DYNAMIC AND COMPOSABLE TRUST FOR INDIRECT INTERACTIONS

By

IOANNA DIONYSIOU

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

AUGUST 2006

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of IOANNA DIONYSIOU find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

_____

_____

ACKNOWLEDGEMENT

What a journey this has been! First, I would like to thank my advisor, Dave Bakken, for his guidance and support throughout my years in WSU. This work would not have been possible without the confidence and enthusiasm provided by him. I'm particularly grateful that he allowed me to name my research $E\sigma\tau\iota\alpha$ to honor my Greek-Cypriot heritage. I would also like to thank Carl Hauser for our discussions. I am also grateful that Deb Frincke accepted the invitation to serve as one of my committee members and provided valuable feedback. I was also lucky and honored to have Anjan Bose in my committee. Last, but not least, I extend my gratitude to John Shovic for giving me the opportunity to start my PhD studies as his research assistant.

Other faculty and staff at WSU have also been of great help to me. Ruby Young offered her assistance to me over the last 10 years. Shira Broschat not only supported me from the very beginning of my studies but she also trusted me with a teaching fellowship. Jack Hagemeister deserves a kind word of thanks as well.

I was lucky to meet and interact with great students while at WSU. A special thanks goes to all of them for making my stay in Pullman enjoyable.

I would also like to thank the organizations that have funded me throughout my studies. My research has been supported in part by the US Department of Commerce, National Institute of Standards and Technology Grant 60NANB1D0016 (Critical Infrastructure Protection Program), in a subcontract to Schweitzer Engineering Labs Inc, and by the National Science Foundation under Grant CCR-0326006 and Grant NSF CNS 05-24695 (CT-CS: Trustworthy Cyber Infrastructure for the Power Grid). In addition, I received teaching assistantships as well as a teaching fellowship from the the School of Electrical Engineering and Computer Science at WSU.

Finally, this accomplishment would not have been a success if it wasn't for my wonderful parents, Omiros and Chryso, my lovely sister Marilena, and my dear husband Harald. I thank you from the bottom of my heart.

ATTRIBUTION

Portions of this dissertation were published in the following conferences or workshops.

- Carl H. Hauser, David E. Bakken, Ioanna Dionysiou, K. Harald Gjermundrød, Venkata S. Irava, Joel Helkey, and Anjan Bose, Security, trust and QoS in next-generation control and communication for large power systems, *International Journal of Critical Infrastructures*, Inderscience, to appear, 2007.

- Carl H. Hauser, David E. Bakken, Ioanna Dionysiou, K. Harald Gjermundrød, Venkata S. Irava and Anjan Bose, Security, trust, and QoS in next-generation control and communication for large power systems, in *Proceedings of the Workshop on Complex Network and Infrastructure Protection (CNIP06)*, Rome 28-29 March, 2006.

- Drugan, O., Dionysiou, I., Bakken, D., Plagemann, T., Hauser, C., and Frincke, D., On the Importance of Composability of Ad Hoc Mobile Middleware and Trust Management, Lecture Notes in Computer Science 3694, Springer, 2005.

- Dave E. Bakken, Ovidiu-Valentin Drugan, Ioanna Dionysiou, Thomas Plagemann, Deborah Frincke, and Carl Hauser, Composability of Ad Hoc Mobile Middleware and Trust Management for Critical Infrastructures, *Service Availability, Second International Service Availability Symposium*, ISAS 2005, Berlin, Germany, April 25-26, 2005.

- Harald Gjermundrød, Ioanna Dionysiou, David Bakken, and Carl Hauser, Fault Tolerance Mechanisms in Status Dissemination Middleware, in *Supplement of the International Conference on Dependable Systems and Networks (DSN-2002)*, IEEE/IFIP, San Francisco, CA, June 2003, B-56–57.

