A Generalized Trust Model using Network Reliability

by

Glenn R. Mahoney
B.Sc., University of Victoria, 1988

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

We accept this thesis as conforming
to the required standard

Supervisors: Dr. W. Myrvold and Dr. G. C. Shoja

ABSTRACT

Economic and social activity is increasingly reflected in operations on digital objects and network-mediated interactions between digital entities. Trust is a prerequisite for many of these interactions, particularly if items of value are to be exchanged. The problem is that automated handling of trust-related concerns between distributed entities is a relatively new concept and many existing capabilities are limited or application-specific, particularly in the context of informal or ad-hoc relationships.

This thesis contributes a new family of probabilistic trust metrics based on Network Reliability called the Generic Reliability Trust Model (GRTM). This approach to trust modelling is demonstrated with a new, flexible trust metric called Hop-count Limited Transitive Trust (HLTT), and is also applied to an implementation of the existing Maurer Confidence Valuation (MCV) trust metric. All metrics in the GRTM framework utilize a common probabilistic trust model which is the solution of a general reliability problem. Two generalized algorithms are presented for computing GRTM based on inclusion-exclusion and factoring. A conservative approximation heuristic is defined which leads to more practical algorithm performance. A JAVA-based implementation of these algorithms for HLTT and MCV trust metrics is used to demonstrate the impact of the approximation. An XML-based trust-graph representation and a random power-law trust graph generator is used to simulate large informal trust networks.

Examiners:

Table of Contents

List of Tables

List of Figures

Acknowledgement

This work was made possible by the support and love of my family, especially my wife Jan and my parents David and Colleen Mahoney.

Dr. Ali Shoja, my principle supervisor, patiently supported my exploration and had the courage to allow me to stand on the ideas that emerged. Dr. Wendy Myrvold brought a disciplined and expert focus to the core ideas and the extra energy to help take the results to conclusion. Members the University of Victoria computer science research groups PANDA, especially Dr. Eric Manning, and CAG provided feedback at various stages in the development of the concepts and results. They all have my thanks and the reward of the certain knowledge of the improved results wrought of their input and advice.

Chapter 1

Introduction

Human trust is a complex and context sensitive interaction of risk, value, experience, expectation, uncertainty, social relationships, and individual human qualities [8, 34]. Humans operating in a physical world use a variety of interpersonal skills and social arrangements to create and evaluate trust in others. As such, we have a intuitive notion of what is meant by the word trust and live in a rich social web which sustains our ability to trust other people and organizations, and thus interact in a range of situations [8]. In a network-mediated virtual world, where interaction is represented by the exchange of messages between digital entities shown in Figure 1.1, trust is a fundamental challenge; identity is suspect, the exchange medium is suspect, and there is often no history of interaction or expectation of future interaction between particular entities [44]. Nevertheless, trust remains a requirement for maintenance of cooperative social groups and online economic activity [8]. In the virtual world of the online auction site eBay, buyers and sellers do not know each other, with 98.9% of seller-buyer pairs conducting less than five transactions over a five-month period [43, Section 4]. Yet, after being created in 1995, by 2002 eBay had 61.7 million registered users, 638 million listed items, and facilitated $14.9 billion dollars (US) in gross sales [19]. These results were supported by providing good-enough trust to an effectively anonymous community of ad-hoc buyers and sellers –

> "The key to eBay's success is trust. Trust between the buyers and sellers who make up the eBay community. And trust between the user and eBay, the company." – eBay Web Site [20]

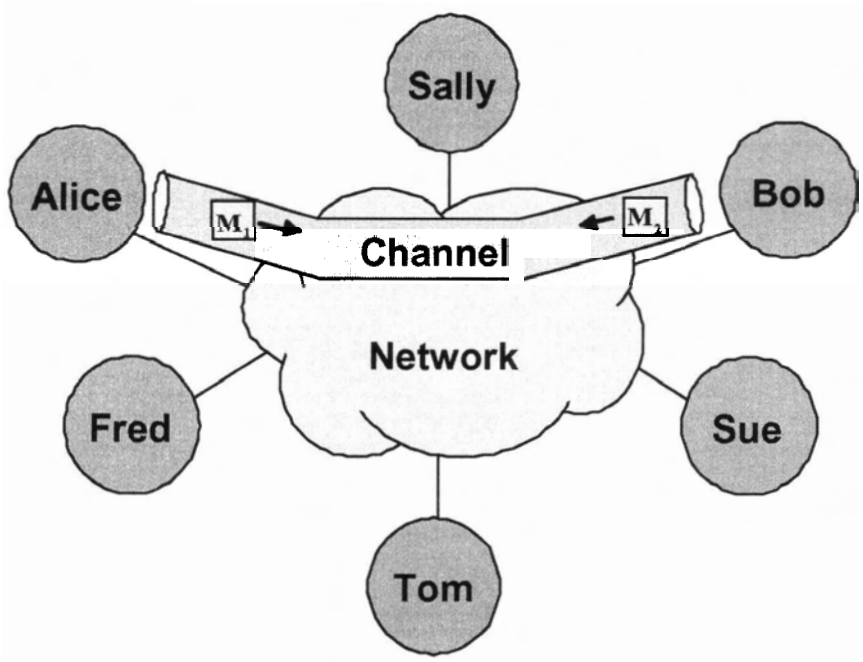


Figure 1.1. *Abstract Trust Context - distributed entities exchanging messages.*

Trust can be viewed as a form of expectation, your expectation that someone will do what they say and treat you fairly; for example, that your car will return from the mechanic in good running order and that you will only be charged for work and parts actually performed and installed. If trust is high, more uncertainty or risks will be accepted; for example, you will loan your car to your responsible daughter whom you have observed using good driving skills. If trust is low, additional controls and risk reduction strategies will be applied; e.g., you will accompany your daughter in the car, or you will limit her trips and passengers. The role of trust in human/social systems is to guide and support decisions where the outcomes depend on the good behaviour of others and where results cannot be completely controlled [8, 24, 34]. This lack of control represents the risk in the situation flowing from the dependence of the outcome on the actions of others – trust as " ...choosing to put ourselves in another's hands..." [34, p. 4]. There are many possible definitions of trust from a variety of fields of research, including the social sciences and philosophy [8, 34]. For this thesis, the following simplifying definition of trust has been

created:

Definition 1.0.1 Trust *is one's reasonable expectation of a positive outcome in a situation where there is less than full control over the actions of the participants.*

Computational trust (formal trust definitions or rules implementable in software) is an emerging field of research applying computational models to trust decisions associated with virtual environments [34]. For the rest of this thesis, the word trust should be interpreted as computational trust. Some solutions supporting aspects of trust in network-mediated interactions are available or emerging; for example, achieving identity authentication and message security through cryptography, evaluating past behaviour through reputations, and creating some expectation of future interaction through community membership [16, 24, 43]. Many of these solutions are limited or application-specific, and do not address the more general notion of trust [24]. Being able to validate the identity of another party and having a secure channel to communicate with that party does not mean they are not a crook, and online communities and reputations are often isolated, application-specific domains.

Another perspective from which to view requirements for and application of trust reasoning capabilities is *information privacy*: issues related to how various parties (including private individuals) decide with whom and how much information to share, and make these decisions using multiple information sources. Reputation in this context is a kind of credit or currency in an information-based economy. Being able to establish higher trust among a larger set of parties, with the help of mechanisms such as reputation, enables an organization to perform its function with greater scale and less friction. Generalized trust reasoning capabilities enables parties – individuals, corporations, human and software agents – to play fully empowered roles in a world of information-based, network-mediated relationships.

1.1 Objectives

The problem addressed by this thesis is a lack of generalized, decentralized, application-independent trust reasoning capabilities for use in ad-hoc, network-mediated environments.

The specific focus of this thesis is on trust mechanisms supporting ad-hoc interaction within a large pool of potential interactors, such as Internet users, where there exists relatively little direct knowledge nor formal relationships or explicit contracts, and without the normal face-to-face aspects of human trust [1]. To this end, this thesis contributes a new generalized computational trust model, the Generic Reliability Trust Model or GRTM, that is simple, flexible, and application and cryptography-framework independent. The proposed model, described in Section 3, is based on a novel application of the generalized probabilistic model of Network Reliability [12]. The seed of this generalization is also seen in the application-specific, confidence-based trust metric called the Maurer Confidence Valuation (MCV) [36]. The generalized nature of the new model is demonstrated with a new representation of the MCV metric and a new trust metric, Hop-count Limited Transitive Trust, which is faster and less complex than the MCV metric. Two exact algorithms are described for computing trust in the Generic Reliability Trust Model which, as with computing reliability, represents a #P-complete problem [12]. Thus, a conservative approximation useful for larger networks has also been created. Experimental results are presented from a Java software implementation and trust network data simulated using a scale-free network topology based random power law graphs [42]. This thesis does not address the question of suitability of the proposed model and metric from a philosophical, social science or economic perspective, nor is a specific application presented. Neither is the quality of the proposed metric compared to existing metrics due to lack of objective quality measures [9].

The remainder of this thesis is organized as follows. Chapter 2 provides a background on trust modelling, including an abstract trust model definition and survey of trust models. Chapter 3 contains the definition of the Generic Reliability Trust Model (GRTM) and associated trust metrics. The algorithms in Chapter 4 presents two general approaches and an approximation heuristic for computing trust metrics based on this model, and includes experimental performance results from a Java-based implementation. The proposed model and metric is compared to related work in Chapter 5. Finally, Chapter 6 concludes with a summary of the primary contributions of this thesis and suggestions for future research.

Chapter 2

Trust Modelling

This chapter presents the necessary background to support the presentation of the trust model proposed by this thesis and allow comparison with previous work. A *trust model* is a limited definition of trust and associated mechanisms used in trust-related decision processing of software-based and/or network-mediated interactions [24, 34]. The model is assumed to be limited in that it does not reflect all aspects of the richer notion of trust found in human interaction. Nonetheless, it is assumed that humans are the ultimate actors in situations requiring trust, for example through establishing policies or controlling the actions of software proxies, agents, or systems. The following sections continue with a description of an example application context and scenario, a definition of common aspects of trust models, a discussion of important concepts and issues related to authentication and security, and finally, a survey of existing trust models.

2.1 Application Context and Scenario: Can Alice trust Bob?

The assumed context of a trust model within this thesis is a distributed application involving digital entities exchanging messages as shown in Figure 1.1. The following application scenario, graphically depicted in Figure 2.1, provides additional context for many of the examples included in this thesis.

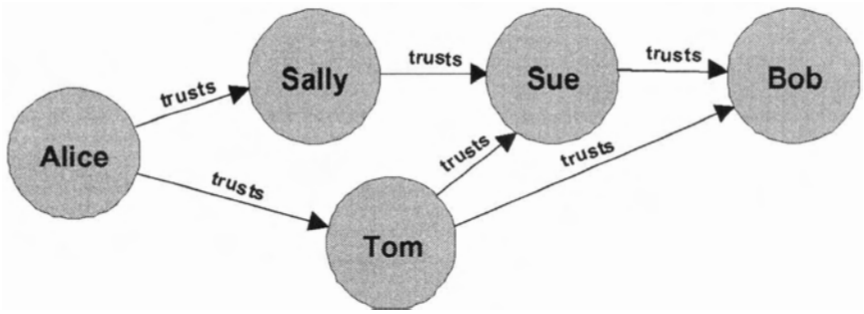


Figure 2.1. *Scenario: Can Alice trust Bob?*

> *Alice owns the rights to a set of movies. Alice delegates access granting power to Sally, who further delegates granting power to Sue. Similarity, Alice delegates authority to Tom who also delegates authority to Sue. Only Tom and Sue have direct knowledge of Bob. Bob purchases, from Sue, the right to play one of Alice's movies during a one week period.*

Instead of looking at the particular mechanisms of delegation or eCommerce transactions, the trust models in this thesis will focus on the evidence and associated relationships supporting Alice's trust decision. The key question to be answered is, *can Alice trust Bob, an entity previously unknown to Alice, when he submits his request to play this movie?*

2.2 Abstract Trust Model

This section presents a general framework for a trust model definition, an *abstract trust model*, by defining the critical aspects common to the trust models encountered in the research for this thesis. This general perspective will be applied in Section 2.4 to summarize a variety of concrete trust models.

2.2.1 Entities

The entities within a trust model are the objects which are directly represented as subjects of trust. Entities represent the sources, targets, and intermediaries – people, agents, software objects, or systems – which are the subjects of, participate in, or provide supporting information for trust decisions. More generally, an entity is a particular personality or actor which you or others have had some experience with in the past, are currently interacting with, and/or expect to interact with in the future. The role entities play within some trust model depends the definition of the model and on the situation. There are three general roles of interest. The *local* or *source entity* is the entity attempting to reason about its subjective trust of another entity, called the *remote* or *target entity*. The local entity may use information supplied by third parties referred to as *intermediate* or *recommender entities*. Alice, Bob, Sally, Sue, and Tom are the entities in scenario of Section 2.1. From Figure 2.2, Alice is the source, Bob is the target, and the others are recommenders. To say more than this, specifically, to say if or how much Alice trusts Bob, a particular trust model must define the necessary mechanisms to validate and measure the amount of trust represented by this set of entities and relationships.

2.2.2 Trust Value

A *trust value*, determined within any concrete trust model, is some measure or quantification assigned by a local entity to its belief in the trustworthiness of another entity. A few possible types of trust value include boolean, confidence, or discrete values such as {no-

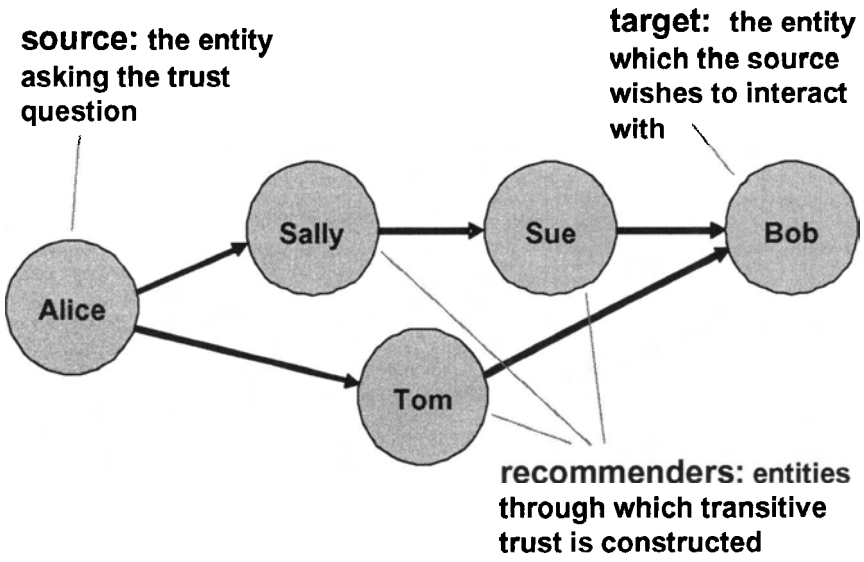


Figure 2.2. *Annotated Alice to Bob Graph*

trust, partial, complete}. The trust value will often indicate the expectation of a successful interaction, through which some desired outcome will be achieved. Table 2.2.2 from [34] provides one possible interpretation for ranges of confidence-based trust values. Marsh[34] provides an extensive discussion on the meaning of trust and definition of metrics. The actual thresholds depend on the sensitivity and requirements of a particular application and situation, and the subjective perspective of the authoring entity. For example, Alice might trust the authenticity of Sally's identity with "high" confidence and assign the confidence value of 0.8.

2.2.3 Subject-matter

Trust is subject-matter specific; that Sally trusts Tom with her lawn mower does not mean she also trusts him with her car. Another way of expressing this often implied aspect of trust is found in the following expansion from [11]:

"A trusts B" is shorthand for "A trusts B about X under certain conditions"

Table 2.1. *Marsh Trust Metric Confidence Value Interpretation*

Value range	Trust Meaning
+1	Blind
0.9	Very High
0.75 to 0.9	High
0.5 to 0.75	High medium
0.25 to 0.5	Low medium
0 to 0.25	Low

Here, the subject-matter of trust is represented by "X", and generally refers to the specific contexts or situations in which a trustor will trust a trustee; e.g. the borrowing of lawn mowers or cars. "Certain conditions" represents possible additional requirements of some trust model, such as how trust from certain authoritative sources might be required; for example, trusting a certain builder to perform a renovation on your house and also requiring they be licensed. Different trust models will vary as to whether and how much the subject-matter of trust can be specified. In [1], the term "trust categories" is used represent the subject-matter of trust.

2.2.4 Trust Roots

Trust roots, also called *seeds of trust*, are the assumptions represented in some trust model related to specific entities made by all entities in some community. These roots are the irreducible beliefs upon which the reasoning within a trust model is based. The label *authority* is often applied to an entity which is the subject of a trust root.

A practical issue for any trust system is how the trust roots are determined by some entity. This is often achieved by some set of explicit statements of trust in specific entities configured into a client system. For example, within the Internet Domain Name System (DNS), a distributed database system which translates names from a hierarchical name space into various resource records such as a network address. The name resolution process

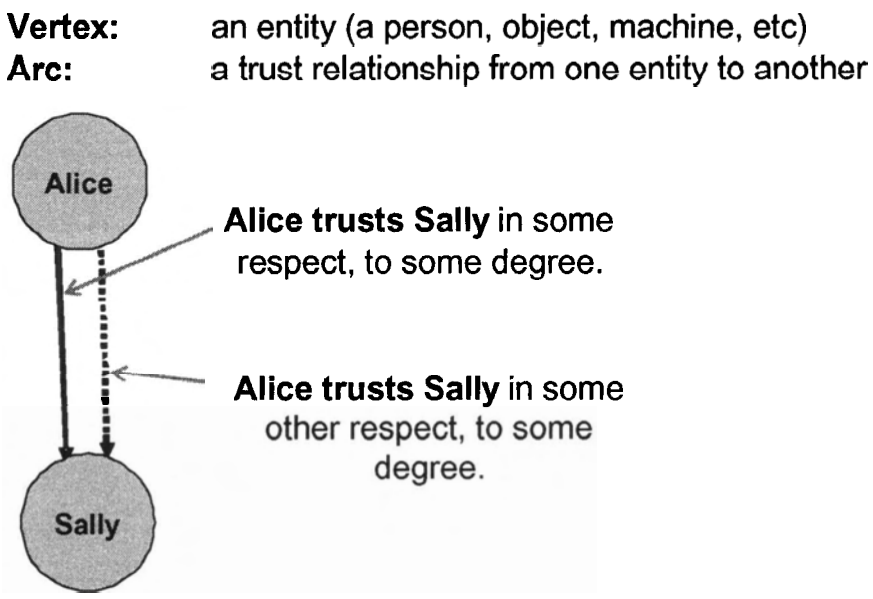


Figure 2.3. *Basic Trust Representation as a Graph*

starts with a query to a name server trusted by the client and which may then refer the client to another server [38]. The trust roots in this example are represented by the configuration of the network services of a computer system with a list of network addresses for (trusted) default name servers.

2.2.5 Direct Trust

Direct trust is some entity's independent belief in the trustworthiness of another entity. Direct trust, and trust in general, is not symmetric; that Sally trusts Bob does not imply that Bob trusts Sally. The direct trust beliefs of an entity will include trust roots, the difference being that the trust roots refer to the beliefs shared by an entire community of entities, whereas direct trust refers to the beliefs of a single entity. From Figure 2.3, Alice directly trusts Sally, meaning in this case that she is willing to interact with Sally regarding two subjects; e.g. receive e-mail, send a credit card number, or receive (and use) recommendation information about another entity.

