{C}(\mathcal{C}) = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$

A retrieves all the normalized trust vectors $(A \xrightarrow{\mathcal{C}_i} B)_t^N$, $i = 1, 2, \ldots, k$. The initial value for the trust relationship $(A \xrightarrow{\mathcal{C}} B)_t^N$ will be calculated as

$$\mathbf{v}(A \xrightarrow{\mathcal{C}} B)_t^N = \frac{1}{k} \sum_{i=1}^{k} [sim(\mathcal{C}, \mathcal{C}_i) \times \mathbf{v}(A \xrightarrow{\mathcal{C}_i} B)_t^N]$$

To illustrate the above, we continue with our earlier example.

Example 15 *Let Alice now want to extrapolate her trust on www.Books.com in the privacy context {Login, Sign-in}. In our earlier example, Alice finds the closest privacy context* *of the privacy context {Login, Sign-in} as {Register, Registration, Sign-up}. For the sake of brevity, let us denote the privacy context {Login, Sign-in} by Login and the privacy context {Register, Registration, Sign-up} by Register. Suppose Alice has a trust relationship* $(Alice \xrightarrow{Register} www.Books.com)_t^N$ *with the server www.Books.com (trustee) in her trust repository. Suppose* $\mathbf{v}(Alice \xrightarrow{Register} www.Books.com)_t^N = 0.8$. *Then the initial value of the trust relationship* $(Alice \xrightarrow{Login} www.Books.com)_t^N$ *is evaluated as* $0.6 \times 0.8 = 0.48$.

We have presented a trust-based approach to allow personal control over privacy of a user while she is interacting with other entities on the Internet. We have used the vector trust model to evaluate the trust on the recipients of the private data. This trust value allows a user to measure the degree of assurance she can have on the recipient to protect her privacy during a transaction.

6.4 Summary

This chapter demonstrates the use of the vector trust model in different security scenarios. We have considered three different security contexts – access control in open systems, secure routing in ad hoc networks, and preserving privacy while interacting on the Internet. In each of these security contexts, we have discussed how measuring trustworthiness of entities can help to make more reasoned decisions about the security of the systems. In particular, we have shown how vector trust model could be used to evaluate the trustworthiness of entities, involved in these scenarios, to develop more secure systems.

Chapter 7

Conclusions

7.1 Contribution and Significance

The research conducted in this dissertation has significance in security research. The main objective of this dissertation work is to propose a flexible trust model which would help us to take more reasoned decision about trustworthiness of an entity in the face of incomplete, ambiguous, or unknown information. Reasoned decision regarding trustworthiness of entities is important for designing secure system for different environments including the Internet. Internet and other open systems like virtual organizations, peer-to-peer networks, e-commerce applications involve entities whose identity is not known in advance. These applications use different information from several entities and work under the premise of trust. However, all these entities are not "trusted" at the same level. Also, the semantic and representation of trust vary over systems and applications. The proposed model helps a user to evaluate the amount of confidence he/she has in his/her decision to accept the assumed or measured amount of competency of a particular system when the system's behavior is influenced by other entities (including human beings). This is illustrated in the following discussion.

Consider the questions posed earlier in the context of the e-commerce example (scenario 2) presented in Section 2.3.2. We need to determine how much trust the customer Alice can have on the Web service providers regarding her privacy protection. One possible way of doing this with this model is as follows. At the beginning Alice starts with a neutral position about the trustworthiness of S_1 and S_2. As time progress, Alice will gradually begin to establish trust relationships with the two. The degree of trust that she establishes with a service provider will depend on different factors. For example, she may become aware that a provider does not store a customer's credit card information after the transaction is over. Alice may begin to have positive interactions with S_1 and some positive and some negative interactions with S_2. At some point then Alice (perhaps) evaluates that S_1 is trusted to a degree of 0.75 and S_2 to degree of 0.25. These trust scores provide a relative trustworthiness of the two entities to Alice. How she uses this information is her choice. Nonetheless, the model helps her to take a more effective decision (e.g., buy the air-ticket from S_1 or restrict the disclosure of personal information to S_2).

The outcome of this research is a trust model that helps to define, represent, evaluate, and manage 'trust' (and 'distrust'), even when all of the relevant information is not available. The systems build around the proposed formalism of trust will be able to assign a fine-grained assurance level, about other involved entities, to the user for making a reasoned decision regarding trustworthiness of those entities. That is, the model will allow the user to evaluate the risks involved in using a system in a better manner and thus to design more secure systems.

The primary contributions of this dissertation are:

1. A flexible quantitative model of trust based on different parameters and providing multilevel of trust. The model is extensible as the parameters are independent to each other. Addition of new parameters does not affect the other features of the model. The model can evaluate trust even when all the relevant information to do so is not available.

2. Formalism of trust context and relationship between different contexts. This formalism can help to make reasoned decisions about trust in a context when no information is available for that context.