- D. Bakken, A. Bose, C. Hauser, I. Dionysiou, H. Gjermundrød, L. Xu, and S. Bhowmik, Towards More Extensible and Resilient Real-Time Information Dissemination for the

Electric Power Grid, in *Proceedings of Power Systems and Communications Systems for the Future, International Institute for Critical Infrastructures*, Beijing, China September 2002.

- I. Dionysiou, H. Gjermundrød, and D. Bakken, Fault Tolerance Issues in Publish-Subscribe Status Dissemination Middleware for the Electric Power Grid, in *Supplement of the International Conference on Dependable Systems and Networks (DSN-2002), IEEE/IFIP*, Washington, DC, June 23–26, 2002, B-62–63.

- D. Bakken, A. Bose, C. Dyreson, S. Bhowmik, I. Dionysiou, H. Gjermundrød, and Lin Xu, Impediments to Survivability of the Electric Power Grid and Some Collaborative EE-CS Research Issues to Solve Them, in *Proceedings of the fourth Information Survivability Workshop (ISW-2001/2002), Impediments to Achieving Survivable Systems*, Vancouver, BC, Canada, March 2002.

Submitted for Journal Publication

- Ioanna Dionysiou, Deborah Frincke, Carl H. Hauser, and David E. Bakken, Dynamic and Composable Trust for Indirect Interactions, submitted for publication at Elsevier Science.

DYNAMIC AND COMPOSABLE TRUST FOR INDIRECT INTERACTIONS

Abstract

by Ioanna Dionysiou, Ph.D.
Washington State University
August 2006

Chair: David E. Bakken

The diversity of the kinds of interactions between principals in distributed computing systems has expanded rapidly in recent years. However, the state of the art in trust management is not yet sufficient to support this diversity of interactions. This dissertation presents a rationale and design for much richer trust management than it is possible today. To do so, it presents a set of requirements for more generalized trust management, an analysis of why they are needed, and how the state of the art in trust management to date does not meet them. It then presents the design of Hestia, our trust management framework. Hestia supports dynamic trust, which enables the specification and management of trust relationships which can be re-evaluated over the lifetime of a relationship. Hestia also supports the composition of trust relationships, which supports indirect interactions between principals as is the case with publish-subscribe systems. Finally, Hestia handles trust for both access control purposes and for reasoning about the quality of (possibly aggregated) data provided by a set of principals. This dissertation also presents formalisms for dynamic and composable trust.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Dedication

This thesis is dedicated to $O\mu\eta\rho o\varsigma$, $X\rho\upsilon\sigma\omega$, and $M\alpha\rho\iota\lambda\epsilon\nu\alpha$.

CHAPTER 1

INTRODUCTION

In the last decade or two distributed computing systems have gone from being largely laboratory curiosities with few wide-area deployment to becoming almost ubiquitous in scope. They are now being used in many ways by a wide spectrum of society in a diverse range of topologies, computations, and services. Indeed, many individuals routinely use dozens of such applications, and virtually all corporations and governments use and provide a multitude of such applications in their supply chain, business-to-business, online government, etc. Similarly, individuals use distributed services for a variety of entertainment, social, financial, and logistical purposes.

Ubiquitous use of distributed applications provides increased convenience, safety, and enjoyment for society. However, such applications and their users are vulnerable with respect to both the diversity of the principals providing these services or data and the interactions between them. In particular, they have no systematic and comprehensive way to reason about the following issues:

1. How to specify and adapt the degree of trust that they place in an entity
2. How much trust to place in data they receive that comes through nontrivial chains of processing or services
3. How to decide how much access to their services to provide other principals.

We call this the *generalized trust* problem, and it is almost completely untouched to date, as detailed in Chapter 7. This problem is hard for general wide-area distributed systems but even harder for critical infrastructures, such as the electric power grid, due to their scale and the threats to which they are subjected.