2.2.6 Indirect Trust

Indirect trust is some entity's belief about the trustworthiness of another entity which is derived from the beliefs of other entities. A *recommendation*, a term often associated with indirect trust, is a statement of direct trust about a remote entity made by an intermediate entity. If Alice makes the statement "Alice trusts Sue", this is direct trust with respect to Alice, but indirect trust with respect to anyone else. The combination of trust roots, direct and indirect trust represents all the belief information, or *trust evidence*, available to a trust model. This distinction between direct and indirect trust is another way to express the *subjectivity of trust*, the property that given the same trust evidence, the trust decisions of one entity may be different from another.

2.2.7 Trust Metric

A *trust metric* in a trust model is a definition of a function that computes a trust value from a collection of trust evidence; it defines how a given local entity can utilize evidence to reach a conclusion about the trustworthiness of a remote entity [33]. As implied by [33], and unlike all the other models surveyed, the metric is a distinct element of some model and there may be multiple metrics defined with a given model. From this perspective, a trust model represents a family of trust metrics. The value type of a metric may be different than the trust value of the associated model. The evidence from which the trust metric will be computed will be related to a particular situation – generally some application-specific interaction context – which is be represented within the trust model. One approach to produce a generalized boolean trust metric is to test a particular trust metric value against some application specific policy; for example, evaluating whether a confidence-based metric result r is greater than some minimum threshold value Θ [33].

Indirect trust, and trust generally, is not always transitive; that Sally trusts Bob, and Bob trusts Tom, does not always imply that Sally trusts Tom. Notwithstanding this general lack of transitivity, the conditions under which transitivity is allowed are a defining aspect

of a trust metric and is an important mechanism for utilizing indirect trust. The transitivity rules of a trust metric will define if and how much trust can be established using indirect knowledge – trust through others. The term "conditional transitivity" has also been used to describe this aspect of trust [1]. Another general characteristic of indirect trust that can be observed is that given A trusts B, B trusts C and the transitivity rules of some metric implying A trusts C, A will tend to trust C less than how much B trusts C; that trust will usually decrease when it is derived using transitivity, and the decrease will often be directly related to some measure of the remoteness of the source.

The subjectivity of trust can be viewed as an appropriate localization of trust knowledge, maintaining under direct control those critical assumptions used to make security-related decisions and preventing *unintensional transitivity*, the ability of another entity to effect the trust assumptions of the local entity without its explicit consent [11]. This view of transitivity is also reflected in the legal maxim: *delegatus non potest delegare*, meaning that a delegate cannot appoint another[1]. Use of non-localized trust knowledge is often explicitly managed through use of recommendations [1], including some way to represent trust in the recommender and additional rules controlling how a local entity can make use of the recommendations. This use of recommendations addresses the concern of unintensional transitivity by forming explicit consent by a local entity to have its trust decisions influenced by recommender entities.

The subjectivity and scalability of a trust model are determined by the specification of a particular trust metric. If the metric does not support any transitivity and there are no trust roots in the model, then all trust will be strictly localized or personal, based solely on the local entity's beliefs. Given an objective of scalability, having a strictly local perspective from which to establish trust presents a significant limitation on the number of potential interactors. For example, in a dyadic interaction pattern where both parties require trust and there is no indirect trust or trust roots, the set of potential interactors will be reduced to a fragmented collection of small cliques or closed communities where all

[1]The Free Online Dictionary – URL http://legal-dictionary.thefreedictionary.com/

parties are symmetrically known and trusted by each other. It is for this reason that indirect trust utilizing the so-called "*weak ties*", bridge individuals through which smaller social clusters are connected, is so important in social networks. These weak times establish the connections with other social clusters and enable wide spread communication in networks of larger populations [25].

Additional characterization of a model and metric are possible by looking at the basis of the metric. Three general types of metrics are Chain-of-Proof (COP), Arithmetic, and Probabilistic. The *arithmetic type metric* combines evidence using simple arithmetic operations; for example, computing the average value of an assigned trust level. The *Chain of Proof or COP type metric* performs a boolean validity test using a chain of evidence: if each link in the chain is valid and correctly linked to the next, then the chain as a whole is valid. The *probabilistic type metric* incorporates some probability-based measure, such as risk or confidence, and combines multiple pieces of evidence in some probability-preserving operation.

2.3 Trust, Security and Related Concerns, Systems, and Mechanisms

This section clarifies the meaning of some concepts and terms often associated with trust. In particular, identity and reputation play important roles in practical trust mechanisms. As well, trust can often require associated properties of security. These concepts will be clarified here in the context of the trust perspective of this thesis.

2.3.1 Identity, and Reputation

An *identity*, or *ID*, is an label used to refer to an entity; for example, a person's name. The relationship between an identity and an entity is generally many-to-many, identity in a distributed system is an abstraction without necessarily any correspondence to actual enti-

ties; different identities do not necessarily represent distinct entities. For example, a person may have multiple login IDs on a computer system and some transaction processing systems perform load-balancing by sending requests for the same service to different physical machines. Because of the central role unique entities play in considerations of trust (see Section 2.2.1), additional mechanisms to restrict this relationship are required. *Identity trust* is the confidence in the one-to-one mapping between an entity and a identity; that an entity is who s/he says that s/he is and that the same identity used by multiple parties refers to the same entity. *Identity authentication* is the verification of the identity of an entity [24].

A *trust mechanism* is a defined trust model and trust metric, along with mechanisms for handling of trust evidence and application interfaces for obtaining trust metric results. A *trust system* is an operational trust mechanism including associated administrative mechanisms. Practical issues associated with any trust-related system or mechanism include managing trust roots and the method of identifying and authenticating entities.

Identity trust is fundamental in any trust system; unless you can confidently associate an identity to a given entity, there is no foundation on which to reason about trust using identity-specific information [18]. Information about an entity with the identity X is useless without some confidence that the subject entity is actually X. Is the entity that Alice is attempting to interact with really Bob, and is it the same Bob that Tom and Sue know? Is Bob really a dog?[2] Thus, the larger notion of trust is built upon identity trust and as a result, the terms authentication and *authentication systems*, a collection of security mechanisms which validate the purported identity of a given entity, are sometimes mistakenly used as synonyms for trust and trust system. Rather, an authentication system can be viewed as an example of a limited trust system focused on the limited subject-matter of identity authenticity.

A *reputation system* collects the interaction experience of many entities and provides

[2]This question refers to the famous 1993 New Yorker cartoon by Peter Steiner with the caption "On the Internet, nobody knows you're a dog." (page 61 of July 5, 1993 issue of The New Yorker, (Vol.69 (LXIX) no. 20) – available online at `http://www.unc.edu/depts/jomc/academics/dri/idog.html`

access to some metric representing a given entity's *reputation*, an objective measure of aggregate past behaviour. The goals of a reputation system are to provide information that allows trustworthy and non-trustworthy participants to be distinguished, encourages participants to be trustworthy, and discourages the participant of non-trustworthy participants [44, 43]. A reputation as defined here is not subjective like trust; if two entities query the same reputation system at the same time about the same subject, they should receive the same result. Reputation systems are a possible source of information for use in making a trust decision. As entity activity is reported using some identity, as in a general trust system, identity trust is also fundamental to operation of a reputation system. For example, buyers use the eBay Feedback system as an aid to decide whether to proceed with a eCommerce transaction with some seller by accessing the feedback rating associated with the seller's eBay identity [44, 43]. An example of a violation of identity trust in this case is where control of an identity with a good reputation is moved from the entity which participated in successful transactions to some other, possibly untrustworthy, entity. After this transfer, potential buyers making a purchase decision based on the reputation associated with this identity may be exposed to a risk of loss higher than they expect.

Another example of the problem associated with the lack of identity trust, in the context of distributed systems reliability, is the assumption of some limited proportion of faulty entities within a system. The *Byzantine General's Problem* is a model of components in a reliable distributed computer system where the distributed components represent generals who can only communicate via a messenger and faulty components represent traitors. The problem here is how the loyal generals (non-faulty components) reach some common plan of action, and to prevent the traitors from causing the loyal generals to adopt a bad plan. No solution exists to this problem unless more than two-thirds of the generals are loyal; in other words, it is required that there be at least $3m + 1$ distinct nodes (generals) to cope with m faulty nodes (traitors) [6, 32]. In a *Sybil Attack*, an attacker uses multiple identities to impersonate distinct entities and in so doing undermine any assumed mapping between identity and entity and hence the actual number of distinct entities. This may allow

the attacker to violate any assumed proportion of faulty nodes, and thus compromise the reliability of such a distributed system [18].

2.3.2 Security

Security-related mechanisms are utilized by many trust models to be surveyed later in Section 2.4. A brief overview of security and cryptography concepts and their relationship to trust is presented in this section. The key point to make about the relationship between trust and security, can be summarized as:

$$\text{Trust} \neq \text{Security}.$$

Security in computer information systems is the expression of some combination of the properties of availability, confidentiality, non-repudiation, and integrity of data and the authenticity and authorization of users [14, 16, 39]. A *security model* for a distributed system looks at how to secure the processes representing the interacting entities, the channels used for interaction, and the information objects which are the subject of interaction [14]. *Authorization* is the assignment of rights to an entity to perform specific actions with some constraints [35, 24]. *Cryptography* is the study of mathematical systems for securing data and in particular creating messages characterized by some combination of being *confidential*, meaning no unauthorized extraction of information, signed, and unmodified [16, 40]. Cryptography offers mechanisms supporting security and these properties of security are important foundations for trust systems, especially in the context of untrusted public networks such as the Internet.

Informally, a *cryptographic key*, or *key*, is a relatively small piece of data, for example a few hundred bits, used by a cryptographic algorithm to perform encryption and decryption on subject data, also called plain-text; without the correct key it should be computationally impractical to determine the original value of data encrypted with that key. Some form of cryptographic key is the secret shared among entities used to create a secure communications channel. *Public key cryptography* consists of asymmetric cryptographic techniques

involving a pair of keys: a *public key* accessible by anyone, and a *private key* known only by the entity associated with the key-pair. In such a system, a message can be encrypted using a public key in such a way that it can only be practically decrypted with the matching private key, and the private key cannot be practically computed from the public key [16, 40, 15]. Conceptually, a *digital signature* S_M associated with a message M is a digest of M processed with the private key $K_{private}$ in such a way that anyone with the matching public key K_{public} and the signed document, $M + S_M$, can validate that the signature was created using the associated private key $K_{private}$ and message M has not been modified. Thus, a signature is proof of both the authenticity of M and that the holder of key $K_{private}$ was the source of M. *Public key certificates* or *Digital certificates*, such as the X.509 certificate, are signed documents containing standardized information items which bind a public key these item values, one of which is generally the identity of the subject entity controlling the associated private key[3]. If Sally signs Bob's public key identity certificate, Sally is asserting that she believes the contained public key really is Bob's. A *certificate chain* is a set of certificates $C_1, C_2, \ldots, C_k$, where the subject entity of C_i is the signing entity of C_{i+1} [36, 40, 28]. An example is a chain composed of Sally's certificate signed by Sue and Bob's certificate signed by Sally.

Public key cryptography has a number of applications related to message security and identity authentication operations [16, 40]. Any user can create a confidential message for entity B by encrypting it using B's public key. Message contents and sender can be authenticated using a signature created by the sender, which if saved also serves for non-repudiation. A user can be authenticated requiring them to sign a randomly generated value with their private key, which is then validated using their public key. Creating a secure

[3]Generally, certificates contain more information than the public key value and subject identity, such as the identity of the signing entity, an expiry date for the certificate, and the amount of effort that went into authenticating the identity of the user. In the case of a CA, this is description of the authentication effort is called a certificate practice statement or CPS. Certificates can be used to represent more than key-identity bindings; for example, they can contain a specification of rights granted to the subject entity by the signing entity.

channel between entity-pairs without a previous security relationship – i.e. without an existing shared secret – can be achieved through a variety of key exchange or establishment protocols. An outline of one possible protocol is as follows:

1. A is in possession of B's public key.
2. A creates a symmetric or session key K, encrypts it using B's public key, and sends the encrypted key to B.
3. B decrypts the message containing K using his private key.
4. A and B exchange messages using key K.

A common example of this last application of public key cryptography is the establishment of a secure channel between a web browser (client) and a web site (server). The presence of the lock symbol in a user's browser window indicates that a secure channel has been established with the associated web site using a symmetric key establishment protocol such as Transport Layer Security (TLS), which is based on the Secure Socket Layer protocol (SSL) [15]. Without public key cryptography, some other secure communication channel would be required to establish the shared secret; for example, Alice might call Bob on the telephone and they could verbally agree on a symmetric key. This communication channel perspective offers an alternate definition of trust:

> "Trust is that which is essential to a [establish a secure] communication channel but cannot be transferred from a source to a destination using that channel." [23]

When using cryptography, the security of keys is the basis for secure communications; how an entity obtains and validates the public key of another entity is a critical challenge in applying public key techniques [36, 40]. Suppose Alice found Bob's public key on some web site. Without a way of authenticating that this is really Bob's public key, this key is useless. Alice cannot know whether some other entity, say Ralph, has supplied their own public key and associated it with Bob's identity – this is known as the man-in-the-middle attack. In this attack, Ralph could decrypt messages intended for Bob, and

possibly invisibly perform other operations that invalidate the security of the messages sent between Alice and Bob. If Alice instead obtains a public key certificate for Bob, from any source, which is signed by someone Alice recognizes (i.e. she knows their public key) and furthermore she trusts them to perform certificate authentication, then she should be able to use the contained public key for secure communication with Bob [36]. Some of the trust models surveyed in Section 2.4 directly address this problem.

Cryptography generally provides mechanisms to provide security, not trust. While security and the authenticity of identity are necessary conditions for trust, they are not sufficient to establish trust as defined in this thesis; being able to authenticate someone's identity does not mean they are trustworthy, and being able to communicate securely with some eCommerce web site does not mean that the operators of that business are competent. From this perspective, the abstract trust model of Section 2.2 identifies the particular concerns, beyond security, to be addressed by a trust system.

2.4 Survey of Existing Trust Models/Metrics

The following is a survey of existing trust models. The selected models present a range of abstract trust model structure and underlying theoretical trust metric, with at least one model-metric pair from each of the metric types described in Section 2.2.7. Each trust model is considered in enough detail to expose all of the elements of the abstract trust model introduced in Section 2.2.

2.4.1 X.509 PKI

A *public key infrastructure*, or *PKI*, is a system for creating and accessing public keys and validating the association between a public key and an identity [40]. The purpose of a PKI system is to ensure that a given public key is correctly associated with the subject entity who owns the associated private key [28, Section 3.1]. X.509 PKI is a particular instance of a PKI standardized by the PKIX working group of the Internet Engineering Task Force (IETF or www.ietf.org). It utilizes a centralized model where a small set of certificate authorities, or CAs, sign public key certificates using the X.509 identity certificate standard and containing the binding between a public key and a subject identity [27, 28]. A summary is provided in Table 2.2.

All users of the X.509 PKI system must configure a small set of CAs as trust roots through which to validate all certificates; any certificate signed by such a root CA will be deemed to be authentic if the signature can be validated using the associated public key of the trust root. This centralized structure was created to ensure a restricted administrative structure to prevent errors and promote ease of use for unsophisticated end users [28]. Note: Web browsers such as Microsoft's Internet Explorer include a set of root certificates for CAs from organizations such as VeriSign, enTrust, and AOL Time Warner. Alice will accept Bob's public key as valid if she can construct a chain of CA certificates, starting with a CA that she has directly authorized and ending with the CA who signed Bob's certificate. "Alice can use [a] chain of certificates if and only if she trusts every entity in the chain

between her and Bob" [36, Section 2.2]. If a CA is compromised, then any certificates validated using the signature of that CA may not be valid. For example, this potential arose in January 2001 when VeriSign (a CA) issued two certificates to an individual fraudulently claiming to be an employee of Microsoft [10]. The complete set of CAs trusted by Alice will form a hierarchy, with the root CAs at the base. One way to model Alice and Bob scenario described in Section 2.1 using a PKI is shown in Figure 2.4. Here, the intermediate entities Sally, Sue, and Tom are represented as CAs.

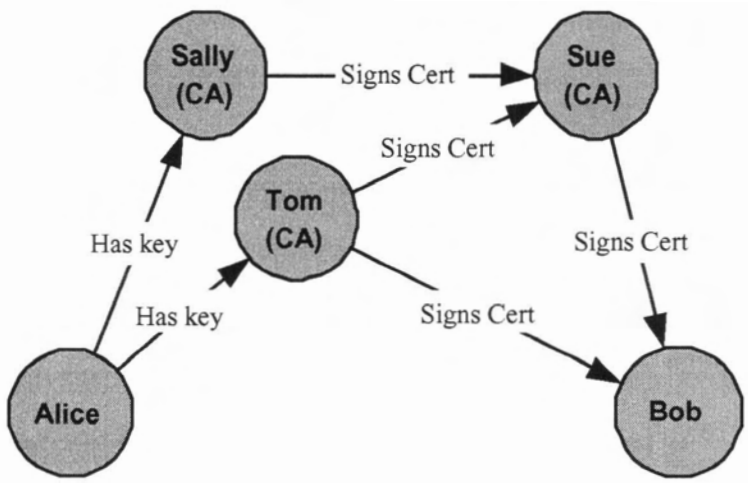


Figure 2.4. *Alice-Bob example using PKI*

Public key authenticity is the subject-matter of a public key infrastructure when viewed as a trust model. Any trust derived through this system only relates to the authenticity of the key-identity binding; it does not necessarily mean you can trust the entity behind a given identity to perform a particular action [21]. The configuration of a root or trusted CA in your computer effectively means three things:

1. you assume that the public key associated with the CA's identity is valid,
2. you trust that organization controlling that CA's private key to correctly authenticate public key-identity associations (via a PKI certificate), and
3. you similarly trust any CA to which you can establish trust via a certificate chain.

Table 2.2. *Trust Model Summary for PKI*

Aspect	Description
roots	root CAs, configured into operating system or Internet browser application
entities	certificates binding a public key to an entity with a given identity
subject-matter	public key validity
direct	digital certificate represents the signing entity's recommendation of the validity of the contained public key and its association with the contained identity.
indirect	a chain of certificates, each signed by a CA
metric	binary: a certificate or certificate chain is valid and not expired, or not valid (cannot validate)
implementations	commercial authentication and certificate services, built in to a variety of applications including email clients and browsers, and is the basis for secure socket layer (SSL) validation

2.4.2 PGP

The Pretty Good Privacy system, or PGP, is an open-source public key cryptography framework created by Phil Zimmermann in 1991. OpenPGP is the name of an IETF standard created in 1997 to define PGP independent of its (first) implementation, and is also the name of an alliance of companies, *openpgp.org*, offering OpenPGP-based products and services [7]. These systems use a combination of public key and symmetric cryptography to provide security services for electronic communications and data storage. For the purpose of this thesis, OpenPGP and PGP are synonymous. A summary of the trust model of PGP is in Table 2.3

Similar to X.509 PKI, identity-key bindings are authenticated by being signed by some entity trusted to perform this function. Unlike PKI, any user can sign a public key certificate, PGP does not restrict this function to a predefined set of centralized certificate authorities. PGP uses a decentralized system of trusted introducers, who operate in a similar role to a CA – hence, PGP's trust model is sometimes referred to as a web of trust. When Alice signs Bob's certificate, she is introducing Bob's key to anyone who trusts Alice as an introducer [48].