In particular, the model contributes the following:

1. Formally differentiate between trust, distrust, neutrality. Representation of trust, distrust, and neutrality as numerically measurable objects using a single continuous

scale $[-1, 1]$ to represent trust, distrust, and neutrality. 'Unknown' state is also captured and represented using the symbol $\perp$. Continuous numeric scale helps to define fine-granular levels of trust and distrust.

2. Formal definition of trust context and relationships between different contexts. Formalization of context helps to use the model in different security scenarios. It also enhances the interoperability of the model. Relationship between contexts is useful to reason about trustworthiness in a context when no information is available in that context.

3. Proposition of different independent parameters that influence evaluation of trust and distrust, together with mechanisms to numerically measure these parameters. Parameters are independent – adding a new parameter or dropping an existing parameter will not result in changing other constructs or methods to a great extent. Also, it helps in evaluation of trust based on different information. This makes the model, unlike the existing trust models, easily extensible.

4. Propensity to trust that defines relative importance of different parameters. This helps in wide applicability of the model. If an application does not have a scope to use a parameter, the corresponding weight in trust-parameter weight vector just needs to be set to 0. Therefore, it is not needed to search for a suitable model that fits the application. A single model (vector trust model) is sufficient in different applications.

5. Dependence of trust on time as well as previous trust. Dependence on time captures more "human way" of evaluating trust to corroborate the observation from social science [Hir84] – *"time the great leveler"*. Dependence on previous trust helps to reason about trust if current information is not available, but some old information is in store. Methods are so formed that the user has more control on how much effect of old trust he/she wants to have in current evaluation.

6. Formal method to compare two different trust relationships. The method can differentiate two trust relationships even when their corresponding trust values are same. It helps a user to make more fine-granular comparison decisions.

7. Mechanisms to combine different trust relationships. The method is flexible to assign relative importance to different trust relationships involved in the combination. The combination operator can also be chosen suitably according to the application need or organizational policy. Method has the scope of considering different cases (one-to-many, may-to-one, many-to-many). It also guides how to consider the effect of reconfiguration of a group of truster or trustee. The last two points, especially, are useful in application of the model in co-operative domain.

The above contributions demonstrate that the model is useful in making fine-grained security related decisions in different security contexts where other mechanisms or other trust models are not sufficient to make such decisions.

7.2 Future Work

An initial version of the vector trust model was proposed in [RC04]. This dissertation presents the current extended version of the model, its validation and some potential applications. However, still there are some possible extensions of the current work in future. At present we have not addressed the 'trust chain' or 'trust transitivity' concept. A restricted form of trust transitivity is used in the evaluation of recommendations where positive, negative or neutral judgments are scaled with positive trust only. The trust model can be extended to answer the questions like "If A distrusts B and B distrusts C, then what is the result of trust relationship between A and C?", "If A is neutral about B and B trusts C, then what is the result of trust relationship between A and C?" etc.

The current work have not addressed the issue of determining different policies involved in trust evaluation. At present we have assumed that the truster has existing policies, but how to design these policies in the first hand? How to categorize the different underlying parameters as trust-positive, trust-negative or trust-neutral? What will be an appropriate guideline for that? These are some of the questions that can be considered to be addressed in future. Work on the decision theory may be useful in this regard. However, we believe, these policies will depend on a user's or organization's other policies (e.g., personal preferences or business policies) as well as on the target applications.

Last but not the least, at present there is no tool support for this model. A tool developed to handle the trust management system would be useful. Future work in developing such a tool involves modules to gather, manage, and store trust related information. The tool may work as an independent entity or can be embedded into an existing system (e.g., as a plug-in for a browser). A query language can be developed or an existing query language can be used to interact with the trust management system supported by the tool.

REFERENCES

[AABF02] Nabil. R. Adam, Vijayalakshmi Atluri, Elisa Bertino, and Elena Ferrari. A Content-Based Authorization Model for Digital Libraries. *IEEE Transactions on Knowledge and Data Engineering*, 14(2):296–315, March 2002.

[ACR99] Mark S. Ackerman, Lorrie F. Cranor, and Joseph Reagle. Privacy in E-Commerce: Examining User Scenarios and Privacy Preferences. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, pages 1–8, Denver, Colorado, USA, 1999. ACM Press.

[AD01] Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-2-Peer Information System. In Henrique Paques, Ling Liu, and David Grossman, editors, *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001)*, pages 310–317, Atlanta, Georgia, USA, October 2001. ACM Press.

[Adl05] Terry. R. Adler. The Swift Trust Partnership: A Project Management Exercise Investigating the Effects of Trust and Distrust in Outsourcing Relationships. *Journal of Management Education*, 29(5):714–737, October 2005.