This dissertation presents a configurable trust management system for dynamic and composable trust for indirect interactions, which are chains of processing of data that goes through multiple

principals. This novel system defines and provides an initial set of solutions for the first two issues of the generalized trust problem, as listed earlier. In order to understand the context of the new approach, a discussion of why the current state of the art in computer security alone can not solve the generalized trust problem is presented first. Other mechanisms must be used to reason about generalized trust and these mechanisms are the focus of the next subsection. To be more specific, a brief overview of the current state of the art in trust management systems is presented and an example illustrates the lack of existing trust management systems to provide comprehensive solutions for indirect interactions. This chapter concludes with the thesis and contributions of the dissertation work, followed by an outline of the remaining chapters.

1.1 Security Services and Mechanisms

Internet security consists of measures to deter, prevent, detect, and correct security violations that involve the transmission of information. Security violations, also known as security attacks, include interruption, interception, modification, and fabrication of an information flow [41]. Security services such as confidentiality, integrity, and authentication enhance the security of the transmitted information by safeguarding the data against these security attacks. Various security mechanisms exist that provide these services, including encryption, digital signatures, authentication protocols, just to name a few. The difference between security services and security mechanisms is in essence a difference between "what" versus "how".

However, cryptographic algorithms and access control schemes cannot be used to reason about and manage the more general concept of data quality. Consider the case where a recipient of a message would like to evaluate whether or not the enclosed information is *authentic* (authentic is defined as having a genuine origin, in opposition to that which is false, fictitious, and counterfeit.) and credible, knowing the following three facts: First, a cryptographic checksum was computed over the message content, second, it was signed by the sender's private key and attached to the original message, and third, the entire message was encrypted with the receiver's public key. Integrity

techniques, such as checksums, can verify that message contents are not altered during transmission but can not guarantee the sender's ability to generate correct and reliable data. Authentication protocols verify the sender's identity, however this is not adequate to reason about the data quality. Finally, confidentiality schemes allow only legitimate users to view the message contents, but provide no other assurances regarding the content itself.

As a result, security services are not adequate to solve the generalized trust problem without be enhanced or complemented by other mechanisms and techniques.

1.2 Trust Management Systems

Related to the above classical security services is the notion of trust. Trust is an abstraction of individual beliefs that an entity has for specific situations and interactions. Creating a universally acceptable set of rules and mechanisms to specify trust is a difficult process because of the variety in trust definitions. In recent years, researchers have investigated various definitions of trust for many perspectives, with the result that trust definitions overlap or contradict each other [3].

Trust is useful only if it is managed in an appropriate and systematic manner. An entity's beliefs are not static but they change as time progresses and new information is processed into knowledge. Trust must evolve in a consistent manner so that it still abstracts the entity's beliefs accurately. In this way, an entity continuously makes informed decisions based on its current beliefs.

Trust management concerns itself with four primary tasks that are involved in updating and maintaining trust relationships: collecting evidence to make a trust decision, analyzing existing trust relationships to infer new ones, evaluating criteria related to trust relationships, and monitoring as well as re-evaluating trust relationships. There are numerous models of trust management [45, 39, 47, 30, 28, 27, 12, 34, 15, 42, 11, 49], although no rigorous classification of either trust or its models has been developed yet. For example, there are trust management systems that focus on representing specific aspects of trust, such as authentication, reputation, and cooperation whereas there are others that manage more general trust relationships.

Trust management systems (TMSs) to date support characterization of the quality of data or services coming from external sources. However, such research does not yet support the full spectrum of trust requirements for highly interdependent applications, as we demonstrate in Chapter 2. For example, trust management systems either lack automatic dynamic re-evaluation of the level of trust an entity has in the data or services or provide limited re-evaluation of trust relationships. As a result, the trust levels are established once even if changing conditions merit increasing or decreasing the trust one places in a given application or data element.

Additionally, trust management systems focus on specification and evaluation of data or services that are directly received by another principal. They do not allow the specification and evaluation of trust involving indirect interactions. In other words, TMSs do not support the property we call *trust composition*. Because of this, existing TMSs cannot directly deal with topologies such as publish-subscribe (where multiple principals may forward or filter data) or with data aggregation involving data from multiple principals or chains of processing of data passing through several principals.