Indirect trust is a consideration within PGP when a subject entity cannot directly verify the authenticity of a public key and must instead rely on the judgment of other users [41, Section "Managing Keys"]. In this case, as is the case with X.509 PKI, the question of trust is focused on the narrow subject-matter of authenticity of key-identity binding; the decision to accept another entity's claim of the authenticity of a public key-identity binding not in your local trust knowledge base. The trust knowledge base in PGP is a set of certificates, with associated trust levels, called a key-ring and stored on a user's machine. There are no trust roots in PGP, unlike the centralized CA role in X.509 PKI; each user must independently evaluate and configure the entities that can act as recommenders. While there are centralized *key-servers* for PGP, systems offering public access to the public-keys of multiple entities, they are simply repositories, they do not act as signing authorities and keys obtained cannot be assumed to be authentic.

There are two types of trust that can be expressed through PGP; trust in the authenticity of a key, and trust in a key holder to act as an introducer. For the former, the Signature Type indicates the amount of verification the signer has performed on a key, a measure which is conceptually similar to the certificate practice statement of a PKI CA [7, Section 5.2.1]. For the latter, the concept of a trust level is used, where level n trust asserts that a key is trusted to issue level $n - 1$ trust signatures [7, Section 5.2.3.12]. Thus, a *meta-introducer* can recommend (by signing) another key at the lower *introducer* level. A key is trusted as authentic when it is signed by the local key (i.e. direct knowledge), signed by one completely trusted key, or signed by k marginally trusted keys, where k is two by default, but can be set higher by individual users. Using the Alice and Bob scenario of Section 2.1, since Alice does not have any direct knowledge of Bob, she needs to evaluate her transitive trust in Sally and Tom to act as an introducer. A possible representation of the PGP trust relationships in this scenario is in Figure 2.5.

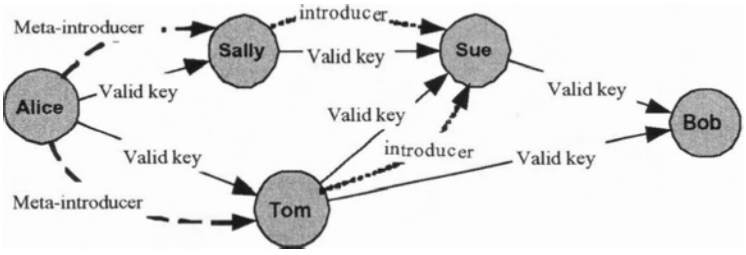


Figure 2.5. *Alice-Bob example using PGP*

Table 2.3. *Trust Model Summary for PGP*

Aspect	Description
roots	Trust Packet datasets (key-rings) specifying which key holders are trustworthy introducers.
entities	key holders, represented by keys.
value	an entity, represented by its key, is assigned a trust level from {unknown,untrusted,marginal,complete,implicit}, where *implicit* is the highest and applied to the local entity's keys. As well, keys have a validity level of {invalid,marginally valid,valid}.
subject-matter	key authentication
direct	as trust level assignments on keys in a local key-ring: direct knowledge, introducer, meta-introducer
indirect	limited to a maximum of two intermediate entities; meta-introducer and introducer
metric	a key is valid if it is signed by the local entity or a suitable number of trusted keys
implementations	commercial and open-source products available for authentication and file encryption services, plug-ins available for email clients

2.4.3 Trust Management

Blaze, et.al. [35] define *Trust Management* as a unified framework for evaluating security policies and credentials to make decisions regarding the authentication and authorization of entities to perform certain operations or access certain resources. A summary of the KeyNote implementation of this model is provided in Table 2.4. Using a trust management approach, access to certain operations or resources is controlled by issuing authorized entities a certificate, also called a *credential assertion*, specifying the authorization, and creating application-specific security policies specifying which certificates can perform controlled operations or authorize other certificates (delegation). An application controlling a certain operation or resource presents to a trust management system some requesting entity's certificate and the application's local security policies. The trust management system evaluates the certificate and security policies and returns an "allow" or "disallow" to the application. This approach has also been called *decentralized trust management* since the point where the control is implemented is separate from where the authorization is evaluated. The KeyNote system [4], which superseded the PolicyMaker system [4], has a simple application-independent language for specifying security policies and is an implementation of a trust management system.

Table 2.4. *Trust Model Summary for KeyNote*

Aspect	Description
roots	local policy assertions
entities	called principles; perform or authorize other entities to perform controlled actions.
subject-matter	access control based on security policies.
direct	security policy statement about a particular certificate.
indirect	delegation of authorization.
metric	programmable: e.g. binary {'reject','approve'} or discrete {'none','restricted','full'}
implementations	KeyNote system has been used in a variety of access control applications such as digital rights management and network control.

Table 2.5. *Trust Model Summary for Distributed Trust*

Aspect	Description
roots	trust statements in local database
entities	generalized software agents
values	integers; from -1 for distrust to 4 for complete trust
subject-matter	trust category of trust statements
direct	statements of direct trust and recommendation trust
indirect	chain of reputation statements created from localized trust knowledge and communicated via a recommendation protocol
metric	average of all recommendation values, trust of a recommendation is the product of the normalized recommendation trust (rec_trust_value/4) and the reputation making up the recommendation chain
implementations	none known, subsequent proposals for virtual communities

2.4.4 Distributed Trust

In [1], Abdul-Rahman and Hailes propose a distributed trust model using recommendations. A summary is provided in Table 2.5. It is termed distributed as it provides a generalized model for sharing and reasoning about trust knowledge between distributed entities. Each entity has a database of trust statements specifying the subjective trust value of remote entities and recommendations from remote entities. Trust statements are subject-matter specific; a "trust category" in the statement specifies the aspect of trust. Trust in remote entities, entities which do not exist in the local database, can be derived using a recommendation protocol to obtain reputation information from remote entities. This model also includes support for searching for relevant trust recommendations and performance improvements through caching of recommendations.

2.4.5 Network Flow Trust Metric

In [33], Levien and Aiken define the *Network Flow Trust Metric* to determine the authenticity of a public-key certificate using a set of digitally signed public key certificates. The trust value computed by the metric indicates the presence of a sufficient number of independent certificate sources, where sufficiency is determined by some application. Associated with this metric is the concept of the *attack resistance* of a trust metric and associated PKI system, a measure of the number of keys that must be compromised before the system will accept an invalid name/key binding [33]. The following is a description of the network flow model and its application by the trust metric. A summary is provided in Table 2.6.

From [5], a network is defined in terms of a digraph $G = (V(G), E(G))$, a source vertex $s \in V(G)$ and a sink vertex $t \in V(G)$, $s \neq t$, and a nonnegative *capacity function* $c(a)$, $a \in E(G)$. If f is a real-valued function defined on $E(G)$, and if $K \subseteq E(G)$, denote $\sum_{a \in K} f(a)$ by $f(K)$. Furthermore, if K is a set of arcs of the form $(S, \overline{S}) = \{(u, v) : u \in S \text{ and } v \in \overline{S}\}$, write $f^+(S)$ for $f(S, \overline{S})$ called the flow *out of* S, and $f^-(S)$ for $f(\overline{S}, S)$ called the flow *into* S. A *flow* in a network is a nonnegative function $f(a)$ defined on $E(G)$ such that

$$\text{for all arcs a, } 0 \leq f(a) \leq c(a) \text{ , called the } \textit{capacity constraint}, \text{ and} \tag{2.1}$$

$$\text{for all } v \notin \{s, t\}, f^+(v) = f^-(v) \text{ , called the } \textit{conservation condition}. \tag{2.2}$$

The value of a flow is the *resultant flow out of* s, which is $f^+(s) - f^-(s)$, or the *resultant flow into* t, $f^-(t) - f^+(t)$. A flow f is a maximal flow, or *maxflow*, if there is no flow f' such that $f' > f$.

In the Network Flow trust metric of [33], also called the *unit capacity maxflow metric*, a set of public key certificates are represented by the digraph G as follows. Key a is represented as a vertex v_a in $V(G)$. A certificate containing key a signed by key b is represented as the arc (v_a, v_b) in $E(G)$. Let $dist(s, t)$ be the length of the shortest path in the digraph G from vertex s to vertex t. The general approach of this metric is to define node capacity functions on $V(G)$ which place lower values on vertices that are farther away

from both s and t, based on the value of $dist()$.

Let $succ(s)$ be the *successor set* of s, the set of vertex v where there exists an arc of the form $(s, v) \in E(G)$. Let d be the minimum of number of signatures required to authenticate a certificate. The following are example functions from [33] used together to determine the capacity of a vertex.

$$f_s(v) = \begin{cases} max(\frac{1}{d}, \frac{1}{|succ(s)|}) & \text{if } dist(s,v) = 1 \\ \frac{1}{d} & \text{if } dist(s,v) \geq 2 \end{cases},$$

$$g_t(v) = \frac{1}{d}.$$

Using these functions, the node capacity function C for a given s, t and assuming a in-degree d for every vertex in $V(G)$, is as follows:

$$C_{s,t}(v) = max(\, f_s(v),\, g_t(v)\,).$$

This capacity function is defined to result in a maxflow value of one should all relevant vertices in G have the required in-degree d, hence the "unit capacity" portion of the metric's name. Any target vertex with an in-degree $\leq d$ will have a maxflow value less than one. Thus, the assumed in-degree d acts as the parameter which determines the minimum required evidence for a source to trust the authenticity of an indirect target. An application determines the required in-degree d and calculates C as above. The maxflow value from s to t in G indicates whether the set of certificates represented by G are sufficient evidence of the authenticity of t. Generally, this maxflow value is used in a threshold test: if maxflow from s to t is $\geq \theta$, then s will trust that key t is authentic.

Table 2.6. *Trust Model Summary for the Network Flow Trust Metric*

Aspect	Description
roots	subjective
entities	public-keys
subject-matter	public key authentication
direct	a signed certificate
indirect	a certificate chain
metric	maximal network flow with a specialized capacity function

2.4.6 Bayesian Network

The *Bayesian Network-Based Trust Model*, denoted here as BNTM, created by Wang and Vassileva [46] uses probabilistic techniques to model and share different aspects of trust. A summary of this model is in Table 2.7. The general notion of trust in BNTM is the competence to provide some service. The model is described in the context of a client selecting a server, called the *file provider*, from which to download a file. File requestors and providers are part of a *peer-to-peer*, or *P2P*, system, meaning individual entities, or peers, can act as both client and server, and there is no centralized server or trusted authority to indicate the best provider to serve a given client request. A Bayesian network and its application in this trust model are described below, as well as an outline of how recommendations are used and trust is computed.

From [29], a *causal network* is a directed acyclic graph $G = (V(G), E(G))$, where each vertex $v \in V(G)$ has finite number of mutually exclusive states and each arc $a \in E(G)$ of the form (u, v) represents a *causal relationship* from u to v, meaning an increase in the certainty of u has the effect of increasing the certainty of v. For a vertex v with entering arcs of the form $(u_1, v), \ldots, (u_k, v)$, the set of vertices $\{u_1, \ldots, u_k\}$ are the *parents* of v. The vertices of a causal network can represent different aspects of trust. In the case of the BNTM, four vertices are used to represent a provider:

1. the provider's competence (T) in providing files with states {*"satisfying", "unsatisfying"*},
2. download speed (DS) with states {*"Fast", "Medium", "Slow"*} ,
3. file quality (FQ) with states {*"High", "Medium", "Low"*}, and
4. files types (FT) offered with states {*"Music", "Movie", "Document", "Image", "Software"*}.

Using a causal network, the BNTM represents the general concept of trust as the probability that the transaction will be satisfactory $P(T = \text{"}satisfying\text{"})$. The causal network represents the relationship between trust and three more specific aspects of the transaction:

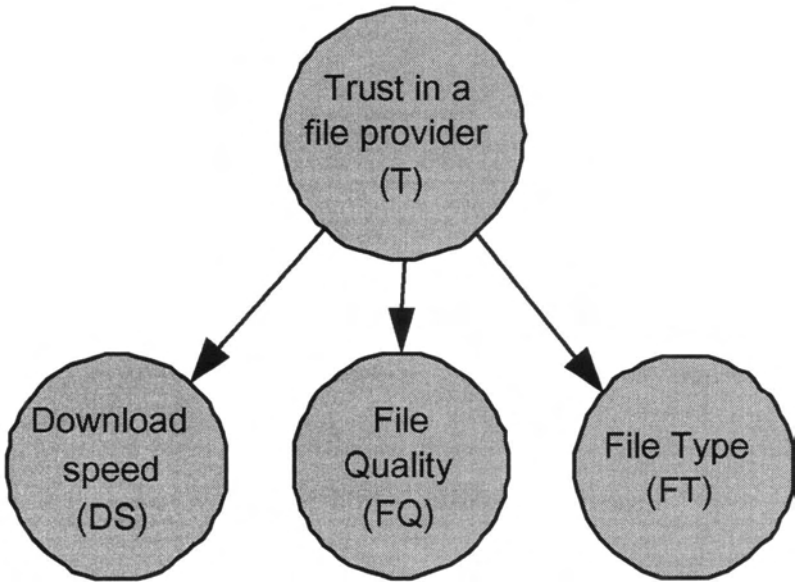


Figure 2.6. *Causal Network for Trust in a File Provider*

the speed, quality, and types of files provided. This is shown in Figure 2.6. Continuing from [29], if the vertex A has the states $a_1, \ldots, a_k$, then $P(A)$ is a probability distribution over these states:

$$P(A) = (x_1, \ldots, x_j)\ , x_k \geq 0\ , \sum_{i=1}^{k} x_i = 1, \tag{2.3}$$

where x_i is the probability of A being in state a_i. Let $P(A, B)$ be the probability of the joint event $A \wedge B$. The *conditional probability statement* "given the event B, the probability of the event A is x" is represented as $P(A|B) = x$ and the *fundamental rule* is:

$$P(A|B)P(B) = P(A, B)\,. \tag{2.4}$$

Bayes' rule and Bayes' rule conditioned by C are:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}, \tag{2.5}$$

$$P(B|A, C) = \frac{P(A|B, C)P(B, C)}{P(A|C)} \tag{2.6}$$

If A is a vertex with the states $a_1, \ldots, a_n$ and B is a vertex with the states $b_1, \ldots, b_m$, a *conditional probability table* or *CPT* representing $P(A|B)$ is an $n \times m$ table, with cell

(i, j) containing the number $P(a_i|b_j)$, and is denoted here as CPT_A. A *Bayesian network* or *BN* is a causal network $G = (V(G), E(G))$, where for each vertex $u \in V(G)$ there is an associated conditional probability table $P(u|v_1, \ldots, v_k)$, where $\{v_1, \ldots, v_k\}$ are the parents of u. The *joint probability distribution* $P(U)$, where U is a subset of $V(G)$, can be calculated using *BN chain rule* [29]: $P(U) = \prod_{v \in U} CPT_v$. No further details of inference calculation using a Bayesian network are covered in this thesis.

In the BNTM, there is a Bayesian network maintained by every peer for every file provider of interest to that peer. Each BN is composed of the four vertices $\{T, FT, FQ, DS\}$, shown in Figure 2.6. Each peer in the network also maintains counts of file downloads for each provider p of interest: total files downloaded T_p, and subtotals for the states of the BN. Let the total files counted for given vertex v representing an aspect of provider p be denoted as $v_{p,state}$; for example, the total music downloads are denoted $FT_{p,Music}$. In addition, the subtotal number of downloads for each aspect which are also $T = \text{“}satisfying\text{”}$ is denoted as $T_{p,satisfying,aspect=\text{“}state\text{”}}$; for example, the total count of music downloads which were satisfying is $T_{p,satisfying,FT=\text{“}Music\text{”}}$. The subscript p will be omitted if obvious in context. The conditional probability table for each vertex of every BN in every client is calculated using Bayes' rule and, in the example application, the file download counters associated with a given provider. For example, the value $P(FT = \text{“}Music\text{”}|T = \text{“}satisfying\text{”})$ for a given provider is the probability that the file downloaded is a music file, given the download is satisfying. It is computed as follows:

$$P(FT = \text{“}Music\text{”}|T = \text{“}satisfying\text{”}) = \frac{P(FT = \text{“}Music\text{”}, T = \text{“}satisfying\text{”})}{P(T = \text{“}satisfying\text{”})} = \frac{T_{satisfying,FT=\text{“}Music\text{”}}}{total} \Big/ \frac{T_{satisfying}}{total}.$$

Recommendations in [46] are one peer's answer to another peer's query of the competence of a certain file provider. The answer to a query such as $P(T = \text{“}satisfying\text{”}|FT = \text{“}Music\text{”}, FQ = \text{“}High\text{”})$ is calculated from the recommending peer's BN for the given provider. All recommendations for a given provider received by a peer are combined after being weighted by a *recommendation trust value* representing that peer's trust in the rec-

ommending peer to act as a reference. After a peer interacts with a provider, it updates its file counters and recalculates the CPTs in the associated BN. It also updates the recommendation trust values associated with all peers that provided a recommendation for this provider; the values are adjusted upward or downward based whether the recommendation was below or above the resulting interaction satisfaction. Another form of learning associated with the reputation of peers and providers is *gossiping*; here, agents exchange and compare their Bayesian networks for a given file provider.

Representing the aspects of trust in a Bayesian network allows a peer to infer additional aspects of trust in file providers when they do not have prior interaction results related specifically to that aspect. For example, the probability that a file provider is trustworthy in providing music files with high quality can be calculated using a BN as $P(T = \text{“}satisfying\text{”}|FT = \text{“}Music\text{”}, FQ = \text{“}High\text{”})$. The construction of the Bayesian network requires repeated interactions; hence, it is applicable to commerce interaction within, for example, a group of specialized collectors, where there are likely to be repeated interactions, but would not be applicable to general buyers and sellers on eBay where repeated interactions are rare [43, 46]. The authors of [46] simulated the effect of this application of BN. Their results indicate an increase in the percentage of satisfying interactions when a BN is used.

Table 2.7. *Trust Model Summary for the Bayesian Network-Based Trust Model*

Aspect	Description
roots	subjective
entities	peers in a P2P network
subject-matter	aspects related to competence
direct	file download counters and recommendation trust value
indirect	recommendations computed from peer's BN
metric	if directly known, use compute probability of satisfying using BN, otherwise, compute weighted sum of recommendations

2.4.7 Maurer Confidence Valuation

The *Maurer Confidence Valuation* (MCV) trust metric and associated trust model are described in [36, 31] and summarized in Table 2.9. This model was created as a tool to evaluate PKI certificate schemes by reasoning about the authenticity and trustworthiness of entities from the perspective of some entity using a set of certificates and recommendations authored by multiple entities. The statements defined in Table 2.8, also called evidence, in the model represent assertions made by entities regarding direct trust in the authenticity of a public key or indirect trust in other entities with respect to their belief in the authenticity of a public key. Maurer's confidence-based trust metric uses probabilistic analysis of evidence supporting the desired conclusion that a source entity believes some target entity's key is authentic, or simply "Can Alice authenticate Bob's public key?".

Table 2.8. *Meaning of Statements in the MCV Trust Model*

Statement Type	Meaning
$Aut_{A,X}$	A believes X's public key is authentic.
$Trust_{A,X,i}$	A believe's X is trustworthy for issuing certificates and recommendations at level $i-1$.
$Cert_{X,Y}$	a certificate signed by X containing Y's key.
$Rec_{X,Y,i}$	a recommendation signed by X for Y.