[AF03] Martín Abadi and Cédric Fournet. Access Control Based on Execution History. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS 2003)*, pages 107–121, San Diego, California, USA, February 2003. The Internet Society.

[AHNRR02] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In W. Douglas Maughan and Nitin H. Vaidya, editors, *Proceedings of ACM Workshop on Wireless Security (WiSe'02)*, pages 21–30, Atlanta, Georgia, USA, September 2002. ACM Press.

[ALRL04] Algirdas Avižienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, January 2004.

[And92] Syed S. Andaleeb. The Trust Concept: Research Issues for Channels of Distribution. *Research in Marketing*, 11:1–34, 1992.

[ARH97] Alfarez Abdul-Rahman and Stephen Hailes. A Distributed Trust Model. In *Proceedings of the 1997 New Security Paradigms Workshop (NSPW'97)*, pages 48–60, Langdale, Cumbria, United Kingdom, September 1997. ACM Press.

[Bai86] Annette Baier. Trust and Antitrust. *Ethics*, 96(2):231–260, 1986.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[Bar83] Bernard Barber. *The Logic and Limits of Trust.* Rutgers University Press, New Brunswick, New Jersey, USA, March 1983.

[BB04] Sonja Buchegger and Jean-Yves Le Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems*, Cambridge, Massachusetts, USA, June 2004.

[BBBB03] Thomas G. Brashear, James S. Boles, Danny N. Bellenger, and Charles M. Brooks. An Empirical Test of Trust-Building Processes and Outcomes in Sales Manager-Salesperson Relationships. *Journal of the Academy of Marketing Science*, 31(2):189–200, April 2003.

[BBG04] Rafae Bhatti, Elisa Bertino, and Arif Ghafoor. A Trust-based Context-Aware Access Control Model for Web-Services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 184–191, San Diego, California, USA, June 2004. IEEE Computer Society.

[BBK94] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of Trust in Open Networks. In Dieter Gollmann, editor, *Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS '94)*, volume 875 of *Lecture Notes in Computer Science*, pages 3–18, Brighton, UK, November 1994. Springer-Verlag.

[BCD05] Elisa Bertino, Barbara Catania, and Maria Luisa Damiani. GEO-RBAC: A Spatially Aware RBAC. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*, pages 29–37, Stockholm, Sweeden, June 2005. ACM Press.

[Ber06] Daniel J. Bernstein. SYN Cookies, Accessed November 20 2006. http://cr.yp.to/syncookies.html.

[BFIK99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. The KeyNote Trust Management System Version 2. Internet Society, Network Working Group. RFC 2704, 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized Trust Management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, California, USA, May 1996. IEEE Computer Society.

[BG00] Michael Bacharach and Diego Gambetta. Trust as Type Identification. In Cristiano Castelfranchi and Yao-Hua Tan, editors, *Trust and Deception in Virtual Societies*, chapter 1, pages 1–26. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 2000.

[BH94] Jay B. Barney and Mark H. Hansen. Trustworthiness as a Source of Competitive Advantage. *Strategic Management Journal*, 15:175–190, 1994.

[BJBG03] Rafae Bhatti, James B.D. Joshi, Elisa Bertino, and Arif Ghafoor. Access Control in Dynamic XML-based Web-Services with X-RBAC. In Liang-Jie Zhang, editor, *Proceedings of the 1st International Conference on Web Services (ICWS'03)*, pages 243–249, Las Vegas, Nevada, USA, June 2003. CSREA Press.

[BKS95] Piero Bonatti, Sarit Kraus, and V.S. Subrahmanian. Foundations on Secure Deductive Databases. *IEEE Transactions on Knowledge and Data Engineering*, 7(3):406–422, June 1995.

[BLG$^+$00] Andrew Barkley, Steve Liu, Quoc Thong Le Gia, Matt Dingfield, and Yashodhan Gokhale. A Testbed for Study of Distributed Denial of Service Attacks (WA 2.4). In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security (IAW'00)*, pages 218–223, United States Military Academy, West Point, New York, USA, June 2000. IEEE Computer Society.

[BS00] Piero Bonatti and Pierangela Samarati. Regulating Service Access and Information Release on the Web. In *Proceedings of the 7th ACM Conference on Computer and Communication Security (CCS'00)*, pages 134–143, Athens, Greece, November 2000. ACM Press.

[BSF02] Lujo Bauer, Michael A. Schneider, and Edward W. Felten. A General and Flexible Access-Control System for the Web. In *Proceedings of the 11th USENIX Security Symposium*, pages 93–108, San Francisco, California, USA, August 2002.

[BSSW02] Dirk Balfanz, Diana K. Smetters, Paul Stewart, and H. Chi Wong. Talking to Strangers: Authentication in Adhoc Wireless Networks. In *Symposium on Network and Distributed Systems Security (NDSS '02)*, San Diego, California, USA, February 2002.