In order to illustrate the requirement of trust composition, consider a data stream that originates at entity A, traverses intermediary entities B, C and reaches destination entity D. Current state-of-the-art trust management mechanisms allow peer D to reason about the quality of the received data by establishing a trust relationship with entity A, something that is inadequate to handle the case of malicious or malfunctioning intermediary entities such as B and C that affect that data as well. End-to-end trust assessment is needed to characterize the data quality by deriving trust assessments for all interacting parties that collaboratively execute the task of data transfers. We note that the reasoning behind the need for trust composition in publish-subscribe topologies also applies to aggregated data that is derived by multiple principals.

In summary, trust management today largely provides for static, pairwise relationships involving two principals. However, it is a basis for providing more generalized form of trust. Indeed, the U.S has founded in September 2001 the Institute for Information Infrastructure Protection (I3P), a

consortium of academic research centers, government laboratories and non-profit organizations, to identify and address critical research problems in information infrastructure protection [2]. Generalized trust is important to several aspects of the I3P research agenda [32], including "Trust Among Distributed Autonomous Parties" (TADAP) and "Enterprise Security Management" (ESM). Furthermore, generalized trust is important for critical infrastructures, such as the power grid. For example, GridStat [38, 25, 48, 43, 29], a data dissemination middleware which is developed to provide better communications services for the electric power grid, must provide guarantees to the thousands of grid participants that the information is trustworthy.

1.3 Thesis Statement

As of today, there is no comprehensive definition that covers the semantics of end-to-end trust in indirect interaction communication models. Furthermore, trust management is not flexible enough in a number of ways to meet the trust needs of indirect interactions. A trust management framework for indirect interactions must cover a broader scope of trust. It must enable the specification and management of trust relationships to change over the operational lifecycle of the system as relevant conditions that affect trust change. It must also identify and compose trust relationships in a way that allows an entity to reason about end-to-end trust.

It is the thesis of this research that trust management techniques can be extended to meet the trust needs of architectures such as publish-subscribe that support indirect interactions between principals. The primary contribution of this research is a flexible and broad trust management framework allowing diverse trust requirements, including security requirements, to be reasoned about, specified, and managed. The four components of a generalized trust management system must be supported for any such trust management framework that is designed to meet the security and related non-functional requirements of indirect interactions.

Thesis: Trust management techniques can be extended to meet the trust needs of architectures such as publish-subscribe that support indirect interactions between principals.

An over-arching contribution of this research is to define issues, concepts, paradigms, and mechanisms that provide this extended trust management framework, and then to present an initial set of solutions to these issues. In particular, this dissertation presents the following research contributions:

- An analysis of why dynamic and composable trust are both needed in many distributed application programs today which span multiple principals, particularly ones involving data aggregation or indirect interactions, such as publish-subscribe applications
- A set of requirements which must be met by any trust management system in order to solve the generalized trust problem
- New trust paradigms which support the implementation of these requirements
- Hestia[1], a trust management system which supports dynamic and composable trust for collaborative environments through configurable trust policies. It provides for:
  - dynamic trust management, including time-aware trust relationships
  - trust composition, in order to support indirect interactions and data aggregation
  - evaluation of both *access trust*, generalized policies for access control, and *data trust*, systematic reasoning about the quality of (possibly aggregated) data provided by a set of principals
- Trust formalisms, including trust ontology, properties, and operations.