The set of statements I which are used as input to the model are called the *initial statements*. Each statement $S \in I$ has an associated probability p_S, representing the authoring entity's confidence in the statement. It is assumed that the events represented by these statements are independent. There are no assumed trust roots within the Maurer trust model; all statements represent direct trust with respect to some entity. Any subjective trust beyond that directly asserted by a given entity, must be derived from initial statements using the transitivity rules of the MCV model. Reasoning about indirect trust is based on deriving new statements using the following rules:

Definition 2.4.1 *For source entity A and entities X, Y,*

for level i=0:

$$\textit{if } Aut_{A,X} \textit{ and } Trust_{A,X,1} \textit{ and } Cert_{X,Y} \textit{ then } Aut_{A,Y} \textit{ is a derived statement.} \quad (2.7)$$

for level $i > 0$:

$$\textit{if } Aut_{A,X} \textit{ and } Trust_{A,X,i+1} \textit{ and } Rec_{X,Y,i} \textit{ then } Trust_{A,Y,i} \textit{ is a derived statement.} \quad (2.8)$$

Let V be a subset of initial statements, $V \subseteq I$. Let $closure(V)$ denote the reflexive, transitive closure of V with respect to Definition 2.4.1.

The confidence of a statement S, denoted $conf(S)$, is the probability that S can be derived from the initial view I, determined as follows:.

$$\begin{aligned} conf(S, I) &= P(S \in closure(I)) \\ &= \sum_{\substack{V \subseteq I: \\ S \in closure(V)}} \prod_{T \in V} p_T \cdot \prod_{T \notin V} (1 - p_T). \end{aligned}$$

Given some set of initial statements I, trust between entities s and t is the value $conf(Aut_{s,t}, I)$, representing s's confidence in the authenticity of t's public key.

A minimal support set for the statement S, called a *minimal subset*, is a subset of initial statements where if, any statement is removed it is not possible to derive S. The algorithm proposed for computing $conf(S)$ performs inclusion-exclusion on the set of all minimal subsets from which the statement S can be derived.

The MCV metric and underlying probabilistic model is a precursor to the generalized trust model of Section 3 which is at the core of this thesis. When Maurer presented his model in [36], he did not present it as a generalized reliability problem and he used very different notation. The notation and form of the definition used here have been changed in order to have a more unified notation and to more easily show how it fits into the proposed generalized reliability framework to be presented in Section 3. Within the generalized model, the definition of $conf(S)$ will be represented as a reliability problem defined by $closure(V)$, where V is a subset of initial statements.

Table 2.9. *Trust Model Summary for Maurer Confidence Valuation*

Aspect	Description
roots	initial statements
entities	certificates and recommendations
subject-matter	public key authentication
direct	statements of authentication (level=0) and statements of recommendation trust (level $>$ 0)
indirect	transitivity rule utilizing statements with level ¿ 0
metric	confidence based on union probability of subsets of initial statements from which the goal statement $Aut_{s,t}$ can be derived
implementations	suggestion of partial use within later versions of PGP

Chapter 3

Proposed Trust Model

This chapter represents the main contribution of the thesis. Here, the probabilistic model and computational approaches found in some network reliability models (defined below) are applied to modelling trust. The goal of the proposed model is to define a general representation of trust between a set of entities and a set of trust metrics providing a means to evaluate some measure of the trust between particular entity pairs. The resulting general trust model is, at the same time, a generic reliability problem and a generalization of previous work associated with the Maurer Confidence Valuation [36]. This unified trust model is used as the framework for a new trust metric called Hop-Count Limited Transitivity, and to redefine two variations of the Maurer Confidence Valuation.

Section 3.1 will provide background on Network Reliability, which provides the theoretical model on which the proposed generalized trust model is based. Section 3.2 defines the proposed Generic Reliability Trust Model (GRTM) and associated trust metrics.

3.1 Network Reliability

Reliability theory is the study of the performance of a system of failure-prone elements. Network reliability looks in particular at evaluating the reliability of computer communications networks where reliability is concerned with the ability of a network to carry out a desired operation [12]. The following network reliability definitions are taken from [12].

A *probabilistic directed graph* $G = (V(G), E(G))$ consists of a vertex set $V(G)$ and

the set of arcs $E(G)$ where each arc in $E(G)$ corresponds to an ordered pair of vertices from $V(G)$. It is assumed that each arc e has an associated probability p_e of being operational. A *state* of G is a subset $S \subseteq E(G)$, interpreted to mean that all arcs in S are operational and all arcs in $E - S$ have failed. So a state S corresponds to the subgraph G_S of G given by $V(G_S) = V(G)$ and $E(G_S) = S$.

To describe a specific reliability model it is necessary to provide rules, here called *operational criteria*, for distinguishing between operational and non-operational states. For example, given a directed graph G, a *directed path* of length k is composed of an alternating sequence of vertices and arcs of the form $v_0, e_1 = (v_0, v_1), v_1, e_2 = (v_1, v_2), \ldots, v_{k-1}, e_k = (v_{k-1}, v_k), v_k$. An s,t-*path* is a directed path which starts with s and ends with t. Vertex s is *connected to* t if there is an s,t-path in G. For a directed s,t-reliability model, the network is defined to be operational if and only if there is an operational s,t-path in the network. Some possible definition of operational are given in Table 3.1 [45]. In general, let $OP(G)$ denote the set of all operational states for some network reliability model. Note: The terms *support set* and *pathset* have also been used as the name for members of $OP(G)$.

Table 3.1. *Traditional Network Reliability Operational Criteria*

Reliability Measure	Operational Criteria on state S
Two-terminal reliability	there exists an s,t-path in G_S
Source-to-K-terminal reliability	$\forall t \in K \subseteq V(G)$, there exists an s,t-path in G_S
Source-to-all-terminal reliability	$\forall t \in V(G) - s$, there exists an s,t-path in G_S

The *reliability* of a network G, denoted $Rel(G)$, is the probability of obtaining an operational state. Assuming that arc failures are independent, meaning the failure of an arc does not change the probability of failure of other arcs, $Rel(G)$ can be defined using state-space enumeration as follows [12, p.9]:

$$Rel(G) = \sum_{\substack{S \subseteq E(G); \\ S \text{ is operational}}} \prod_{e \in S} p_e \cdot \prod_{\substack{e \in \\ E(G)-S}} (1 - p_e). \tag{3.1}$$

A *generic reliability problem* is to determine the probability of having an operational state given a probabilistic directed graph G and the operational criteria. A useful property of all of the operational criteria to be defined in this thesis is that the operational states are *coherent*; that is, every superset of a operational state is an operational state. A *minimal operational state of a coherent system* is a state $S \in OP(G)$ where for all $a \in S$, $S - \{a\} \notin OP(G)$. For coherent systems, exact algorithms for the reliability problem exist which need only consider minimal operational states [12, p. 11].

3.2 Generic Reliability Trust Model

This thesis applies the generalized reliability model of Section 3.1 to trust modelling to create a new generalized trust model: the Generic Reliability Trust Model, or GRTM. Within this new model, trust beliefs or statements are modeled as the arcs of a probabilistic graph; the arcs are labeled with additional trust-related information, and the reliability concept of operation is defined in terms of the probability of a successful outcome of an interaction.

A *trust graph* is a labeled directed multigraph $G = (V(G), E(G))$, with the vertex set $V(G)$ representing entities and the set of arcs E consisting of ordered pairs of vertices (u, v) representing u's trust in v. Each arc $e = (u, v)$ is labeled with a *trust label* of the form $< l, c >$ where,

$l \geq 0$ is an integer representing a level for the trust where
the specific meaning is defined by particular trust metric, and,
c is a confidence value, $c \in [0, 1]$, and,
for any given level l, trust graphs can have at most one
arc (u,v) labelled with l.

In a trust graph G, for each arc e in $E(G)$, the values of the associated label $< l, c >$ can be referenced as l_e and c_e and the arc's operational probability is $p_e = c_e$. The trust label

confidence c_e, or simply *confidence*, is the probability of a successful outcome from an interaction between entities u and v as determined by u; this represents u's confidence in v and it is assumed that it is independent of any other confidence. It is assumed that trust graphs contain arcs relating to a common subject-matter. How the trust knowledge required to construct a trust graph is obtained is not considered in this thesis.

A *generic reliability trust metric* is a general reliability problem and *Trust* is the probability of obtaining an operational state; that is

$$Trust(G) = Rel(G). \tag{3.2}$$

Concrete members of this family of trust metrics are defined by specifying operational criteria associated with a general reliability problem. This process is demonstrated in the following sections through the definition of a new metric, Hop-Count Limited Transitive Trust, and representation of the MCV metric within this framework.

3.2.1 Hop-Count Limited Transitive Trust (HLTT)

Hop-Count Limited Transitive Trust (HLTT) is a new trust metric that fits into the framework of a generic reliability trust metric. For a trust graph G and arc e in $E(G)$, the trust level value l_e specifies the maximum length of a chain of intermediate entities, starting with e, through which trust can be derived. For each ordered pair of vertices (u, v), let $A_{u,v}$ be the set of all arcs of the form (u, v). If $|A_{u,v}|$ is greater than one, all arcs in $A_{u,v}$ will be removed from G except for the one labeled with the maximum trust level in $A_{u,v}$.

Definition 3.2.1 *Given a trust graph G and vertices s and t, an* HL-path *is an s,t-path P of length k, with the form $v_0, e_1 = (v_0, v_1), v_1, e_2 = (v_1, v_2), \ldots$, such that for all $e_i \in P$, $l_{e_i} \geq k - i$.*

For a trust graph G and vertices s and t in $V(G)$, *Hop-Count Limited Transitive Trust*, or $HLTT(s, t, G)$, is a generic reliability trust metric with the operational criteria that a state $S \subseteq E(G)$ is operational if and only if there exists an HL-path $\subseteq G_S$, from s to t.

A *derived statement* is a 3-tuple of the form $< u, v, l >$ where u and v are entities and l is a non-negative integer representing a trust level. These statements provide an additional mechanism for defining operational criteria. A set of initial derived statements can be created from a trust graph G using the rule:

if (u, v), labeled $< l, c >$, is in $E(G)$ then $< u, v, l >$ is a derived statement. (3.3)

Additional derived statements can be created by applying a set R of transitivity rules. The set of HLTT transitivity rules are defined as follows:

Definition 3.2.2 *HLTT Transitive Rules*

(1) Let $i > 0$, $j \geq 0$, *and* $k = min(i - 1, j)$.

If $< u, v, i >$ *and* $< v, x, j >$ *are derived statements*

then $< u, x, k >$ *is a derived statement.*

(2) Let $i > 0$.

If $< u, v, i >$ *is derived statement*

then $< u, v, i - 1 >$ *is a derived statement.*

Thus, an alternate definition of operational criteria for $HLTT(s, t, G)$ that does not use the HL-path concept is, the state $S \subseteq E(G)$ is operational if and only if the derived statement $< s, t, 0 >$ exists in the reflexive, transitive closure of the initial statements corresponding to arcs in S under the HLTT Transitivity Rules.

This use of derived statements can be generalized for multiple trust metrics within the GRTM framework. Given some set of transitivity rules R for derived statements, the trust graph G, and vertices s and t, a *generic two-terminal reliability trust metric* is a generic reliability trust metric with the operational criteria that the state $S \subseteq E(G)$ is operational if and only if the derived statement $< s, t, 0 >$ exists in the reflexive, transitive closure of the initial statements corresponding to arcs in S under R.

3.2.2 MCV Trust Metrics

The original MCV metric from [36] is redefined here within the GRTM framework. Two different metrics are presented, a full version directly emulating the original and a simplified version called MCVa. Both metrics define a generic reliability trust metric on the trust graph G with the operational criteria that the state $S \subseteq E(G)$ is operational if and only if the statement $< s, t, 0 >$ exists in the reflexive, transitive closure of the initial statements corresponding to arcs in S under the specific transitivity rules below.

Definition 3.2.3 *MCV Transitivity Rules*

$$\begin{aligned} &\textit{Let } i > 0 \textit{ and } j = i - 1. \\ &\textit{If} < u, v, 0 > \textit{and} < u, v, i > \textit{and} < v, x, j > \textit{are derived statements} \\ &\textit{then} < u, x, j > \textit{is a derived statement.} \end{aligned} \tag{3.4}$$

Definition 3.2.4 *MCVa Transitivity Rules*

$$\begin{aligned} &\textit{Let } i > 0,\ j \geq 0, \textit{ and } k = min(i - 1, j). \\ &\textit{If} < u, v, 0 > \textit{and} < u, v, i > \textit{and} < v, x, j > \textit{are derived statements} \\ &\textit{then} < u, x, k > \textit{is a derived statement.} \end{aligned} \tag{3.5}$$

The metric based on the MCV Transitivity Rule requires an arc with level zero and arcs with level$= i$ for each possible recommendation level from one to some maximum trust level MTL, thus there are $MTL+1$ trust graph edges between a given vertex-pair. With the simplified metric based on the MCVa transitivity rules, only the level zero and maximum trust level are represented, the intermediate trust levels not being explicitly stated. Thus, trust from u to v in the trust graph G is represented by at most two arcs; one arc (u, v) labeled with trust level zero , and an additional optional arc (u, v) labeled with a maximum recommendation trust level l greater than zero.

Chapter 4

Algorithms

In general, computing reliability is a #P-complete problem[12]. For a graph G with m arcs, complete state enumeration, as described in Section 3.1, would generate or consider 2^m states representing all possible subsets of arcs in $E(G)$. Two algorithms will be presented for calculating an exact value for a variety of reliability problems: inclusion-exclusion in Section 4.1 and factoring in Section 4.2. As with complete state enumeration (see the reliability Definition 3.1), both of these approaches can be used to calculate any generic reliability trust metric as long as the systems are coherent [12, 26, 45]. They require exponential time and possibly exponential memory. A faster algorithm which calculates a conservative approximation is presented in Section 4.3. The chapter concludes with performance results from a software implementation in Section 4.4.

4.1 Inclusion-Exclusion

The basic inclusion-exclusion algorithm to calculate reliability is:

- determine all the minimal operational states, and then
- apply inclusion-exclusion to all combinations of these sets.

Maurer actually defines the original MCV trust metric in terms of inclusion-exclusion on minimal operational subsets of initial statements [36, Section 4.4]. This thesis used the alternate but equivalent definition using state space enumeration when MCV was presented in Section 2.4.7 because this formulation has greater simplicity.

From [12, section 2.4.2], given the trust graph G and a set T of all minimal operational states from $OP(G)$ as defined in Section 3.1, the inclusion-exclusion formula for calculating the reliability of coherent systems is:

$$Rel(G) = \sum_{\substack{S \subseteq T, \\ S=\{T_1,T_2,\ldots,T_k\} \\ \text{for } k \geq 1.}} (-1)^{k+1} \prod_{e \in T_1 \cup T_2 \cup \ldots \cup T_k} p_e. \tag{4.1}$$

One method of generating all nonempty subsets of a k-set T from Equation 4.1 is by incrementing a k bit binary number, starting at the value one. Interpret the ith bit as meaning that T_i is included in the subset if the bit is one and excluded if it is zero. Note that the union of the T_i's produces a set of arcs from $E(G)$. If all subsets of the k-set are enumerated (except the empty set), there are $2^k - 1$ such subsets.

The time to perform inclusion-exclusion is exponential in the number of minimal operational sets $|T|$. In the worst case, the generation of T will require time exponential in the number of arcs, and may require exponential memory as $|T|$ may also be exponential in the number of arcs, and enumeration of all k-subsets takes exponential time; hence the inclusion-exclusion approach is often characterized as doubly-exponential [12, 36]. Thus, this method is only practical when the number of arcs is small and $|T|$ is modest.

The follow two sections contain examples calculations of the HLTT and MCV metrics using inclusion-exclusion.

4.1.1 Example: Does Alice Trust Bob?

Alice has cause to remotely interact with Bob. In order to interact, her personal policy is that she must be able to determine trust with that person with a confidence of at least 80%. Through some mechanism, she collects the relevant trust statements seen in Figure 4.1. One possible interpretation for the statements and the associated confidences is that while Alice trusts Sally and Tom, she does not trust Sally's ability to make recommendations as much as she trusts Tom's; possibly this is based on some knowledge of the process and due-diligence he puts into recommendations.

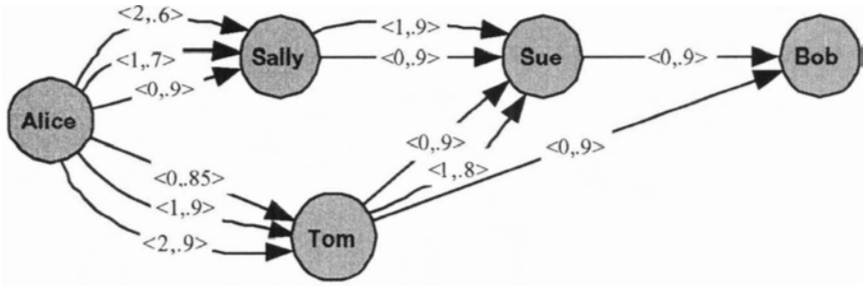


Figure 4.1. *The Trust Graph for Alice and Bob*

In terms of the model used here, Alice's objective is to validate that the confidence of the derived statement $< Alice, Bob, 0 >$ is at least 0.8. The following sections look at how trust may be calculated using the HLTT rules and then the MCV rules.

4.1.1.1 HLTT Example

Step 1: Determine minimal support sets

By manual analysis applying the HLTT transitivity rule (Definition 3.2.2), there are three possible minimal operational states supporting trust from Alice to Bob – these are listed in Table 4.1.

Table 4.1. *Alice-Bob Minimal Operational States under HLTT Rules*

Label	Operational State (membership shown as set of arc/label)
S_1	$\{ (Alice, Sally)/ < 2, .6 >, (Sally, Sue)/ < 1, .9 >, (Sue, Bob)/ < 0, .9 > \}$
S_2	$\{ (Alice, Tom)/ < 2, .9 >, (Tom, Bob)/ < 0, .9 > \}$
S_3	$\{ (Alice, Tom)/ < 2, .9 >, (Tom, Sue)/ < 1, .8 >, (Sue, Bob)/ < 0, .9 > \}$

Step 2: Perform Inclusion-Exclusion

Enumerate subsets of operational states, compute the union of the arcs in each subset, compute the signed products of the arc confidences [terms with k-subsets are multiplied by

$(-1)^{k+1}$], and combine the results using inclusion-exclusion.

1-subsets:

$$(-1)^2(P(S_1) + P(S_2) + P(S_3))$$
$$= (1)(0.486 + 0.810 + 0.648)$$
$$= 1.944$$

2-subsets:

$$(-1)^3(P(S_1 \cup S_2) + P(S_2 \cup S_3) + P(S_1 \cup S_3))$$
$$= (-1)(0.394 + 0.583 + 0.350)$$
$$= -1.327$$

3-subsets:

$$(-1)^4(P(S_1 \cup S_2 \cup S_3))$$
$$= 0.315$$

The sum of these subsets is 0.932.

Step 3: Application-specific Trust Decision

The result 0.932 is greater than .8. Therefore, Alice can trust Bob. Observe that state S_2 on its own could have satisfied the desired 80% confidence threshold. This suggests an early stopping condition for an approximate algorithm – this is considered in Section 4.3.

4.1.1.2 MCV Example

Step 1: Determine minimal support sets

By manual analysis and applying the MCV transitivity rule (Definition 3.2.3), there are five possible minimal operational states supporting trust from Alice to Bob – these are listed in Table 4.2 and shown in Figure 4.2.

.