[C$^+$04] Lorrie.F. Cranor et al. The Platform for Privacy Preferences 1.1 (P3P 1.1) Specification. http://www.w3.org/tr/2004/wd-p3p11-20040720, World Wide Web Consortium, July 2004.

[CE95] Mathew S. Corson and Anthony Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, 1(1):61–82, February 1995.

[Cen97] CERT Coordination Centre. IP Spoofing Attacks and Hijacked Terminal Connections. http://www.cert.org/advisories/CA-1995-01.html, 1997.

[Cen00] CERT Coordination Centre. TCP SYN Flooding and IP Spoofing Attacks. http://www.cert.org/advisories/CA-1996-21.html, November 2000.

[CJ95] Steven Curral and Timothy Judge. Measuring Trust Between Organizational Boundary Role Persons. *Organizational Behaviour and Human Decision Processes*, 64(2):151–170, November 1995.

[COB03] David W. Chadwick, Alexander Otenko, and Edward Ball. Role-Based Access Control with X.509 Attribute Certificates. *IEEE Internet Computing*, 7(2):62–69, March/April 2003.

[Coo91] Roger M. Cooke. *Experts in Uncertainty: Opinion and Subjective Probability in Science.* Oxford University Press, New York, USA, 1991.

[CORa] CORAS (2000–2003). IST-2000-25031 CORAS: A Platform for Risk Analysis of Security Critical Systems.

[Corb] AT&T Corp. Privacy Bird Project. http://www.privacybird.org.

[CPF97] Marvin S. Cohen, Raja Parasuraman, and Jared T. Freeman. Trust in Decision Aids: A Model and a Training Strategy. Technical Report USAATCOM TR 97-D-4, Cognitive Technologies Inc., Fort Eustis, Virginia, USA, 1997.

[CPR07] Sudip Chakraborty, Nayot Poolsappasit, and Indrajit Ray. Reliable Delivery of Event Data from Sensors to Actuators in Pervasive Computing Environments. In Steve Barker and Gail-Joon Ahn, editors, *Proceedings of 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec'07)*, volume 4602 of *Lecture Notes in Computer Science*, pages 77–92, Redondo Beach, California, USA, July 2007. Springer.

[CR06] Sudip Chakraborty and Indrajit Ray. TrustBAC - Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems. In *Proceedings of 11th ACM Symposium on Access Control Models and Technologies (SACMAT'06)*, pages 49–58, Lake Tahoe, California, USA, June 2006. ACM Press.

[CTWS02] Eve Cohen, Roshan K. Thomas, William Winsborough, and Deborah Shands. Models for Coalition-based Access Control (CBAC). In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT'02)*, pages 97–106, Monterey, California, USA, June 2002. ACM Press.

[CWLG97] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In *5th IEEE Singapore International Conference on Networks (SICON'97)*, pages 197–211, Kent Ridge, Singapore, April 1997. IEEE Computer Society.

[Das88] Partha Dasgupta. Trust as a Commodity. In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, chapter 4, pages 49–72. Basil Blackwell, New York, USA, 1988.

[Deu60] Morton Deutsch. The Effect of Motivational Orientation upon Trust and Suspicion. *Human Relations*, 13:123–139, 1960.

[Deu73] Morton Deutsch. *The Resolution of Conflict: Constructive and Destructive Processes.* Yale University Press, New Haven, Connecticut, USA, 1973.

[DF02] Josep Domingo-Ferrer, editor. *Inference Control in Statistical Databases: From Theory to Practice*, volume 2316 of *Lecture Notes in Computer Science.* Springer-Verlag, London, UK, 1 edition, May 2002.

[Dri78] James W. Driscoll. Trust and Participation in Organizational Decision Making as Predictors of Satisfaction. *Academy of Management Journal*, 21(1):44–56, March 1978.

[DRWT97] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks. *IEEE Personal Communications Magazine*, 4(1):36–45, February 1997.

[DVPS03] Ernesto Damiani, Sabrina De Capitani Di Vimercati, Stefano Paraboschi, and Pierangela Samarati. Managing and Sharing Servants' Reputations in P2P Systems. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):840–854, July-August 2003.

[DVS05] Ernesto Damiani, Sabrina De Capitani Di Vimercati, and Pierangela Samarati. New Paradigm for Access Control in Open Environments. In *Proceedings of the 5th IEEE Symposium on Signal Processing and Information Technology (ISSPIT'05)*, pages 540–545, Athens, Greece, December 2005.

[EAC98] Guy Edjlali, Anurag Acharya, and Vipin Chaudhary. History-based Access Control for Mobile Code. In *Proceedings of the 5th ACM Conference on Computer and Communication Security (CCS'98)*, pages 38–48, San Francisco, California, USA, November 1998. ACM Press.

[EU 01] EU Project EP-27046-ACTIVE. EP-27046-ACTIVE, Final Prototype and User Manual, D4.2.2, Ver. 2.0, 2001-02-22., 2001.