[1]Hestia ($E\sigma\tau\iota\alpha$) is the name of a Greek goddess who was the estate manager for Zeus and was trusted with the keys to the family home on Mount Olympus. Her virtues included being secure and stable and remaining above the fray rather than taking part in the struggles of men or gods.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows: Chapter 2 describes three motivating examples: a set of application suites for the electric power grid that the electricity industry uses today or has plans to do so. These applications have a diverse set of trust requirements but all suffer from the generalized trust problem and could greatly benefit from dynamic and composable trust. In addition, trust-modulated transparency is presented, which is one of the future internet technologies. Chapter 3 describes the requirements for any trust management system which intends to solve the generalized trust problem. Chapter 4 presents the Hestia trust management system and its operation. Chapter 5 discusses the Hestia policy space. Chapter 6 presents trust formalisms, including its ontology, properties and operations. Chapter 7 overviews related trust management research and compares other trust management systems against the requirements from Chapter 3. Chapter 8 concludes and discusses future research directions for Hestia.

CHAPTER 2

CRITICAL INFRASTRUCTURES AND TRUST MANAGEMENT

The protection of critical infrastructures is an essential element of ensuring a nation's security [4]. In order to do that, private, public and federal entities must collaborate in a way that valuable information is shared without compromising it. This collaborative environment is very difficult to establish and operate because, given the state of art in trust management as well as current security mechanisms, its participants do not have the necessary knowledge and tools to assess the quality of the received data and the risk of compromising that data.

Consider the North American electric power grid, for instance, with nearly 3500 utility organizations [23]. These individually owned utility systems have been connected together to form interconnected power grids, which must be operated in a coordinated manner. There are many points of interactions among a variety of participants and a local change can have immediate impact everywhere. In order to detect disturbances that could escalate into cascading outages and take corrective actions, real-time information about the grid dynamics must be obtained to enhance the wide-area system observability, efficiency, and reliability. For example, the Electric Power Research Institute (EPRI), a consortium of academic and industrial experts that work collaboratively on approaches to electric power challenges, created the IntelliGrid conceptual architecture that envisions a next generation power system that consists of automated transmission and distribution systems that support efficient and reliable supply and delivery of power [21]. IntelliGrid assumes, but does not provide, an appropriate QoS managed middleware (for example, like GridStat) that is used to support the framework's mechanisms for data retrieval from field equipment and issue control commands to power system equipment, among field devices, between field devices and systems located in substations, and between field devices and various systems including SCADA, utility control centers, engineering, and planning centers. Data can be real-time data, statistical

data, or other calculated data and informational data from the power system to systems and applications that use the data.