Table 4.2. *Alice-Bob Minimal Operational States under MCV Rules*

Label	Operational State (membership shown as set of arc/label)
S_1	$\{(Alice, Sally)/ < 0, .9 >, (Alice, Sally)/ < 1, .7 >, (Alice, Sally)/ < 2, .6 >$ $, (Sally, Sue)/ < 0, .9 >, (Sally, Sue)/ < 1, .9 >, (Sue, Bob)/ < 0, .9 >\}$
S_2	$\{(Alice, Tom)/ < 0, .85 >, (Alice, Tom)/ < 1, .9 >, (Tom, Bob)/ < 0, .9 >\}$
S_3	$\{(Alice, Tom)/ < 0, .85 >, (Alice, Tom)/ < 1, .9 >, (Alice, Tom)/ < 2, .9 >$ $, (Tom, Sue)/ < 0, .9 >, (Tom, Sue)/ < 1, .8 >, (Sue, Bob)/ < 0, .9 >\}$
S_4	$\{(Alice, Sally)/ < 0, .9 >, (Alice, Sally)/ < 1, .7 >, (Sally, Sue)/ < 0, .9 >$ $, (Alice, Tom)/ < 0, .85 >, (Alice, Tom)/ < 1, .9 >, (Alice, Tom)/ < 2, .9 >$ $, (Tom, Sue)/ < 1, .8 >, (Sue, Bob)/ < 0, .9 >\}$
S_5	$\{(Alice, Sally)/ < 0, .9 >, (Alice, Sally)/ < 1, .7 >, (Alice, Sally)/ < 2, .6 >$ $, (Sally, Sue)/ < 1, .9 >, (Alice, Tom)/ < 0, .85 >, (Alice, Tom)/ < 1, .9 >$ $, (Tom, Sue)/ < 0, .9 >, (Sue, Bob)/ < 0, .9 >\}$

Step 2: Perform Inclusion-Exclusion

Enumerate subsets operational states, compute the union of the arcs in each subset, compute the signed product of the arc confidences, and combine results using inclusion-exclusion.

1-subsets:

$$(-1)^2(P(S_1) + P(S_2) + P(S_3) + P(S_4) + P(S_5))$$
$$= (1)(0.276 + 0.689 + 0.446 + 0.281 + 0.211)$$

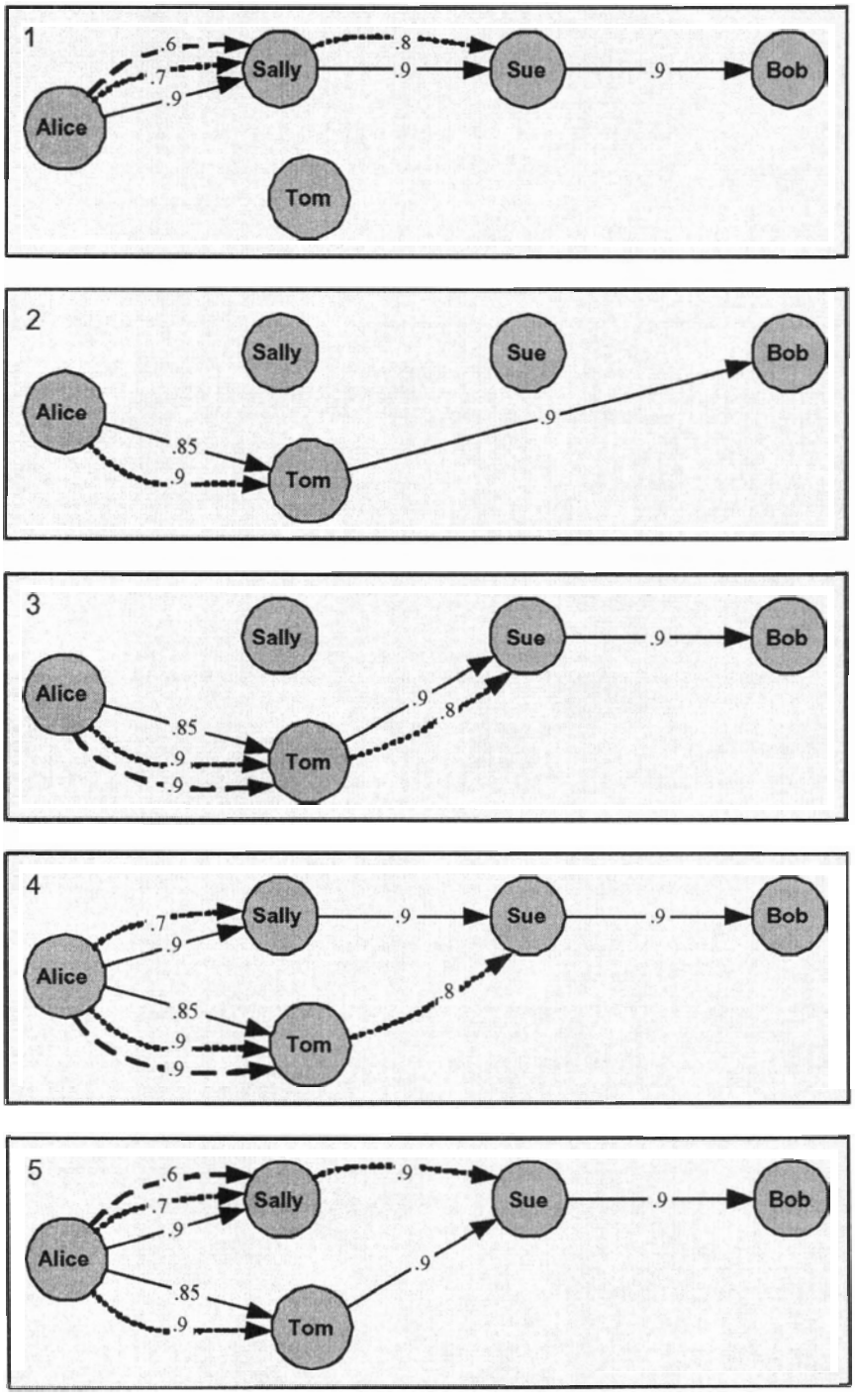


Figure 4.2. *Minimal Support Subgraphs for trust from Alice to Bob*

$$= 1.902$$

2-subsets:

$$(-1)^3(P(S_1 \cup S_2) + P(S_2 \cup S_3) + P(S_1 \cup S_3) + P(S_3 \cup S_4) + P(S_2 \cup S_4)$$
$$+ P(S_1 \cup S_4) + P(S_4 \cup S_5) + P(S_3 \cup S_5) + P(S_2 \cup S_5) + P(S_1 \cup S_5))$$
$$= (-1)(0.190 + 0.402 + 0.137 + 0.253 + 0.253$$
$$+ 0.152 + 0.137 + 0.152 + 0.190 + 0.190)$$
$$= -2.053$$

3-subsets:

$$(-1)^4(P(S_1 \cup S_2 \cup S_3) + P(S_1 \cup S_3 \cup S_4) + P(S_2 \cup S_3 \cup S_4) + P(S_1 \cup S_2 \cup S_4)$$
$$+ P(S_1 \cup S_4 \cup S_5) + P(S_2 \cup S_4 \cup S_5) + P(S_3 \cup S_4 \cup S_5) + P(S_1 \cup S_3 \cup S_5)$$
$$+ P(S_2 \cup S_3 \cup S_5) + P(S_1 \cup S_2 \cup S_5))$$
$$= (1)(0.123 + 0.137 + 0.228 + 0.137$$
$$+ 0.137 + 0.123 + 0.137 + 0.137$$
$$+ 0.137 + 0.171)$$
$$= 1.464$$

4-subsets:

$$(-1)^5(P(S_1 \cup S_2 \cup S_3 \cup S_4) + P(S_1 \cup S_2 \cup S_4 \cup S_5) + P(S_2 \cup S_3 \cup S_4 \cup S_5) +$$
$$P(S_1 \cup S_3 \cup S_4 \cup S_5) + P(S_1 \cup S_2 \cup S_3 \cup S_5))$$
$$= (-1)(0.123 + 0.123 + 0.123 +$$
$$0.137 + 0.123)$$
$$= -0.628$$

5-subsets:

$$(-1)^6(P(S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_5))$$
$$= 0.123$$

The sum of these values is 0.807.

Step 3: Application-specific Trust Decision

The result 0.807 is greater than 0.8. Therefor, Alice can trust Bob. Observe that if only some of the operational subsets are taken into account, the desired threshold cannot be reached.

4.1.2 Enumerating Minimal Operational States

The following sections describe algorithms for enumerating minimal operational states which are the defining characteristic of the trust metrics presented in this thesis. For each metric, some essential pattern in the corresponding transitivity rules of Section 3.2 is used as the basis of an algorithm to search the arcs of a trust graph.

4.1.2.1 HLTT

This section describes an algorithm for generating all possible minimal operational states for the trust metric Hop-count Limited Transitive Trust, defined in Section 3.2.1. The basic approach is to perform an exhaustive depth-first search of the trust graph G, attempting to construct HL-paths from vertex s to t (see Definition 3.2.1). Pseudo-code for this algorithm, called here *dfsHLpaths()*, is shown in Figure 4.3.

Unlike the generalized depth-first search algorithm *dfs()* described in [2, Section 7.3.1], the dfsHLpaths() algorithm has the objective of enumerating all HL-paths to a particular target vertex. The functions of marking and backtracking described in the dfs() algorithm are represented in dfsHLpaths() by the use of *currentPath* variable, a list containing all the vertices/edges of a partial path, or *path prefix*, which has been discovered up to a specific point in the search. It ensures no cycles by testing for the existence of a vertex in the current path prefix before the vertex is appended. The HL-paths discovered by dfsHLpaths() are minimal by construction.

```
    G is a trust graph.
    s is the source vertex.
    t is the target vertex.
    visiting is the current vertex being visited.
    maxHops is the maximum hops which can be taken from the current vertex.
    currentPath is an HL-path under construction.

1   Function GenerateMCV-OperationalStates(G, s, t)
2     return dfsHLpaths(G, s, t, {s}, ∞);

3   Function Set dfsHLpaths(G, visiting, t, currentPath, maxHops): returns a set of sets
    of arcs
4     set foundPaths = {};
5     for each arc e = (u, v) ∈ E(G) s.t. u = visiting
6       set newMaxHops = min(e_l, maxHops − 1);
7       set newPath = currentPath;
8       append e to newPath;
9       append v to newpath;
10      if v = t then
11        foundPaths = foundPaths ∪ {newPath};
12      else if newMaxHops > 0 and
13           v ∉ currentPath then
14        foundPaths = foundPaths∪
              dfsHLpaths(G, nextVertex, t, newPath, newMaxHops);
15    end-for;
16    return foundPaths;
```

Figure 4.3. *Generation of HLTT minimal operational states.*

Even with the hop-count limit which is inherent to HLTT, for a graph with m arcs this algorithm might consider $m!$ arcs and generate a possibly exponential number of paths. Let $MaxLevel(s)$ be the maximum level label of the arcs out of vertex s. Based on the definition of an HL-path, the maximum length of any path considered which starts at s is $MaxLength = MaxLevel(s) + 1$. Assuming a worst case of a complete graph with n vertices, but with $MaxLength < n$, the maximum number of arcs considered will be:

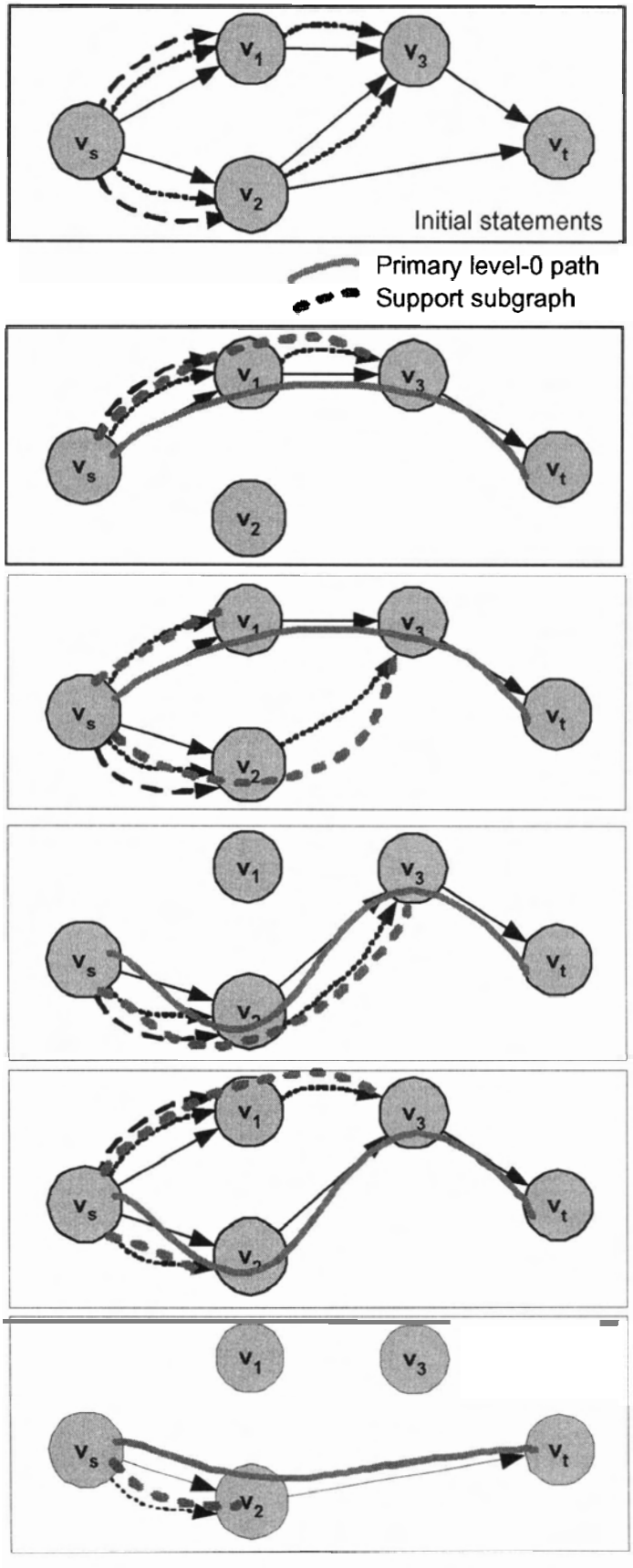
$$\prod_{i=1}^{MaxLength} (n-i) = (n-1)(n-2)\ldots(n-MaxLength)$$

$$\in O(n^{MaxLength})$$

4.1.2.2 MCV

This section describes an algorithm for enumerating the minimal operational states of a graph using the MCV rule in Definition 3.2.3. The spine of the solution is the observation that an *level zero path*, an s,t-path in trust graph G formed when $E(G)$ is restricted to edges with a level label value of zero, is a subset of an operational state. An example of this is shown in Figure 4.4. The basic algorithm is to enumerate all level zero paths from vertex s to t, and then backtrack along each of these paths and exhaustively search for additional arc and vertices required to support the MCV rule. These additional vertices and arcs are called here a *support subgraph*. This general approach can be used for enumerating states valid under both the MCV and MCVa transitivity rules. Pseudo-code specifically for the more complicated MCV rules is found in Figures 4.5, 4.6, 4.7, and 4.8.

The MCV minimal operation state enumeration algorithm exhaustively searches for minimal sets of arc required to satisfy the MCV transitivity rules, with costs triply exponential in both time and space. The outer loop in *GenerateMCV-OperationalStates()* in Figure 4.5 involves generation of all, possibly exponential in number, s,t-paths composed of level zero arcs of the input graph G. A *support set* for vertex $s \in V(G)$, arc $e = (u,v) \in E(G), u \neq s$, is a minimal set of arcs in $E(G)$ required to derive the statement $< s, v, l_e >$ using a metric's transitivity rules. The next step is to generate all, possibly exponential in number, support sets for each arc of every path using MCV transitivity rules (Definition 3.2.3) which require each intermediate vertex in the path to have a incoming level $i+1$ arc to support an exiting level i arc. Finally, operational states are created by combining each of the level zero paths with all possible combinations of support sets associated with each arc of the path – another step with possible exponential cost in time and space.


v_1
v_3
v_s
v_t
v_2
Initial statements
Primary level-0 path
Support subgraph

```
    G is a trust graph.
    s is the source vertex.
    t is the target vertex.
    level is minimum trust level of the arcs entering t,
      this is required for generation of support subgraphs.

1   Function GenerateMCV-OperationalStates(G, s, t, level): returns a set of sets of arcs

    minOpStates is a set of sets representing operational states, initially empty.

2     for each level zero path p in G
3     do;
4         supportCombinations = GenerateMCV-PathSupport(G, s, level, p); [defined in
    Figure 4.6]
5       minOpStates = minOpStates ∪ {supportCombinations};
6     end;
7     return minOpStates;
```

Figure 4.5. *Generation of MCV minimal operational states.*

```
    G is a trust graph.
    s is the source vertex.
    level is the required trust level.
    zpath is a level zero path (a kind of s,t-path).

1   Function GenerateMCV-PathSupport(G, s, level, zpath): returns a set of sets of arcs

    representing possible operational states based on zpath
      v_i is the i'th vertex in zpath.
      support is an array, indexed from one, of sets of arcs, initially empty,
        where support[i] represents the support subgraph associated with v_i.
        support[0] is not relevant as v_0 is the source vertex.
      set k = length of zpath;

2     if level > 0 then
3       // note: level zero support already represented by zpath
4       set support[k] =GenerateMCV-VertexSupport(G, s, v_k, level);
              [defined in Figure 4.7]
5       if support[k] is empty then
6         return {};
7     end-if;

      //search for level one support for each intermediate vertex in the given level zero path
8     for i = k - 1 down to 1
9       set support[i] = GenerateMCV-OperationalStates(
10            G - {v_i, v_{i+1}, ..., v_k}, s, v_i, 1 );
11      if support[i] is empty then
12        return {};
13    end-for;

14    return GeneratSupportCombinations(zpath, support);
          [defined in Figure 4.8]
```

Figure 4.6. *Generate possible support subgraphs for each intermediate vertex of a level zero path.*

```
    G is a trust graph.
    s is the source vertex.
    t is the target vertex.
    level is the required trust level.

1   Function GenerateMCV-VertexSupport(G, s, t, level) returns a set of arcs
      supportOptions is a set of sets of arcs, initially empty.
2     set inSupport be the set of arcs e = (u, v) in E(G) such that v = t and l_e ≥ level

3     for each arc e = (u, v) in inSupport
4       if u = s then
5         set supportOptions = supportOptions ∪ {{e}};
6       else
7         set intermediateSupport =
8             GenerateMCV-OperationalStates(G, s, u, level + 1);
9         if intermediateSupport is not empty then
10          supportOptions = supportOptions ∪ {intermediateSupport};
11    end-for;

12    return supportOptions;
```

Figure 4.7. *Generate MCV additional level+1 support.*

```
    zpath is a level zero path.
    supportOptions is a array, starting at one, of sets of arcs representing support
      combinations associated with each vertex in zpath after the source vertex v_0.
    supportCombinations is an array of sets of arcs.

1   Function GenerateSupportCombinations(zpath, supportOptions): returns a set of sets
    of arcs
2     set supportCombinations[1] = {};
3     for i = 1 to length of zpath;
4       set e_i be the i'th arc in zpath;
5       set numCombinations = size of supportCombinations;
        // append the arc from the zero level path
6       for j = 1 to numCombinations
7         supportCombinations[j] = supportCombinations[j] ∪ {e_i};
8       end-for;

        // generate all combinations of supportOptions[i] for each step in zpath
        // supportCombinations[] contains a partial result or prefix based
        // on previous steps in zpath
9       set count =| supportOptions[i] |;
        // create new combinations if multiple support options and
        // append arcs from option to all previous combination prefixes
10      for each set s in supportOptions[i]
11        for j = 1 to numCombinations
12          supportCombinations[j * count] = supportCombinations[j] ∪ s;
13        end-for;
14        set count = count - 1;
          // the last value will be 1, which will update the previous set of prefixs
15      end-for;
16    end-for;

17    remove duplicates from supportCombinations;
18    return supportCombinations;
```