[FK92] David F. Ferraiolo and D.Richard Kuhn. Role-Based Access Controls. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Bultimore, Maryland, USA, October 1992.

[FSG$^+$01] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D.Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and Systems Security*, 4(3):224–274, August 2001.

[GB81] Eli M. Gafni and Dimitri P. Bertsekas. Distributed Algorithms for Generating Loop-free Routes in Network with Frequently Changing Topology. *IEEE Transactions on Communications*, 29(1):11–18, January 1981.

[GHKM00] L.H.J. Goossens, Frederick T. Harper, Bernard C.P. Kraan, and Henri Métivier. Expert Judgement for a Probabilistic Accident Consequence Uncertainty Analysis. *Radiation Protection and Dosimetry*, 90(3):295–301, 2000.

[Gla97] Henry M. Gladney. Access Control for Large Collections. *ACM Transactions on Information Systems*, 15(2):154–194, April 1997.

[GM02] Jeremy Goecks and Elizabeth Mynatt. Enabling Privacy Management in Ubiquitous Computing Environments Through Trust and Reputation. In *Proceedings of CSCW 2002 Workshop on Privacy in Digital Environments*, New Orleans, Louisiana, November 2002.

[GMPT01] Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas. Flexible Team-based Access Control Using Contexts. In *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, pages 21–27, Chantily, Virginia, USA, May 2001. ACM Press.

[Goo00] David Good. *Individuals, Interpersonal Relations, and Trust*, chapter 3 of Trust: Making and Breaking Cooperative Relations, pages 31–48. Electronic edition, 2000.

[Gru93] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GS00] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4), Fourth Quarter, 2000.

[GT95] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, Mobile, Multimedia Radio Network. *Wireless Networks*, 1(3):255–265, 1995.

[HCRR08] Siv H. Houmb, Sudip Chakraborty, Indrakshi Ray, and Indrajit Ray. Estimating Security Level of a Security Solution using Trust-based Information Aggregation. Submitted and under review, 2008.

[HGF$^+$05a] Siv H. Houmb, Geri Georg, Robert B. France, James M. Bieman, and Jan Jürjens. Cost-Benefit Trade-Off Analysis using BBN for Aspect-Oriented Risk-Driven Development. In *Proceedings of 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005)*, pages 195–204, Shanghai, China, June 2005. IEEE Computer Society.

[HGF$^+$05b] Siv H. Houmb, Geri Georg, Robert B. France, Raghu Reddy, and James M. Bieman. Predicting Availability of Systems using BBN in Aspect-Oriented Risk-Driven Development (AORDD). In *2nd Symposium on Risk Management and Cyber-Informatics (RMCI '05)*, pages 396–403, Orlando, Florida, USA, July 2005.

[Hir84] Albert O. Hirschman. Three Ways of Compilcating Some Categories of Economic Discourse. *American Economic Review*, 74(2):89–96, 1984.

[HJ03] Anthony Harrington and Christian Jensen. Cryptographic Access Control in a Distributed File System. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT'03)*, pages 158–165, Como, Italy, June 2003. ACM Press.

[HPJ02] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In Ian F. Akyildiz, Jason Yi-Bing Lin, Ravi Jain, Vaduvur Bharghavan, and Andrew T. Campbell, editors, *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom'02)*, pages 12–23, Atlanta, Georgia, USA, September 2002. ACM Press.

[HS96] John Humphrey and Hubert Schmitz. Trust and Economic Development. Discussion Paper 355, 1996. Institute of Development Studies, Brighton, UK.

[HW04] Bernardo A. Huberman and Fang Wu. The Dynamics of Reputations. *Journal of Statistical Mechanics: Theory and Experiment*, page P04006, April 2004.

[JBLG05] James B.D. Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, January 2005.

[JF00] Andrew J.I. Jones and Babak S. Firozabadi. On the Characterization of a Trusting Agent – Aspects of a Formal Approach. In Cristiano Castelfranchi and Yao-Hua Tan, editors, *Trust and Deception in Virtual Societies*, chapter 8, pages 157–168. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 2000.

[JI02] Audun Jøsang and Roslan Ismail. The Beta Reputation System. In *Proceedings of 15th Bled Electronic Commerce Conference: e-Reality: Constructing the e-Economy*, Bled, Slovenia, June 2002.

[JM99] Sara Jones and Philip Morris. TRUST-EC: Requirements for Trust and Confidence in E-Commerce. European Communities EUR Report 2, European Comission, Joint Research Centre, Report of the Workshop held in Luxembourg, April 1999.

[Jøs97] Audun Jøsang. Artificial Reasoning with Subjective Logic. In Abhaya C. Nayak and Maurice Pagnucco, editors, *Proceedings of the 2nd Australian Workshop on Commonsense Reasoning*, Perth, Australia, December 1997.