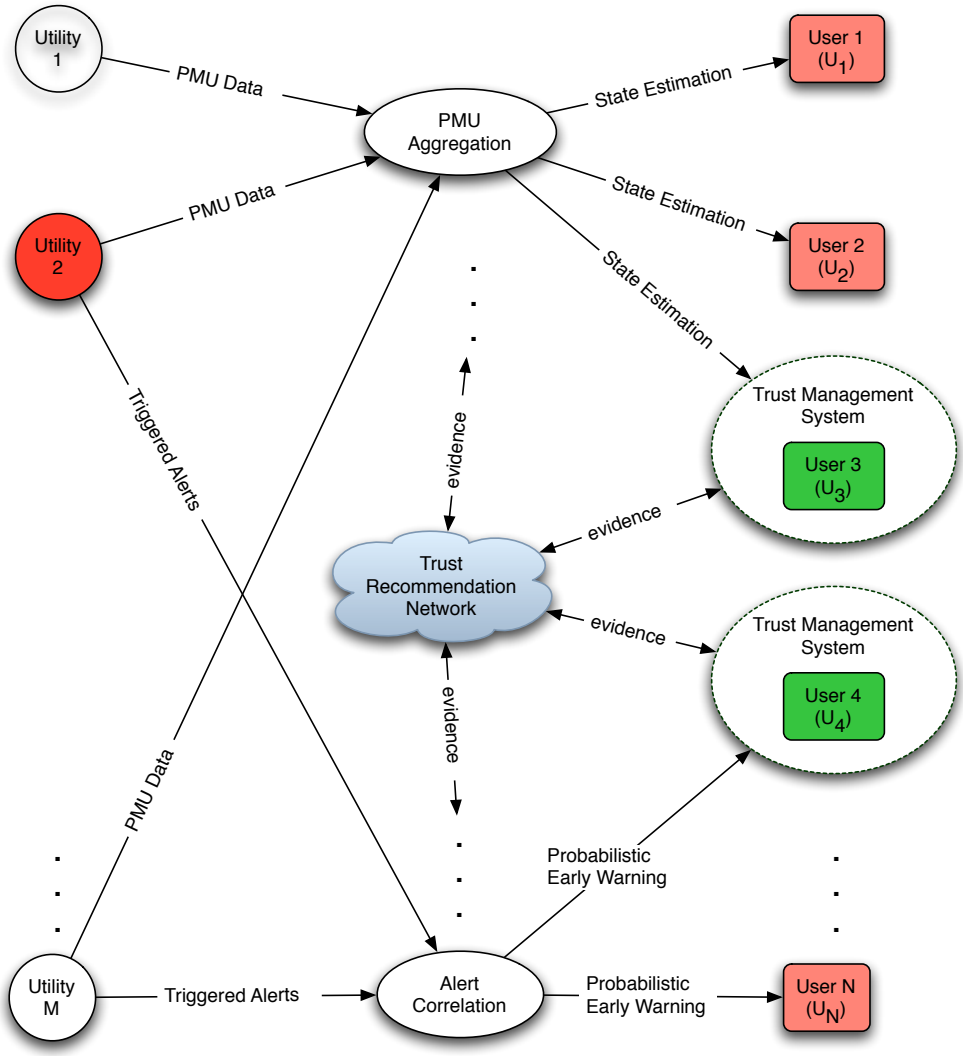

Figure 2.1: Power Grid Applications

Unfortunately, as of today, power utilities are reluctant to disclose information in order to protect themselves financially and legally. Sharing of data might jeopardize their business due to their inability to quantify the risk regarding interactions with other grid participants. For example, unrestricted access to a utility's data that are market-sensitive indicators could give a competitor

an unfair advantage in adjusting its own contracts and prices. Similarly, a utility could distribute inaccurate data to mislead the other market participants. This trend is not only observed in the power industry but it rather concerns other markets as well; according to the recent CSI/FBI report [18], companies are reluctant to report security incidents because of the potential loss of revenue.

The "no sharing" policy could be relaxed under normal operating conditions if the risk of sharing were systematically contained, and could be relaxed much more if needed to help avert a looming crisis. Trust management is a service that, when used properly, has the potential to enable the entities that have knowledge about critical infrastructures to share confidential, proprietary, and business sensitive information with reliable partners.

In order to illustrate how trust management is related to the dissemination of data, two power grid application suites are considered (Figure 2.1). We target the power grid because it is more complex and harder than the other infrastructures, mainly due to its geographical scale and real-time requirements. The application domain setting involves M utility organizations that generate data and N end users that receive computation results on that data. Both applications span a number of regional utility districts with different administrative policies and ownership. The *Phasor Measurement Unit (PMU) Aggregation* application involves the dissemination and aggregation of PMU data, and the *Alert Correlation* application provides (probabilistic and imperfect) early warning of an impending power crisis. In addition to the power grid application families, we also discuss a family of applications from current network research which uses trust assessment at the packet level.

We now overview these three application suites, but prior the discussion we note a couple of simplifications made in the two power grid application suites. First, without loss of generality, the trust management system is placed at the end users U_3, U_4. However, user-customized aggregation or correlation policies that conform to the user's trust policies, could also be loaded at the entity that performs the operation (e.g. PMU Aggregation entity, Alert Correlation entity). Second, the two applications disseminate the results of their processing to entities other than the original sources

of data. In an actual deployment, feedback could (and likely would) be provided to the original utilities as well.