Figure 4.8. *Generate MCV additional level support for a vertex.*

4.2 Factoring

The basic factoring algorithm considers the reliability of the graph G by looking at two smaller versions of G after selecting an arc e in $E(G)$; one version where e is assumed operational, or always present, and one where e is assumed to have failed. More specifically, given a probabilistic undirected graph $G = (V(G), E(G))$, and some arc $e \in E(G)$ with probability of operation p_e, $G * e$ denotes 'G *contract* e' and means to reduce or mark edge e as always operational. Similarly, $G - e$ is read 'G *delete* e' and means to reduce or mark e as permanently failed. From [12], the *Factoring Theorem* states that for any reliability measure Rel of a coherent system,

$$Rel(G) = p_e Rel(G * e) + (1 - p_e) Rel(G - e) \tag{4.2}$$

In each iteration of the factoring algorithm, the input graph is reduced by contracting and deleting one arc, representing the consideration of the impact of that arc on overall reliability. In the path-based models of some network reliability models where factoring has been applied, a variety of additional probability preserving reductions are possible in each iteration. These are used to significantly reduce the number of states to be examined [12, 26]. For example, series reduction would replace two arcs $e_1 = (u, v), e_2 = (v, x)$, with associated operational probabilities p_{e_1}, p_{e_2}, with a new arc $e_{1,2} = (u, x)$, with $p_{e_{1,2}} = p_{e_1} \cdot p_{e_2}$. A parallel reduction would replace two arcs $e_1 = (u, v), e_2 = (u, v)$ with associated operational probabilities p_{e_1}, p_{e_2}, with a new arc $e_{1,2} = (u, v)$, with $p_{e_{1,2}} = p_{e_1} + p_{e_2} - p_{e_1} p_{e_2}$. In application of factoring to the metrics of this thesis, these additional graph reductions are not utilized. This is because they are not generally probability-preserving due to the presence of additional dependencies between arcs represented by the transitivity rule of a trust metric. In its use here, contracted arcs are simply marked as "always operational". This use of factoring without additional graph reductions can also be seen as a recursive algorithm to perform state enumeration [12]. The algorithm shown in Figure 4.9 presents use of the factoring theorem to compute reliability. It does not perform complete state enumeration as the new graphs produced are only explored until the set of contracted (i.e.

always present or operational) edges either always or never contain an operational state.

```
     G is a trust graph.
     s is the source vertex.
     t is the sink or target vertex.
     contractedSet is the set of contract arcs in E(G) (a.k.a. always-present).
     notDeletedSet is the set of arcs in E(G) which have not been deleted or contracted.

1    Function PerformFactoring(G, source, target,
          contractedSet, notDeletedSet): returns reliability
2    Set candidateSet = notDeletedSet - contractedSet;
3    if (candidateSet is empty) then return (0.0);
4    Set e = some (u, v) in candidateSet;

5    contractedSet = contractedSet ∪ {e};
6    if there exists an always operational state in contractedSet then
7       operationalProb = p_e;
8    else
9       operationalProb = p_e *
                PerformFactoring( G, s, t, contractedSet, notDeletedSet);
10   end-if

11   contractedSet = contractedSet − {e};
12   notDeletedSet = notDeletedSet − {e};

13   if there exists an operational state in notDeletedSet then
14      failureProb = (1 − p_e)*
                PerformFactoring(G, s, t, contractedSet, notDeletedSet );
15   else
16      failureProb = 0;
17   end-if

18   notDeletedSet = notDeletedSet ∪ {e};
19   return operationalProb + failureProb;
```

Figure 4.9. *Pseudo-code for basic factoring.*

Processing in Figure 4.9 is started with the call,

```
trust = PerformFactoring(G, s, t, {}, E(G));
```

Line 4 represents selecting an arc for factoring. With the random arc selection approach, the total number of recursive calls varies. Lines 6 and 13 represent the points where the model-specific operational criteria would be applied; for example, the restricted path definition

3.2.1 in the case of the HLTT metric. Lines 2 and 12 are points where the graph has been reduced using the selected arc.

For a graph with m arcs, this use of factoring will consider in the worst case $O(2^m)$ states. Unlike the previous inclusion-exclusion algorithm, it does not require pre-calculation of all minimal operational states – in practical terms, where the inclusion-exclusion algorithm will run out of memory, the factoring algorithm will not. This method is practical when the number of arcs is small and may be more effective than the inclusion-exclusion algorithm when there are a large number of minimal operational states.

4.2.1 Testing for Operational States

The following sections describe algorithms for determining if a state is operational for the different trust metrics presented in this thesis. These recognition algorithms are used within the algorithm of Figure 4.9 to test if the set of contracted arcs contains an operational state or if the set of arcs not deleted contains an operational state, corresponding to Lines 6 and 13. The basic method is to the apply transitivity rules of a trust metric of Section 3.2 to search the given arcs. A key difference in this approach is the search for any operational state, as opposed to the enumeration of states which formed part of the inclusion-exclusion approach of Section 4.1.

The algorithms to test for an operational state in a subset of arcs apply a limited form of transitive closure using on a modified version Warshall's matrix algorithm [2, Section 9.3]. Let S be a state of the trust graph G. From [2], a binary relation A on S can be represented by an $n \times n$ Boolean matrix with entries:

$$a_{ij} = \begin{cases} 1 & \text{if } s_i A s_j \\ 0 & \text{otherwise.} \end{cases}$$

The trust graph G, $|V(G)| = n$ is represented as an $n \times n$ integer matrix M such that

$$M_{i,j} = \begin{cases} l & \text{iff the derived statement } < v_i, v_j, l > \\ & \text{exists in the reflexive transitive closure of} \\ & \text{initial statements representing S,} \\ & \text{under the HLTT transitivity rules, and} \\ -1 & \text{otherwise.} \end{cases}$$

The operation of the transitivity rule is limited to the perspective of a given source entity. In addition, an early stopping condition is introduced; at any time the existence of transitive trust from a given source to target will cause the algorithm to complete.

4.2.1.1 HLTT

To apply the matrix approach to testing for HLTT operational states, the binary relation T on S is defined as: sTt if and only if there exists the derived statement $< s, t, l >$ where $l \geq 0$ in the reflexive, transitive closure of the initial statements corresponding to arcs in S under the HLTT Transitivity Rules. Note that in [2], the relation T is defined in terms of the existence of a path from s to t and is known as the *reachability relation* R. The following is a limited transitive closure matrix algorithm for using the HLTT transitivity rule. The fixed source perspective is represented by the fixed matrix index representing the source vertex in the transitivity test on line (1).

The algorithm in Figure 4.10 computes the HLTT transitivity rule for all possible vertex-pairs until the desired trust level is achieved (see line 18) or until the closure is completed, as indicated by no new transitivity being computed after evaluating all vertex-pairs. The rule test is represented by lines 12 and 13. For a strongly connected graph with n vertices, this algorithm might, in the worst-case, perform $O(n^3)$ iterations. With a maximum trust level l_{max} of the arcs exiting the source, where l_{max} is less then n, the maximum number of iterations will be function of the possible longest s,t-path, which will be l_{max} plus one. In the worst case, each execution of the outer loop would only compute one hop in the path, requiring $O(l_{max}n^2)$ iterations.

4.2.1.2 MCVa

The MCVa transitivity rule in Section 3.2.2 requires representation of both the zero and maximum trust levels. The pseudo-code in Figure 4.11 presents an algorithm to test for the existence of an operational state based on this transitivity rule.

```
    G is a trust graph.
    source is the source vertex.
    target is the target vertex.
    desiredLevel is the desired trust level.
    M is a 2-D integer matrix representing the maximum trust level the pairs of vertices.

1   Function testHLTT(G, source, target, desiredLevel): return true if
      trust exists from source to target at the desired level, as defined by HLTT
2    Set M = E(G) represented as a matrix;
3    Set s = index of source in M;
4    Set t = index of target in M;
5    if M_{s,t} ≥ desiredLevel then return true; // trivial case

6    Set hasChanged = true;
7    while hasChanged
8        Set hasChanged = false;
9        for i = 1 to n
10           if M_{s,i} ≤ 0 then continue;

11           for j = 1 to n
12               if M_{i,j} ≥ 0 then
13                 Set newLevel = min( M_{s,i} − 1, M_{i,j} );
14                 if newLevel > M_{s,j} then
15                    Set M_{s,j} = newLevel;
16                    Set hasChanged =true;
17                 end-if;

18                 if M_{s,t} ≥ desiredLevel then return true;
19               end-if;
20           end-for;

21       end-for;
22   end-while;
23   return false;
```

Figure 4.10. *Pseudo-code for HLTT matrix transitive closure.*

```
    G is a trust graph.
    source is the source vertex.
    target is the target vertex.
    desiredLevel is the desired trust level.
    M an integer matrix representing the maximum trust level between two vertices.
    Z an binary matrix representing existence of zero trust level between two vertices.

1   Function testMCVa(G, source, target, desiredLevel): return true if
      trust exists from source to target at the desired level, as defined by MCVa.
2     Set M = E(G) represented as a matrix, using max trust level between vertex pairs;
3     Set Z = E(G) represented as a matrix, using zero trust level between vertex pairs;
4     Set s = index of source in M;
5     Set t = index of target in M;
6     Set hasChanged = true;
7     while hasChanged
8       Set hasChanged = false;

9       for i = 1 to n

10        if M_{s,i} ≤ 0 then continue;

11        for j = 1 to n

12          if Z_{s,i} and M_{s,i} > 0 and M_{i,j} ≥ 0 then
13            if Z_{i,j} and not Z_{s,j} then
14              Set Z_{s,j} = true;
15              Set hasChanged = true;
16              if M_{s,j} < 0 then Set M_{s,j} = 0;
17            end-if;

18            Set newLevel = min(M_{s,i} − 1, M_{i,j});
19            if newLevel > M_{s,j} then
20              Set M_{s,j} = newLevel;
21              Set hasChanged =true;
22            end-if;
23            if M_{s,t} ≥ desiredLevel then return true;
24          end-if

25        end-for

26      end-for

27    end-while
28    return false;
```

Figure 4.11. *Pseudo-code for MCVa matrix transitive closure.*

4.3 Approximation Heuristic

Since general reliability problems are #P-complete [12], it is not surprising that none of the exact algorithms run in polynomial time. This section looks at improving the performance of trust metric calculation by relaxing the requirement for an exact solution. Instead, the focus is on calculating a conservative approximation. In the context of trust, this means that the approximation must be less than or equal to the exact value. This section presents the outline of a heuristic algorithm *discard-inclusion-exclusion* which produces a conservative estimate of a trust metric and is faster than the exact version of inclusion-exclusion.

The *discard-inclusion-exclusion* algorithm modifies both phases of the the inclusion-exclusion algorithm presented in Section 4.1. The *searching phase* is the phase of the algorithm which enumerates all the minimal operational states T. This is followed by a second phase, the *inclusion-exclusion phase*, where the $2^{|T|}$ subsets of the set T are considered in the inclusion-exclusion calculation. The basic approach of the heuristic is to restrict the generation of all minimal operational states and further reduce the size of T before performing inclusion-exclusion.

Given the trust graph G, and the state $S \subseteq E(G)$ which is a potential operational state, it is assumed that in the search phase, there are points in any metric-specific algorithm where S is updated to include a new arc $e \in E(G)$, such that $S' = S \cup \{e\}$ and,

$$Prob(S) = \prod_{e \in S} p_e.$$

From the definition of $Prob(S)$ it is obvious that for any arc set S and arc e not in S, if $T = S \cup \{e\}$ then $Prob(T) \leq Prob(S)$. Thus, a minimum threshold value, $ProbabilityFloor$, can be used to discard a *candidate operational state* S, a set arcs which is a subset of some operational state, if $Prob(S) < ProbabilityFloor$. This threshold can be used in a metric-specific minimum operational state search or an enumeration algorithm to abandon further consideration of candidate states. In addition, if the application context can specify a minimum desired metric value, $DesiredConfidence$, this value can be used globally to stop the algorithm at any point where the probability is high enough. As each minimum

operational state S is enumerated, if $Prob(S)$ greater or equal to $DesiredConfidence$ then the search can be stopped and $Prob(S)$ returned as the approximate metric result, without performing further computation. An additional limit on the clock time of algorithm execution, $MaxTime$, can be used to stop the search-phase before enumerating all states.

Let T be the set of minimal operation states of the trust graph G. The key observations used in the inclusion-exclusion phase are the following.

1. If the number of states input to inclusion-exclusion operation is limited to some maximum $MaxStates$, then there will be at most $2^{MaxStates} - 1$ subsets enumerated.
2. The states $S \in T$ that contribute the most to the result of inclusion-exclusion are generally the ones that have the highest values of $Prob(S)$ and share the fewest arcs with the other members of T.
3. Performing inclusion-exclusion using a subset T' of all the minimal operational states will produce a result that is less-or-equal to the result obtained if the entire set T had been used.

Based on the above observations, our algorithm performs the following reduction on T in the inclusion-exclusion phase, just before performing inclusion-exclusion. For the state $S \in T$, let $DisjointCount(S,T)$ be the number of arcs in S which are not members of any other state in T. The algorithm sorts the input set of minimal operational states (enumerated in the search phase) in decreasing order of $Prob(S)$ and $DisjointCount(S,T)$, then deletes all but the first $MaxStates$ members.

If state searching is stopped before exhausting all viable options, or states are removed from consideration in the inclusion-exclusion phase then the algorithm may no longer enumerate or consider all minimum operational states and the resulting metric will a conservative approximation of the exact metric value. As these approximation decisions are explicit, based on threshold value tests, the implementation can use a flag variable to record and report the fact that the result is an approximation. If approximation flag is $false$, then the algorithm is reporting an exact result. There is no indication as to the quality of the

```
    G is a trust graph.
    s is the source vertex.
    t is the target vertex.
    probFloor is the minimum contribution for an operational state.
    desiredConfidence is the target minimum probability desired.
    maxStates is the maximum desired number of states on which to perform inclusion-
    exclusion.
    maxTime is the maximum real time allowed for operational state generation.
    approximationFlag is a global flag.

1   Function TrustMetricApproximation(G, s, t,
        probFloor, desiredProb, maxStates, maxTime ): return confidence
2     Set approximationFlag = false;
3     Set startTime = current time;

4     Set operationalStates =EnumerateOperationalStates(G, s, t,
        probFloor, desiredConfidence, maxTime);

5     if |operationalStates| > maxStates then
6       Sort operationalStates by Prob(S), breaking ties by placing sets with
          a max number of element disjoint from other sets earlier in the ordering.
7       Delete all elements of operationalStates with an index higher than maxStates;
8       Set approximationFlag =true;
9     end-if;
10    return InclusionExclusionProb( operationalStates );
```

Figure 4.12. *Pseudo-code for generic approach to approximation of a trust metric using inclusion-exclusion.*

approximation, other than the approximation flag. A pseudo-code for both phases of the discard-inclusion-exclusion algorithm is in Figures 4.12 and 4.13.

From Equation 4.1, if subsets are considered in increasing order, the number of states in T from which they are composed, then stopping after all subsets of i states have been considered produces an upper bound on the desired probability if i is odd and a lower bound if i is even. This approach is suggested for future research.

```
    G is a trust graph.
    s is the source vertex.
    t is the target vertex.
    probFloor is the minimum contribution for an operational state.
    desiredConfidence is the target minimum probability desired.
    maxTime is the maximum real time allowed for operational state generation.
    approximationFlag is a global flag.

1   Function EnumerateOperationalStates(G, s, t,
        probFloor, desiredConfidence, maxTime ): return set of sets of arcs
2    Set approximationFlag = false;
3    Set startTime = current time;
4    Set operationalStates = {};
5    while searching for minimal operational states of G
6      if current time - startTime < maxTime then
7        Set approximationFlag =true;
8        return operationalStates;
9      end-if;
10     ... perform metric-specific processing ...
11     candidateState = candidateState ∪ {e};
12     if Prob(candidateState) < probFloor then
13       Set approximationFlag =true;
14       // abandon further searching based on candidateState
15     end-if;
16     if candidateState is an operational state then
17       if Prob(candidateState) >= desiredDesired then
18         Set approximationFlag = true;
19         return candidateState;
20       end-if;
21       operationalStates = operationalStates ∪ {candidateState};
22     end-if;
23     ... continue metric-specific processing ...
24   end-while;

25   return operationalStates;
```

Figure 4.13. *Pseudo-code for search phase of approximation heuristic.*

The performance of the search phase depends on the input graph and value of the various thresholds. The difference between the approximation and exact value will depend on the number and probability contribution of excluded states. The number of minimum operational states not enumerated will depend on the specific value of $DesiredConfidence$, the range of arc probabilities in the input graph, the value of $MaxTime$ relative to the actual number of operational states and machine performance. If the value of $ProbabilityFloor$ is zero, all of the arc probabilities are one, $MaxStates = \infty$, and $MaxTime = \infty$, then no early stopping or state discards will be performed; the search will enumerate all operational states which will all be considered in the inclusion-exclusion and hence an exact result will be produced, the same as the basic inclusion-exclusion algorithm of Section 4.1. If $MaxTime < \infty$, then the worst case time to obtain the approximate result will be within that time limit. Without the $MaxTime$ limit, the performance of the inclusion-exclusion-stage of the heuristic algorithm is based on performance of the sort followed by inclusion-exclusion. Given, T is the set of states enumerated by the search phase, and $MaxStates < |T|$. Assume a standard sorting algorithm requiring $O(|T| \log |T|)$ iterations. The worst-case cost of the inclusion-exclusion, recomputing the unions and probability product each time, is $O(2^{MaxStates})$ to enumerate the k-subsets, times $n = |V(G)|$ subset unions, times $m = |E(G)|$ multiplications; $O(2^{MaxStates} * n * m)$.

In practice the value of the various thresholds will require adjustment to find a useful balance between no operational states being found and an excessive algorithm run time. In practice, without using the $MaxTime$ limit, based on the experimental results in Section 4.4, this discard-inclusion-exclusion algorithm provides a practical approximation of the HLTT metric for graphs of up to 1500 vertices.

4.4 Experimental Results

A JAVA implementation was created for the trust metrics presenting using the GRTM framework: HLTT, MCV, and MCVa. The following figures show the performance of the discard-inclusion-exclusion algorithm. All results are based on simulated trust graphs described in Appendix A. The parameters used for the powerlaw distribution were: α=0.7, β=0.8. The parameters used for the approximation heuristic were: $ProbabilityFloor = 0.3$, $MaxStates = 12$. The x-axis is the number of vertices in the test graph. The discard-inclusion-exclusion algorithm and the HLTT model demonstrate practical performance in processing trust graphs with scalable performance for over 1000 vertices. This meets the practical objective of handling hundreds of entities within one second. The approximate HLTT metric result was 99% accurate when compared to the exact result for simulated trust graphs of up to forty vertices. These results demonstrate that a practical application of GRTM+HLTT is possible from a computational perspective.