[Jøs98] Audun Jøsang. A Subjective Metric of Authentication. In Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, and Dieter Gollmann, editors, *Proceedings of the 5th European Symposium on Research in Computer Security (ESORICS'98)*, volume 1485 of *Lecture Notes in Computer Science*, pages 329–344, Louvain-la-Neuve, Belgium, September 1998. Springer-Verlag.

[Jøs99] Audun Jøsang. An Algebra for Assessing Trust in Certification Chains. In *Proceedings of the 1999 Network and Distributed Systems Security Symposium (NDSS'99)*, San Diego, California, USA, February 1999. Internet Society.

[JSS97] Sushil Jajodia, Pierangela Samarati, and V.S. Subrahmanian. A Logical Language for Expressing Authorizations. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 31–42, Oakland, California, USA, May 1997. IEEE Computer Society.

[JT99] Catholijn M. Jonker and Jan Treur. Formal Analysis of Models for the Dynamics of Trust Based on Experience. In Francisco J. Garijo and Magnus Boman, editors, *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent System Engineering (MAAMAW 1999)*, volume 1647 of *Lecture Notes in Computer Science*, pages 221–231, Valencia, Spain, June-July 1999. Springer-Verlag.

[KC98] Anil Kini and Joobin Choobineh. Trust in Electronic Commerce: Definition and Theoritical Considerations. In *Proceedings of the 31st Annual Hawaii International Conference on System Sciences (HICSS-31)*, volume 4, pages 51–61, Big Island, Hawaii, USA, January 1998. IEEE Computer Society.

[KL01] David Karig and Ruby Lee. Remote Denial of Service Attacks and Countermeasures. Technical Report CE-L2001-002, Department of Electrical Engineering, Princeton University, Princeton, New Jersey, USA, October 2001.

[KNS05] Karl Krukow, Mogens Nielsen, and Vladimiro Sassone. A Framework for Concrete Reputation-Systems with Applications to History-Based Access Control. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *Proceedings of 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 260–269, Alexandria, Virginia, USA, November 2005. ACM Press.

[KSGM03] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th Internatioal Conference on World Wide Web*, pages 640–651, Budapest, Hungary, May 2003.

[LcC06] Shibiao Lin and Tzi cker Chiueh. A Survey on Solutions to Distributed Denial of Service Attacks. Technical report RPE TR-201, Department of Computer Science, Stony Brook University, Stony Brook, New York, USA, September 2006.

[LM03a] Ninghui Li and John C. Mitchell. Datalog with Constraints: A Foundation for Trust-management Languages. In *Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages (PADL'03)*, pages 58–73, New Orleans, Louisiana, USA, January 2003.

[LM03b] Ninghui Li and John C. Mitchell. RT: A Role-based Trust Management Framework. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX-III)*, pages 201–212, Washington D.C., USA, April 2003. IEEE Computer Society.

[LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a Role-Based Trust-Management Framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130, Oakland, California, USA, May 2002. IEEE Computer Society.

[LTG06] Roy J. Lewicki, Edward C. Tomlinson, and Nicole Gillespie. Models of Interpersonal Trust Development: Theoretical Approaches, Empirical Evidence, and Future Directions. *Journal of Management*, 32(6):991–1022, December 2006.

[Luh79] Niklas Luhmann. *Trust and Power*. Wiley, Chichester, 1979.

[LW85] David Lewis and Andrew J. Weigert. Social Atomism, Holism and Trust. *Sociological Quarterly*, 26(4):455–471, 1985.

[LWM03] Ninghui Li, William H. Winsborough, and John C. Mitchell. Beyond Proof-of-Compliance: Safety and Availability Analysis in Trust Management. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 123–139, Oakland, California, USA, May 2003. IEEE Computer Society.

[Man00] Daniel W. Manchala. E-Commerce Trust Metrics and Models. *IEEE Internet Computing*, 4(2):36–44, March 2000.

[Mar94] Stephen P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland, UK, April 1994.

[MDS95] Roger C. Mayer, James H. Davis, and F. David Schoorman. An Integrative Model of Organizational Trust. *Academy of Management Review*, 20(3):709–734, 1995.

[Mee84] Barbara F. Meeker. Cooperative Orientation, Trust, and Reciprocity. *Human Relations*, 37(3):225–243, March 1984.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):Article 3, March 2007.

[MMH02] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A Computational Model of Trust and Reputation for E-Businesses. In *35th Annual Hawaii International Conference on System Sciences (HICSS-35)*, volume 7, Big Island, Hawaii, USA, 2002. IEEE Computer Society.

[MS79] Byron A. Matthews and Eliot Shimoff. Expansion of Exchange: Monitoring Trust Levels in Ongoing Exchange Relations. *Journal of Conflict Resolution*, 23(3):538–560, September 1979.