2.1 PMU Data Aggregation

The first application suite deals with a type of real-time electric power data, which is PMU data. PMUs are instruments that take measurements of voltages and currents and time-stamp these measurements with high precision. They are able to measure phase difference at different substations and have been implemented as a source of information to detect faults on transmission lines. These measurements are collected and aggregated at a central place in order to derive system state estimations. In order to preserve correlation of these readings, it is important to temporally synchronize the PMU measurements during aggregation [37, 33]. These estimations are disseminated to interested end users U_i, including entities that monitor or control the grid, such as Independent System Operators (ISOs), Regional Transmission Operators (RTOs) and, in the future, the Department of Homeland Security (DHS.) Currently, the Eastern Interconnection Phasor Project (EIPP) is deploying PMUs on the eastern U.S. grid [1].

A collaborative environment like the above gives rise to new challenges involving the data quality of the aggregated state estimation that is disseminated to network participants. Suppose that the aggregated function $f(d_1, d_2, \cdots, d_M)$ takes as inputs PMU data d_i from utilities $Utility_i$ and outputs state estimations. Assume that a faulty sensor at $Utility_2$ produces out-of-range PMU measurements. Including these d_2 readings in f gives an inaccurate view of the current state of the power grid. Users $U_i, \cdots, U_N$ are not able to detect the source of the inaccurate readings since they only receive the aggregated result.

Trust management can help in providing information about uncertainty in situations such as this. In order to manage the risk of using incorrect data, users U_3, U_4 join a *network of trust* that allows them to exchange evidence with other trustworthy entities regarding the behavioral trends

of utilities. Fault detectors could be used as evidence acquisition mechanisms that detect faulty operation of sensors. Once the faulty sensor at $Utility_2$ is detected by a member of the trust network, users such as U_3, U_4 are informed about it and take the appropriate course of action depending on their policies, including ignoring the state estimations or informing the PMU aggregation entity about the faulty sensor, which in turn will exclude the particular PMU readings from any calculations until the sensor is replaced or reconfigured.

2.2 Power Grid Alert Aggregation and Early-Warning System

The second application family involves better sharing of current information beyond the scope of a single electric utility in order to help increase robustness of the power grid. Today there are two fundamental challenges in sharing more operational information between electric utilities. Better communication for the power grid, such as provided by GridStat, can provide mechanisms to help alleviate these two problems that are explained shortly. However, these mechanisms must be controllable by trust management systems or the greater communications and sharing flexibility can make the problems worse, not better.

The first challenge is that when problems start happening in a part of the power grid, many alerts can fire in a short period of time, and utilities can get buried in the (largely redundant) barrage of such alerts. Alert aggregation allows for transformations of a series of lower-level alerts into higher-level alerts which have a much lower false alarm rate and much richer semantics, and are thus a much more useful indicator of trouble. Such transformations should be based on taxonomies of power grid devices, and be policy-programmable, since different configurations of a given device will have different thresholds for operational reasons. However, alert aggregation must be complemented by similar data quality assessment techniques as the one described in the earlier example. There are cases where incorrect data may suggest a catastrophic situation that does not occur. For example, $Utility_2$'s faulty sensor incorrectly triggers alerts about device failures. In this case, user U_N will be flooded with faulty alarms whereas user U_4 will be more cautious

because it has already been made aware about $Utility_2$ problems with faulty sensors.

The second challenge is that some data are market sensitive, meaning if a competitor has the reading of some key data (for example, the output of a utility's generators) it can, over time, deduce the company's production and pricing strategies. As an example of this problem, instead of sharing market sensitive data directly, derived values such as the instantaneous rate of change (or moving averages thereof) can be shared. Thresholds for particular kinds of devices can be monitored, and alerts generated if they exceed a certain threshold. Since there are currently no means to quantify the risk of sharing sensitive data or derived indicators, the next best alternative is to restrict their access to non-competitors. More generalized trust management allows utilities to reason about the behavior of their peers for different situations. Based on observed behavioral trends, a utility can decide whether or not access to sensitive data should be granted or denied.