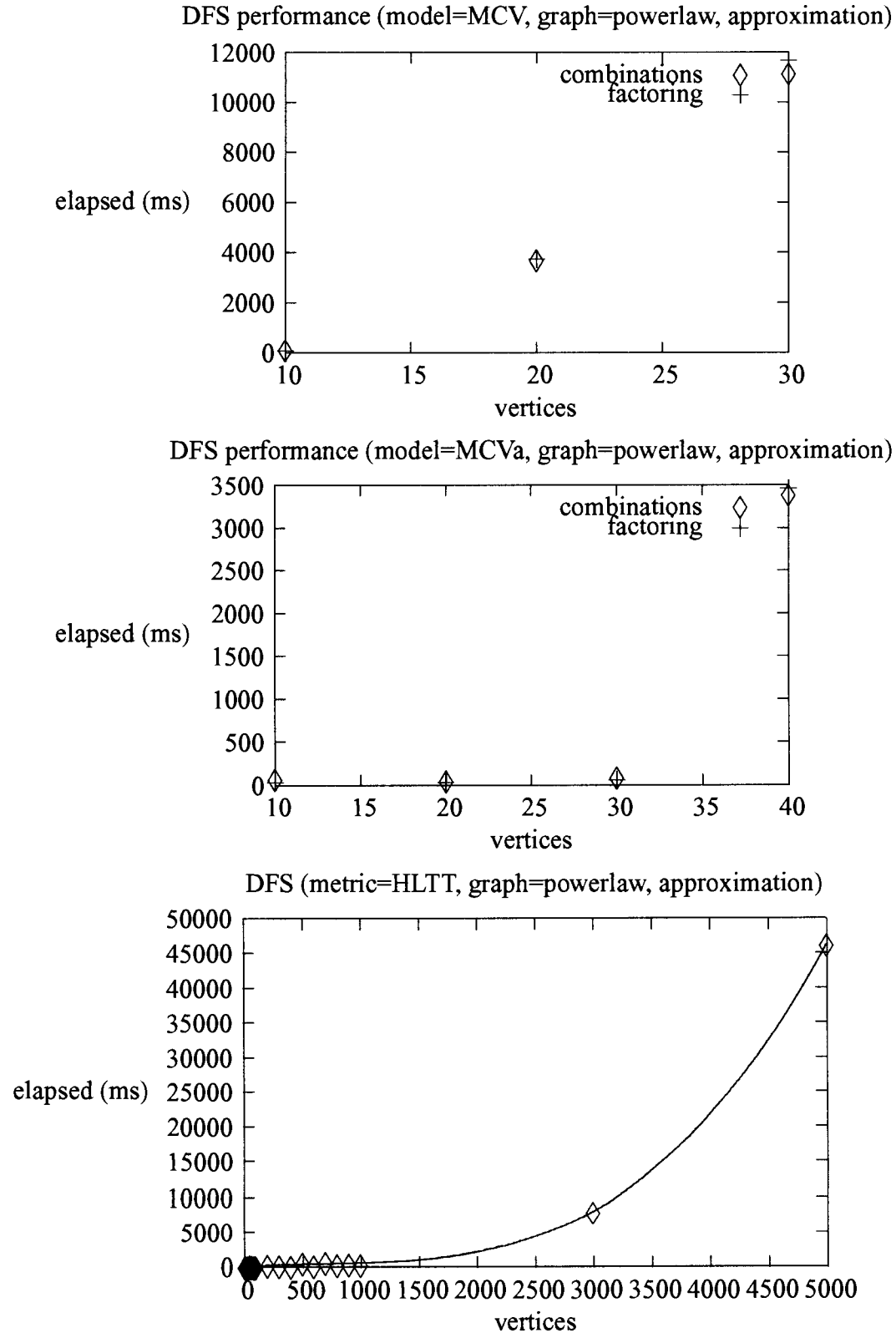


Figure 4.14. *Time of trust metric calculation using discard-inclusion-exclusion.*

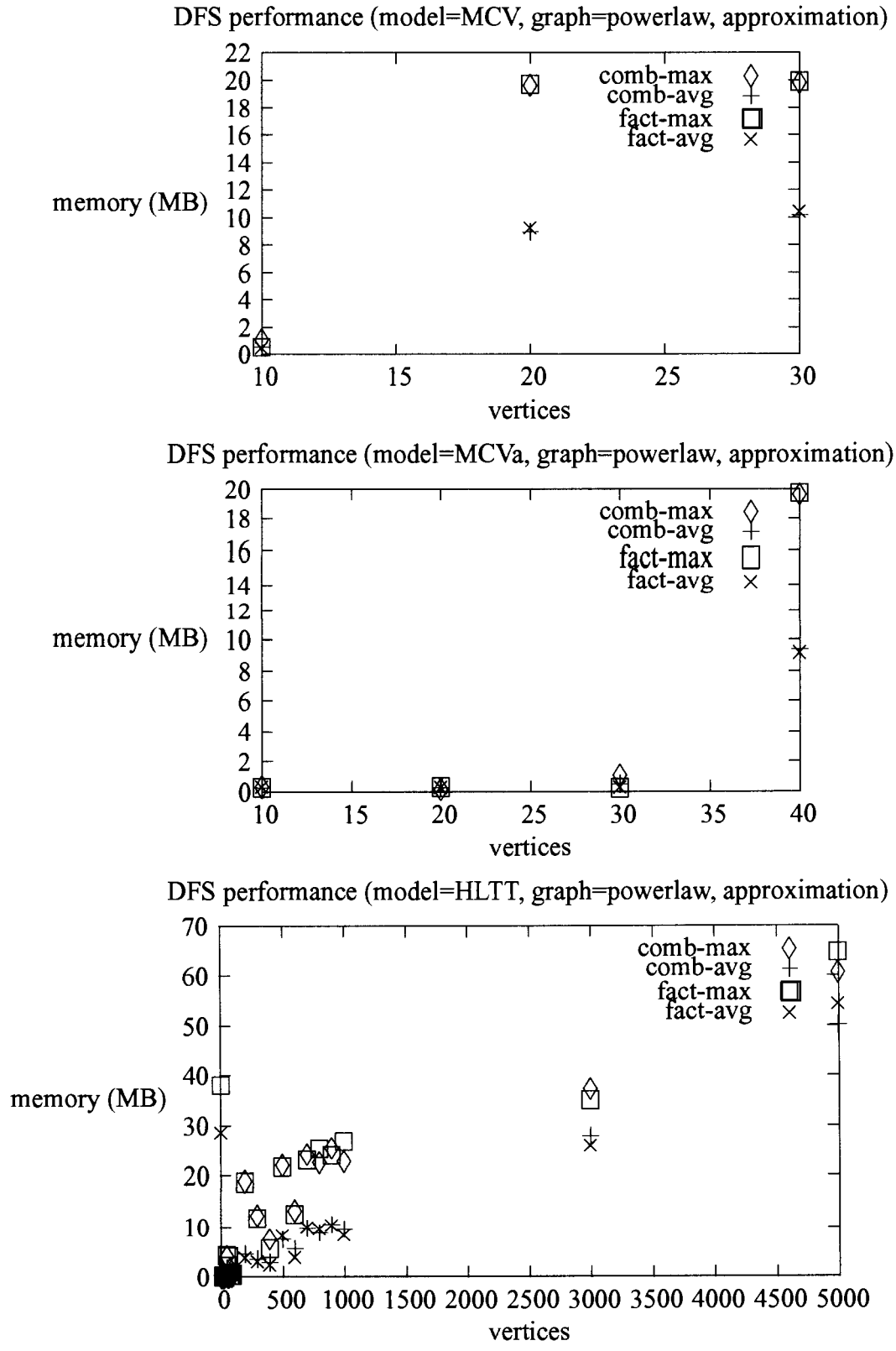


Figure 4.15. *Memory demands of trust metric calculation with discard-inclusion-exclusion.*

Chapter 5

Comparison and Related Work

This chapter compares the proposed Generic Reliability Trust Model (GRTM) and Hop-count Limited Transitive Trust (HLTT) metric with the trust models/metrics surveyed in Section 2.4 using concepts from the abstract model of Section 2.2. Table 5.1 provides a summary of GRTM/HLTT from this perspective. An important distinction between GRTM and all the other models is that GRTM is a general framework for specifying a trust metric; all the other models surveyed are, using the definitions introduced in this thesis, both a model and a specific metric, and as such are comparable to the combination of GRTM plus HLTT. Table 5.2 provides a summary of the differences of the proposed model/metric and the model/metrics of Section 2.4. A more detailed comparison of the Maurer Confidence Valuation is in Section 5.1.

The general approach used in this thesis of representing trust as a graph and implementing metrics through graph algorithms has been applied before to varying degrees. In Maurer's original MCV definitions, a graph is used to depict the various examples, but actual trust metric proposed processing was based on sets and logic. In Levien & Aiken[33], certificate-based trust is explicitly modelled as a graph, with keys as vertices, and two types of certificates as arcs. Various other trust models are transformed to this graph-based representation in order to evaluate their attack resistance.

The operational probability or confidence value associated with arcs of a trust graph (see Section 3.2) represents a much simpler version of trust compared to the example heuristic formalism of Marsh's computational trust[34]. At the same time it is a semantically richer

Table 5.1. *Trust Model Summary for GRTM*

Aspect	Description
roots	No trust roots assumed – could be configured.
entities	abstract entities; authors of trust beliefs.
subject-matter	not current specified.
direct	an entity-specific belief, represented as an arc in the trust graph.
indirect	transitive closure with respect to a particular trust metric's rule.
metric	reliability
implementations	JAVA-based implementation associated with this thesis.

notion than the existence of some message carrying path for network reliability. In general, the meaning of the trust value is based on the meaning of the associated subject-matter. This value could embody an aggregation of an entity's beliefs and assessments related to Marsh's basic, general, and situational trust, and include concepts such as risk and competence.

Table 5.2. *Comparison of GRTM/HLTT*

Model	Section	Qualitative Comparison
GRTM/HLTT	3	Generalized model and probabilistic trust metric. Currently not subject-matter specific.
X.509 PKI	2.4.1	Centralized model and chain-of-proof metric based on digital certificates. A restrictive subject-matter of identity authentication.
PGP	2.4.2	Decentralized model and chain-of-proof metric based on certificates. A restrictive subject-matter of public key authentication.
Trust Management	2.4.3	Some aspects of model decentralized and chain-of-proof metric with delegation. Subject-matter focus on access control.
Distributed Trust	2.4.4	Decentralized model with arithmetic trust metric and exchange of recommendations. Some subject-matter flexibility.
Network Flow	2.4.5	Generalized model with a network-flow metric test for chain-of-proof sufficiency.
Bayesian Network	2.4.6	Generalized model with an arithmetic trust metric. Supports adaption of different subject-matters using Bayesian networks.
Maurer Confidence Valuation	2.4.7	A somewhat generalized model of a certificate chain-of-proof system with a probabilistic trust metric. See Section 5.1.

Original: $Aut_{A,X}$, $Trust_{A,X,i}$, $Cert_{X,Y}$, $Rec_{X,Y,i}$

Simplfied: $S_{A,X,0}$, $S_{A,X,i}$, $S_{X,Y,0}$, $S_{X,Y,I}$

Assume associated statement confidences: c_1, c_2, c_3, and c_4

Trust Graph:

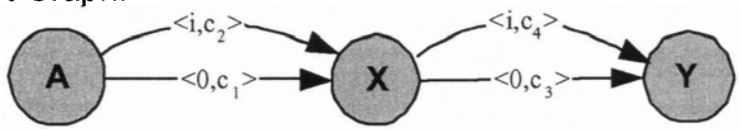


Figure 5.1. *Maurer Statements as a Trust Graph*

5.1 Comparison with MCV

Maurer's underlying probability model is the same as reliability; thus, the ease by which the MCV metrics have been represented and implemented using the GRTM framework. The computational algorithm suggested by Maurer is inclusion-exclusion using minimal subsets, the same as the exact algorithm for reliability presented in Section 4.1 [36, 31].

Trust graph arcs can be viewed as a generalization of the Maurer Confidence Valuation initial view statements about public keys, and a generalization of the simplified notation suggested in a remark by Maurer associated with the definition of this metric [36, Section 3]. There, using the entities A, X, and Y, and trust levels 0 and i, the Maurer statements $Aut_{A,X}, Trust_{A,X,i}, Cert_{X,Y}, Rec_{X,Y,i}$, with associated confidence values c_1, c_2, c_3, and c_4, could be represented as $S_{A,X,0}$, $S_{A,X,i}$, $S_{X,Y,0}$, $S_{X,Y,i}$ with the same associated confidence values. Either form of these statements can be translated into a GRTM trust graph with vertices A, X, and Y, and the arc set $\{ (A,X), (A,X), (X,Y), (X,Y) \}$ with the associated labels $< 0, c_1 >$, $< i, c_2 >$, $< 0, c_3 >$, and $< i, c_4 >$ – this is shown in Figure 5.1. Using the HLTT metric, the closest representation of this situation would be the arc set $\{(A,X),(X,Y)\}$ with the associated labels $< i, c_2 >$ and $< i, c_4 >$.

The HLTT metric is similar to MCV metric defined in [36]; this is seen more clearly in the representation of the MCV metric in Section 3.2.2 as a member of the new Generalized

Reliability Trust Model family of metrics. As a member of this family, the underlying probabilistic model of the MCV metric defined by Maurer is equivalent to the underlying generic reliability problem of the GRTM. Relative to GRTM/HLTT, MCV has a narrow application to PKI schemes, lacks computational algorithms, the transitivity rules are more complex, and MCV lacks the generalizing model of reliability.

Both versions of the MCV transitivity rules presented in Section 3.2.2 require explicit level zero arcs in any path from source s to target t before supporting the derivation of a new level zero statement $< s, t, 0 >$. Within MCV, this represents explicit separation of authentication and recommendation roles. This corresponds to the identity subject-matter domain requirement for a public key certificate to be signed by some entity, a kind of recommendation by the signing entity, before any other entity can authenticate the contained public key/identity binding. In this case, the authentication mechanism is based on establishing recommendation trust to the signing entity via a certificate chain. In HLTT, using the same scenario, a trust arc $e = (u, v)$ with trust level $l_e > 0$ assumes that u has authenticated v. This means the confidence and level values of an arc in the HLTT model represent the authoring entity's confidence with respect to both their direct trust and trust in the recommendations of other entities. Observe that, unlike with the HLTT rules, the level values in statements using the MCV-variant transitivity rules do not limit the number of hops to the target entity. An arbitrary length chain of level zero arcs is possible if there is support of multiple level one arcs, as shown in Figure 5.2.

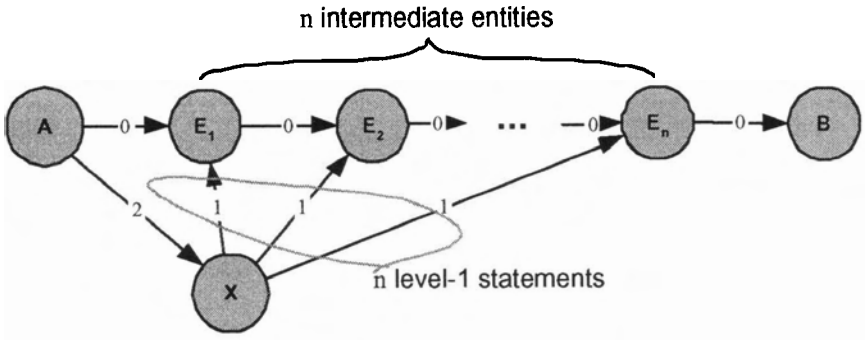


Figure 5.2. *Long chain of level-0 statements in MCV model.*

Chapter 6

Conclusions

This thesis contributes a new trust model and metric to the field of computational trust. The presented algorithms and implementation results support further research in the general field of computational trust and application of such trust mechanisms in systems and applications seeking to improve support for a range of value-based community interactions.

- A new generalized trust model based on a probability graph, Generic Reliability Trust Model (GRTM), has been created that can be used to flexibly represent trust beliefs between multiple parties.
- GRTM was based on a reliability model which supports the application of results from the more mature research area of Network Reliability.
- The generalized nature of GRTM was demonstrated through the definition of multiple metrics within this framework.
- A new trust metric, Hop Count Limited Transitive Trust (HLTT), was defined using the GRTM framework.
- Two variations of the Maurer Confidence Valuation (MCV) trust metric were defined using the GRTM framework.
- Two exact algorithms for generalized metric calculation were presented based on inclusion-exclusion and factoring.
- A new practical approximation heuristic was presented for evaluating GRTM trust metrics.

- An implementation of the exact and approximation algorithms was created in Java. Using this implementation, the approximation heuristic for the HLTT metric, without use of a hard time limit, produced a result in under one second on a simulated trust graph of up to 1000 entities.
- A randomized trust graph generator was created which simulates a large trust graph as a small-world/scale-free network and uses an XML/GXL representation.

6.1 Future Research and Potential Application

With additional research, the approximation heuristic could be enhanced to provide a practical result for thousands of entities. It is not intended that the algorithm be directly applicable to a trust graph of millions of entities as the eBay example in Section 1 implies. It is expected that any practical application will utilize only a subset of available information to form a trust graph for metric computation. This might be achieved by filtering for relevant subject-matter starting at the source entity.

Potential applications of GRTM, with or without HLTT, are manyfold; examples include use within improved introducer mechanisms for friend-of-a-friend (FOF) networks and its use as part of a dynamic authentication mechanism for ad-hoc networks. Generally, suitable applications are network-based applications and involve ad-hoc, somewhat informal interactions between potentially large sets of entities, which include some form of clustering. This can be thought of as the *mushy middle* between valueless interactions on one side and formal (e.g. contract-based) high-value interactions on the other. While application to larger notions of trust is outside of the scope of this thesis, as in [34] this computational model might be useful for modeling human or social-agent trust-based interaction.

The following items summarize future research which would broaden and deepen the theoretical and practical support for computational trust in general, and GRTM in particular.

Identify a Trust Model Quality Measure. Directly address the somewhat philosophical

question of the usefulness for representing trust through the proposed model and metric. Behind this question is a requirement for some general measure of the usefulness or quality of a trust model. Such a measure would enable better comparison of various models, and include representational power and computational complexity.

Develop the ability of GRTM to handle multiple subject-matters. A possible enhanced form of the arc label of trust graphs for use with GRTM which could represent subject-matter is the 3-tuple $< s, l, c >$ where,

s is a subject-matter string,

l is a level, and,

c is confidence.

This enhanced form could be considered further, including computing metrics using trust graphs containing multiple subject-matters, possibly using a Bayesian Network model (see Section 2.4.6).

Develop additional approximation techniques. The approximation heuristic in Section 4.3 should be enhanced to utilize the upper/lower bounds results of intermediate steps of inclusion-exclusion. Additionally, related results from Network Reliability could be applied, including bounding techniques [45].

Create a some form of standardized representation and exchange protocol. Review existing trust systems which include the ability to exchange trust information (recommendations) and similar mechanisms in reputation management systems. Clarify the unique requirements related to computational trust. Determine the suitability of existing protocols and consider the creation of a standardized trust statement (belief) representation and exchange protocol.

Extend GRTM to represent distrust.

Evaluate the attack resistance[33] of GRTM plus HLTT.

Apply GRTM-type solutions to trust establishment in Ad-Hoc Networks. For example, look at peer-to-peer (P2P) or Mobile Ad-hoc Networks (MANETs) [13].

Bibliography

[1] A. Abdul-Rahman and S. Hailes, "A distributed trust model," in *Proceedings of the 1997 Workshop on New Security Paradigms*. ACM Press, 1997, pp. 48–60.

[2] S. Baase and A. Gelder, *Computer Algorithms*, 3rd ed. Addison Wesley, 2000.

[3] A. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.

[4] M. Blaze, "Using the keynote trust management system," Matt Blaze Web Site, March 2001, accessed on June 20, 2004, URL: http://www.crypto.com/trustmgt/kn.html.

[5] J. Bondy and U. Murty, *Graph Theory with Applications*, Fifth printing (1982) ed. New York: Elsevier Science Publishing Co., Inc., 1976.