[NM02] David H. Nguyen and Elizabeth D. Mynatt. Privacy Mirrors: Understanding and Shaping Socio-technical Ubiquitous Computing Systems. Technical Report GIT-GVU-02-16, Georgia Institute of Technology, 2002.

[Øst03] Mona E. Østvang. The Honeynet Project, Phase 1: Installing and Tuning Honeyd using LIDS, 2003. Project assignment, Norwegian University of Science and Technology.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers. In *Proceedings of ACM Conference on Communication Architectures, Protocols and Applications (SIGCOMM'94)*, pages 234–244, London, UK, August-September 1994. ACM Press.

[PH03a] Panagiotis. Papadimitratos and Zygmunt Haas. Secure Data Transmission in Mobile Ad Hoc Networks. In *Proceedings of 2nd ACM Workshop on Wireless Security (WiSe'03)*, pages 41–50, San Diego, California, USA, September 2003. ACM Press.

[PH03b] Joon S. Park and Junseok Hwang. Role-based Access Control for Collaborative Enterprise in Peer-to-Peer Computing Environments. In *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT'03)*, pages 93–99, Como, Italy, June 2003. ACM Press.

[Pur01] Steve Purser. A Simple Graphical Tool For Modelling Trust. *Computers & Security*, 20(6):479–484, September 2001.

[Ran88] P. Venkat Rangan. An Axiomatic Basis of Trust in Distributed Systems. In *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, pages 204–211, Oakland, California, USA, April 1988. IEEE Computer Society.

[RC04] Indrajit Ray and Sudip Chakraborty. A Vector Model of Trust for Developing Trustworthy Systems. In Pierangela Samarati, Peter Y. A. Ryan, Dieter Gollmann, and Refik Molva, editors, *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS'04)*, volume 3193 of *Lecture Notes in Computer Science*, pages 260–275, Sophia Antipolis, France, September 2004.

[RC08] Indrajit Ray and Sudip Chakraborty. Facilitating Privacy Related Decisions in Different Privacy Contexts on the Internet By Evaluating Trust in Recipients of Private Data. In *Proceedings of 23rd IFIP International Information Security Conference (SEC 2008)*, Milan, Italy, September 2008.

[RJ96] Lars Rasmusson and Sverker Jansson. Simulated Social Control for Secure Internet Commerce. In C. Meadows, editor, *Proceedings of the 1996 New Security Paradigms Workshop*, pages 18–25, Lake Arrowhead, California, USA, September 1996. ACM Press.

[Rot67] Julian B. Rotter. A New Scale for the Measurement of Interpersonal Trust. *Journal of Personality*, 35:651–665, 1967.

[RRC08] Indrakshi Ray, Indrajit Ray, and Sudip Chakraborty. An Interoperable Context Sensitive Model of Trust. *Journal of Intelligent Information Systems*, 2008. In Press.

[SA02] Narendar Shankar and William A. Arbaugh. On Trust for Ubiquitous Computing. In *Workshop on Security for Ubiquitous Computing (UBICOMP'02)*, October 2002. Invited paper.

[Sam01] Pierangela Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, November/December 2001.

[Sat88] Kaori Sato. Trust and Group Size in a Social Dilemma. *Japanese Psychological Review*, 30(2):88–93, 1988.

[SBWS05] Natalia Stakhanova, Samik Basu, Johnny Wong, and Oleg Stakhanov. Trust Framework for P2P Networks Using Peer-profile Based Anomaly Technique. In *Proceedings of the Second International Workshop on Security in Distributed Computing Systems (SDCS) (25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05))*, volume 2, pages 203–209, Columbus, Ohio, USA, June 2005. IEEE Computer Society.

[SCFY96] Ravi Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *IEEE Computer*, 29(2):38–47, February 1996.

[SDB03] Brian Shand, Nathan Dimmock, and Jean Bacon. Trust for Ubiquitous, Transparent Collaboration. In *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pages 153–160, Dallas, Ft. Worth, Texas, USA, March 2003.

[SFWC04] Natalia Stakhanova, Sergio Ferrero, Johnny Wong, and Ying Cai. A Reputation-based Trust Management in Peer-to-Peer Network Systems. In David A. Bader and Ashfaq A. Khokhar, editors, *Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems (ISCA PDCS'04)*, pages 510–515, San Francisco, California, USA, September 2004.

[Sha87] Susan P. Shapiro. The Social Control of Impersonal Trust. *The American Journal of Sociology*, 93(3):623–658, 1987.

[SK03] Aaron Schiff and John Kennes. The Value of Reputations Systems. In *Proceedings of the 1st Summer Workshop in Industrial Organization (SWIO)*, Auckland, New Zealand, March 2003.

[SS98] Pierangela Samarati and Latanya Sweeney. Generalizing Data to Provide Anonymity When Disclosing Information (abstract). In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Seattle, Washington, USA, June 1998. ACM Press.

[Ste07] Michael Stevens. Use of Trust Vectors in Support of the CyberCraft Initiative. Master's thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, USA, March 2007.

[SUP04] Ali A. Selçuk, Ersin Uzun, and Mark R. Pariente. A Reputation-Based Trust Management System for P2P Networks. In *4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, pages 251–258, Chicago, Illinois, USA, April 2004. IEEE Computer Society.

[Swe02] Latanya Sweeney. K-Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[Swi67] Robert L. Swinth. The Establishment of the Trust Relationship. *Journal of Conflict Resolution*, 11(3):335–344, 1967.

[SZ05] Ravi Sandhu and Xinwen Zhang. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*, pages 147–158, Stockholm, Sweeden, June 2005. ACM Press.

[Tho97] Roshan K. Thomas. Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments. In *Proceedings of the Second ACM Workshop on Role-based Access Control (RBAC'97)*, pages 13–19, Fairfax, Virginia, USA, November 1997. ACM Press.

[Toh96] Chai-Keong Toh. A Novel Distributed Routing Protocol To Support Ad hoc Mobile Computing. In *Proceedings of 1996 IEEE 15th Annual International Phoenix Conference on Computers and Communication (IPCCC'96)*, pages 480–486, Scottsdale, Arizona, USA, March 1996. IEEE Computer Society.

[TPC00] Yun Teng, Vir V. Phoha, and Ben Choi. Design of Trust Metrics Based on Dempster-Shafer Theory. citeseer.ist.psu.edu/461538.html, 2000.

[UG96] Mike Uschold and Michael Gruninger. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2):93–155, March 1996.

[vdASC01] Ty van den Akker, Quinn O. Snell, and Mark J. Clement. The YGuard Access Control Model: Set-based Access Control. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, pages 75–84, Chantily, Virginia, USA, May 2001. ACM Press.

[WCJS97] Marianne Winslett, Neil Ching, Vicki Jones, and Igor Slepchin. Assuring Security and Privacy for Digital Library Transactions on the Web: Client and Server Security Policies. In *Proceedings of the IEEE International Forum on Research and Technology Advances in Digital Libraries (ADL'97)*, pages 140–151, Washington D.C., USA, May 1997. IEEE Computer Society.

[WJI05] Andrew Whitby, Audun Jøsang, and Jadwiga Indulska. Filtering Out Unfair Ratings in Bayesian Reputation Systems. *Icfain Journal of Management Research*, 4(2):48–64, February 2005.

[WL04] Horst F. Wedde and Mario Lischka. Role-Based Access Control in Ambient and Remote Space. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT'04)*, pages 21–30, Yorktown Heights, New York, USA, June 2004. ACM Press.

[WV03] Yao Wang and Julita Vassileva. Bayesian Network-Based Trust Model. In *2003 IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 372–378, Halifax, Canada, October 2003. IEEE Computer Society.

[XL03] Li Xiong and Ling Liu. A Reputation-Based Trust Model For Peer-To-Peer Ecommerce Communities. In *Proceedings of IEEE Conference on E-Commerce (CEC'03)*, pages 275–284, Newport Beach, California, USA, June 2003. IEEE Computer Society.

[XL04] Li Xiong and Ling Liu. PeerTrust: Supporting Reptation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):843–857, July 2004.

[YKB93] Raphael Yahalom, Birgit Klein, and Thomas Beth. Trust Relationship in Secure Systems: A Distributed Authentication Perspective. In *Proceedings of the 1993 IEEE Computer Society Symposium on Security and Privacy*, pages 150–164, Oakland, California, USA, May 1993. IEEE Computer Society.

[YKB94] Raphael Yahalom, Birgit Klein, and Thomas Beth. Trust-based Navigation in Distributed Systems. *Computing Systems*, 7(1):45–73, Winter 1994.

[YNK01] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-Aware Ad Hoc Routing for Wireless Networks. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 299–302, Long Beach, California, USA, October 2001. ACM Press.

[Zan72] Dale E. Zand. Trust and Management Problem Solving. *Administrative Science Quarterly*, 17(2):229–239, June 1972.

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30, 1999.

[ZMHT05] Charikleia Zouridaki, Brian L. Mark, Marek Hejmo, and Roshan K. Thomas. A Quantitative Trust Establishment Framework for Reliable Data Packet Delivery in MANETs. In Vijay Atluri, Peng Ning, and Wenliang Du, editors, *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05)*, pages 1–10, Alexandria, Virginia, USA, November 2005. ACM Press.

[ZMP98] Akbar Zaheer, Bill McEvily, and Vincenzo Perrone. Does Trust Matter? Exploring the Effects of Interorganizational and Interpersonal Trust on Performance. *Organization Science*, 9(2):141–159, March-April 1998.