[6] C. Cachin, "Distributing trust on the internet," in *Proceeding of the International Conference on Dependable Systems and Networks, 2001*. Goteborg, Sweden: IEEE, 1-4 July 2001, pp. 183 – 192.

[7] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "Rfc2440: Openpgp message format," IETF Web Site, Internet Engingeering Task Force (IETF), Network Working Group, Nov 1998, accessed on June 1, 2004, URL: http://www.ietf.org/rfc/rfc2440.txt.

[8] L. Camp, H. Nissenbaum, and C. McGrath, "Trust: a collision of paradigms," in *Financial Cryptography. 5th International Conference, FC 2001. Proceedings*, ser. Lecture Notes in Computer Science, P. Syverson, Ed., vol. 2339, Program on Internet and Telecoms Convergence, Center for Technology, Policy, and Industrial Development (CTPID), MIT. Springer-Verlag, Germany, 2002, pp. 91–105.

[9] J. Carter and A. A. Ghorbani, "Value centric trust in multiagent systems," in *Proc. of the 2003 IEEE/WIC International Conference on Web Intelligence (WI 2003)*, Halifax, Nova Scotia, Oct 13-17 2003, pp. 3–9.

[10] CERT Coordination Center, Technical Cyber Security Alert, "CA-2001-04: Unauthentic Microsoft Corporation Certificates," March 21 2001, accessed on May 3, 2004, URL http://www.cert.org/advisories/CA-2001-04.html.

[11] B. Christianson and W. Harbison, "Why isn't trust transitive?" in *Security Protocols, International Workshop Proceedings*, ser. Lecture Notes in Computer Science,

T. M. A. Lomas, Ed., vol. 1189. Cambridge, United Kingdom, April 10-12, 1996: Springer, 1997, pp. 171–176.

[12] C. Colbourn, *The Combinatorics of Network Reliability*, J. Hopcroft, Ed. Oxford University Press, 1987.

[13] S. Corson and J. Macker, "RFC2501: Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," IETF Web Site, Internet Engingeering Task Force (IETF), Network Working Group, January 1999, accessed on October 20, 2004, URL: http://www.ietf.org/rfc/rfc2501.txt.

[14] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems, Concepts and Design*, 3rd ed., ser. International Computer Science, A. McGettric, Ed. Addison-Wesley, 2001.

[15] T. Dierks and C. Allen, "Rfc2246: The tls protocol, version 1.0," IETF Web Site, Internet Engingeering Task Force (IETF), Network Working Group, January 1999, accessed on June 1, 2004, URL: http://www.ietf.org/rfc/rfc2246.txt.

[16] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22(6), 1967.

[17] P. Dodds, R. Muhamada, and D. Watts, "An experimental study of search in global social networks," *Science*, vol. 301(5634), pp. 827–829, 2003.

[18] J. Douceur, "The Sybil Attack," in *1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, MIT Faculty Club, Cambridge, MA, USA, March 2002.

[19] eBay Inc., "2002 annual report," eBay Web Site, San Jose, California, 2002.

[20] eBay Inc., "Trust, safety, and privacy," eBay Web Site, San Jose, California, 2004.

[21] C. Ellison and B. Schneier, "Ten risks of pki: What you're not being told about public key infrastructure," *Computer Security Journal*, vol. 16(1), pp. 1–7, 2000.

[22] I. Frommer and G. Pundoor, "Small-worlds: A review of recent books," *Networks*, vol. 41(3), pp. 174 – 180, 2003.

[23] E. Gerck, "Toward real-world models of trust Reliance on received information," MCG.org Web Site, 1998, accessed on February 20, 2004, http://www.mcg.org.br/trustdef.htm.

[24] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Communications Surveys*, vol. 3(4), pp. 2–16, 2000.

[25] M. Granovetter, "The strength of weak ties," *American Journal of Sociology*, vol. 78(6), pp. 1360–1380, 1973.

[26] D. Harms, M. Kraetzl, C. Colbourn, and J. Devitt, *Network reliability: experiments with a symbolic algebra environment*. CRC Press, Inc., 1995.

[27] "Public-key infrastructure (x.509)," IETF Web Site, Internet Engineering Task Force (IETF), PKIX Working Group, accessed on June 1, 2004, URL: http://www.ietf.org/html.charters/pkix-charter.html.

[28] "Rfc2459: Internet x.509 public key infrastructure certificate and crl profile," IETF Web Site, Internet Engingeering Task Force (IETF), PKIX Working Group, January 1999, accessed on June 1, 2004, URL: http://www.ietf.org/rfc/rfc2459.txt.

[29] F. V. Jensen, *Introduction to Bayesian Networks*. Springer-Verlag, New York, 1996.

[30] J. Kleinfeld, "The small world problem (or could it be a big world after all? the "six degrees of spearation" myth)," *Society*, vol. 39(2), pp. 61–66, 2002.

[31] R. Kohlas and U. Maurer, "Confidence valuation in a public-key infrastructure based on uncertain evidence," *Public Key Crytography*, vol. 1751, pp. 93–112, 2000.

[32] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, July 1982.

[33] R. Levien and A. Aiken, "Attack-resistant trust metrics for public key certification," in *Proceedings of the Seventh USENIX Security Symposium*. San Antonio: USENIX Assoc., 1998, pp. 229–241.

[34] S. Marsh, "Formalising trust as a computational concept," Ph.D. dissertation, University of Stirling, Department of Computing Science and Mathematics, April 1994, 214p.

[35] J. L. Matt Blaze, Joan Feigenbaum, "Decentralized trust management," in *1996 IEEE Conference on Privacy and Security*, Oakland, CA, 1996.

[36] U. Maurer, "Modelling a public-key infrastructure," in *Proceedings of 1996 European Symposium on Research in Computer Security (ESORICS'96)*, E. Bertino, Ed., vol. 1146. Rome: Springer-Verlag, 1996, pp. 325–350.

[37] S. Milgram, "The small-world problem," *Psychology Today*, vol. 1, pp. 61–67, 1967.

[38] P. Mockapetris, "Rfc1034: Domain names - concepts and facilities," IETF Web Site, Internet Engingeering Task Force (IETF), Network Working Group, Nov 1987, accessed on August 11, 2004, URL: http://www.ietf.org/rfc/rfc1034.txt.

[39] *Special Publication 800-12: An Introduction to Computer Security: The NIST Handbook*, National Institude of Standards and Technology, U.S. Department of Commerce, October 1995, uRL: http://csrc.nist.gov/publications/nistpubs/800-12/.

[40] "Introduction to cryptography, pgp, version 7.0," PGP International Web Site, Network Associates Inc., 2001, accessed on June 1, 2004, URL: ftp://ftp.pgpi.org/pub/pgp/7.0/docs/english/IntroToCrypto.pdf.

[41] "Pgp freeware for windows 95, windows 98, windows nt, windows 2000 & windows millennium, user's guide, version 7.0," PGI International Web Site, Network Associates Inc., 2001, accessed on June 1, 2004, URL: ftp://ftp.pgpi.org/pub/pgp/7.0/docs/english/PGPWinUsersGuide.pdf.

[42] C. Palmer and J. Steffan, "Generating network topologies that obey power laws," in *Proceedings of the Global Internet Symposium, IEEE Globecom2000.* San Francisco: IEEE, 2000, pp. 434–438.

[43] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Emperical analysis of ebay's reputation system," *Advances in Applied Microeconomics*, vol. 11, pp. 127–157, 2002.

[44] P. Resnick, R. Zeckhauser, R. Friedman, and K. K., "Reputation systems," *Communications of the ACM*, vol. 43(12), pp. 45–48, 2000.

[45] D. Shier, *Network reliability and algebraic structures.* Oxford University Press, 1991.

[46] Y. Wang and J. Vassileva, "Bayesian network-based trust model," in *Proceedings of IEEE/WIC International Conference on Web Intelligence, 2003 (WI 2003)*, Halifax, Canada, October 13-17 2003, pp. 372– 378.

[47] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–42, 1998.

[48] P. Zimmermann, "Why OpenPGPs PKI is better than an X.509 PKI," OpenPGP Web Site, Feb 2001, accessed on June 1, 2004, URL: http://www.openpgp.org/technical/whybetter.shtml.

Appendix A

Trust Graph Simulation

This appendix describes the underlying model of the simulated trust graphs used to test the various algorithms presented in this thesis.

The concept of six-degrees of separation emerged from a 1967 social experiment to look at underlying model of social connectedness behind the phrase "My, it's a small world" [37]. The experiment involved a booklet being sent between two random individuals using acquaintances. The result was the at-the-time counter-intuitive observation that it required on the average only five intermediaries (or size hops) to link any two randomly chosen individuals in the United States. Even though the specific metric result of five is not well supported by the evidence, popular interest was generated in the small-world concept [30]. The social science importance of the small-world model was solidified by later research which further applied graph theory in looking at the importance of weak ties between bridge individuals in the diffusion of messages in an acquaintance network, and then a generalized mathematical model and experimental verification with email within small-world networks [25, 47, 17]. Given that trust can be viewed as a kind of social relationship between individuals, it is reasonable to assume larger-scale networks made up of these relationships will follow such a small-world model.

In a *scale-free network*, the probability $P(k)$ that a vertex in the network interacts with (has a link to) exactly k other vertices decays as a power law [3], with the following general form:

$$P(degree = k) \sim k^{-\beta} \tag{A.1}$$

A scale-free network is similar to a small-world network; there is typically a short distance between any two vertices, there exist short-cuts (or bridges) between clusters of vertices, and there is exponential decay in the distribution of vertex degree. A scale-free network is different from a small-world network in that it has a few vertices with high degree, called hubs [22]. The scale-free network model has been shown to be useful in modelling many complex networks, from representing hyperlinks between documents on the Web, to the graph of collaborations between movie actors [3, 42, 47].

It is the combination of representing small-world behaviour and the presence of hubs that makes the scale-free model useful for simulating trust graphs. The concept of a hub represents a special kind of trusted agency or a trust root entity serving as the broker for a large number of trust relationships. Examples of this type of entity include a PKI Certificate Authority or film-reviewer.

A.1 Power-Law Random Graph Generation

A *power law random graph* (PLRG) is a randomly generated graph in which the degrees of the vertices follow a power law distribution. The characteristic equation of a PLRG is as follows:

$$P(\text{degree=k}) \sim \alpha k^{-\beta} \tag{A.2}$$

The following is pseudo code for generating a PLRG is adapted from the PLOD (power law out degree) algorithm from [42]. The inverse power law distribution of the out-degree of vertices produced by this algorithm can be observed in Figure A.2, with inputs $n = 1000$, α=0.7, β=0.8. The effect of α and β can be observed in Figure A.3. The lower values of β produce the hub or super-node effect characteristic of scale-free networks.

```
1    n is the required number of vertices.
2    α – see Equation A.2.
3    β – see Equation A.2.
4    adjacent is an n x n binary matrix, initially all zero.
5    outDegree is an integer array of size n.

6    Function genPowerLawAdjacency(n,α,β): return adjacency matrix
7      totalArcs = 0;
8      for i = 1 to n;
9        x = random number from 1 to n − 1;
10       outDegree[i] = α ∗ n ∗ x^−β;
11       if outDegree[i] > (n − 1) then odegree[i] = n − 1;
12       totalArcs+ = outDegree[i];
13     end-for;
14     assignedArcs=0;
15     while (assignedArcs < totalArcs)
16       u = random number from 1 to n;
17       v = random number from 1 to n;
18       if u ≠ v and outDegree[u] > 0 and not adjacent[u][v]
19         adjacent[u][v] = 1;
20         outDegree[u] − −;
21         assignedArcs + +;
22       end-if;
23     end-while
24   return adjacent;
```

Figure A.1. *Pseudo-code for generating graph adjacency with a powerlaw out-degree.*

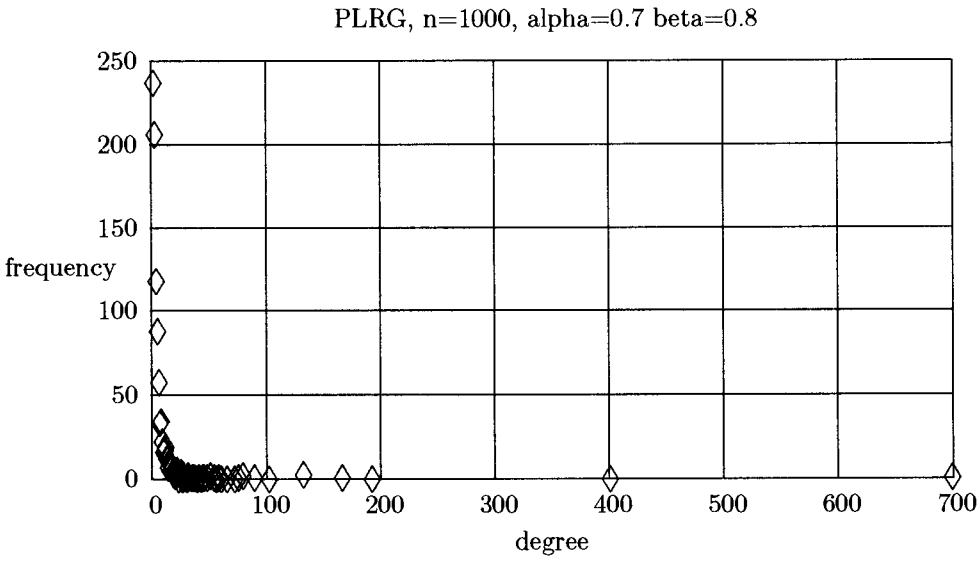


Figure A.2. *Scale-free out-degree distribution*

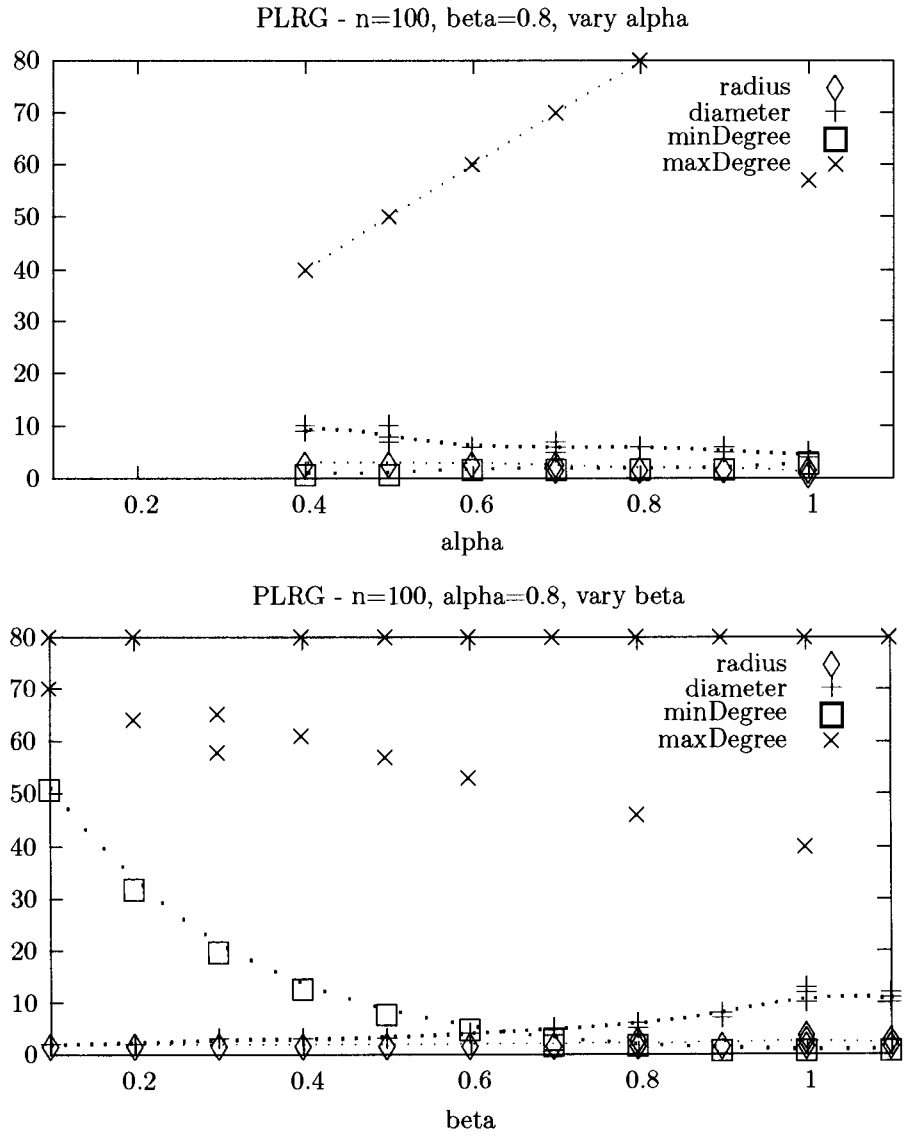


Figure A.3. *Effect of varying α and β on PLRG out-degree distribution*

A.2 Test Graph Generation

The scale-free trust graphs used here are restricted to only those that are connected (there is a path between all vertex pairs). This is an aspect of the simulated data that is similar to the original small-world social experiment in [37]; the graphs produced by the generator are not always connected, and in the experiment some of the chains of people could not find their target. The connected graphs used for this thesis were produced by simply repeating the random generation process if the result was not a connected graph. As observed in [37] and later in [30], this will tend to produce an over-optimistic view of the connectedness of social networks. As the focus of this thesis is on establishing basic ability to compute results, rather than on direct application to social situations, the use of connected graphs does not invalidate the results. If graphs with multiple connected components were to be used, there would be vertex-pairs for which no trust can be discovered.

The general process used to generate the test graphs used for this thesis is as follows:

1. PLRG graphs were generated with the parameters $\alpha = 0.7$, $\beta = 0.8$ with orders $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000\}$
2. General graph measures were calculated for each of these random graphs. This includes determining the center vertices (representing hubs), which were not used as a source or target vertex for testing.
3. Three versions of each of these basic graphs were created, one for each of the the metrics {MCV,MCVs,HLTT}, using arc probabilities of 0.8 and trust level of four. This ensured that the same underlying connectivity exists between any vertex pair in all three graph versions. Note: this does necessary mean the same trust exists between a given vertex pair under the different metrics due to differences in their transitivity rule.

A.2.1 Graph Representation using XML

A common trust graph representation was created based on the eXtensible Markup Language (XML) and Graph Exchange Language (GXL, see xml.coverpages.org/gxl.html). The following figure shows a portion of the file contents of a generated trust graph represented in this format.

```
< gxl >
 < graph id = "PowerLaw_10_09:31:48 (MCVs)" >
  < attr name = "Model" >
   < string >MCVs< /string >
  < /attr >
  < attr name = "Note" >
   < string >
Power law random graph, size 10, alph=0.7, beta=0.8, maxLevel=4, fixed conf=0.8, gen-
erated Thu Apr 22 09:31:48 PDT 2004 by models.algo.GraphGen
   < /string >
  < /attr >
  < node id = "V3"/ >
  < node id = "V10"/ >
  < node id = "V5"/ >
  < node id = "V8"/ >
  < node id = "V4"/ >
  < node id = "V6"/ >
  < node id = "V1"/ >
  < node id = "V9"/ >
  < node id = "V7"/ >
  < node id = "V2"/ >

  < edge from = "V1"to = "V3" >
   < attr name = "Level" >
    < int > 4 < /int >
   < /attr >
   < attr name = "Confidence" >
    < float > 0.8 < /float >
   < /attr >
  < /edge >

 ...

 < /graph >
< /gxl >
```

Figure A.4. *XML representation of a trust Graph.*