2.3 Trust in Next Internet Technology

The National Science Foundation has launched the Future Internet Design (FIND) initiative that focuses on how to best equip the internet infrastructure to meet the needs of the future. One of the proposals that have emerged is *trust-modulated transparency* [5, 16]. The idea behind this, is to perform packet-forwarding decisions based on the trustworthiness of the packets, which in turn is determined by the level of trust between the sender and the receiver.

Trust-modulated transparency, or packet trustworthiness, complements trust assessments for data flows. Starting with an initial trust level between a sender and a receiver, the quality of the packet information could affect the trust relationship between the two entities. In addition, on the absence of any trust between two collaborators, monitoring the quality of the packet information could establish an initial trust level. Therefore, the packet itself could serve as evidence to trust management system that operates on the receiver. Trust management systems could also be operational at critical internet points, such as ISP and routers, to do filtering of data based on the end user's preferences.

CHAPTER 3

TRUST MANAGEMENT SYSTEM (TMS) OBJECTIVES AND REQUIREMENTS

In order to address trust in indirect interactions, such as power grid interactions described earlier, a trust management system (TMS) framework must be designed to meet two objectives that are related to the characteristics of these interactions: dynamic trust that deals with the evolving relationships between participants and composable trust that handles the association of multiple participants for joint operations. In order to further enhance trust assessment, a TMS should also strive for two additional objectives: broad trust scope and support of collaborative environment.

This chapter presents the trust approach that a TMS is built on, followed by a detailed discussion of the four TMS objectives. The requirements that the TMS must meet in order to achieve these objectives are presented next. Finally, a number of paradigms new to trust management are described.

3.1 New Trust Approach

Trust has been traditionally assumed to be inherent in security services. Consider the confidentiality service, for instance, that uses encryption algorithm E. A correct implementation of the E encryption algorithm will always conform to its design principles and thus the algorithm will always perform as expected. Consequently, trust in the E implementation is absolute, and in turn it is inferred that trust in the service is absolute. However, this inference is invalid because it neglects the digital keys that are used as inputs to the encryption algorithm. Digital keys are exchanged directly between two interacting entities as part of the communication protocol or third parties (out-of-band channels, certificate authorities) provide certified key–owner bindings to the interested entities. Regardless of the delivery method, the validity of that binding has to be verified as well. Thus, the confidentiality service requires not only a trusted implementation of an encryption

algorithm but a trusted user-key binding too.

Trust encompasses even more than message confidentiality and source authentication, which have been the traditional trust scopes as shown in the example above. Trust's broader scope covers not only security issues but behavioral and QoS issues as well. Consider a data dissemination system, based on the publish-subscribe paradigm, that operates on the following policy: valid and non-malicious information (behavioral requirement) is publicly available but must not be tampered with (security requirement) and must be received in a timely manner (QoS requirement). In order to enforce this policy the appropriate security, behavioral, and QoS mechanisms must be in place to implement the policy, as shown in Figure 3.1. Digital signing algorithms can guarantee message integrity but they offer no assurance about the quality of the message contents; this is the task of behavioral mechanisms that deduce behavioral patterns and trends for the information producer. Finally, QoS mechanisms are needed to provide guarantees that the information producer and the network will meet the QoS properties as contracted. We call behavior, security and QoS the three *general trust facets*, which are further refined into more specific facets called *properties*. Properties include authentication, competence, and delivery rate. Any trust requirement for a distributed application can be categorized as security, behavioral, or QoS requirement.

An *assessment service* quantifies and evaluates each of these requirements. Subsequently, there are three services: behavioral, security and QoS assessment services. Without loss of generality, we consider all three services as integrated parts of an entity's functionality. The entity supports mechanisms to implement the services. However, it is reasonable to distribute the services to other entities as well.

A behavioral assessment service provides two evaluations: a competence evaluation and a motivation evaluation. The criteria for each evaluation are used to quantify the evaluator's expectations regarding the behavior of a trustee. A trustee's competence is its ability to carry out an action as expected. A trustee may be competent with respect to a set of expectations, and incompetent with

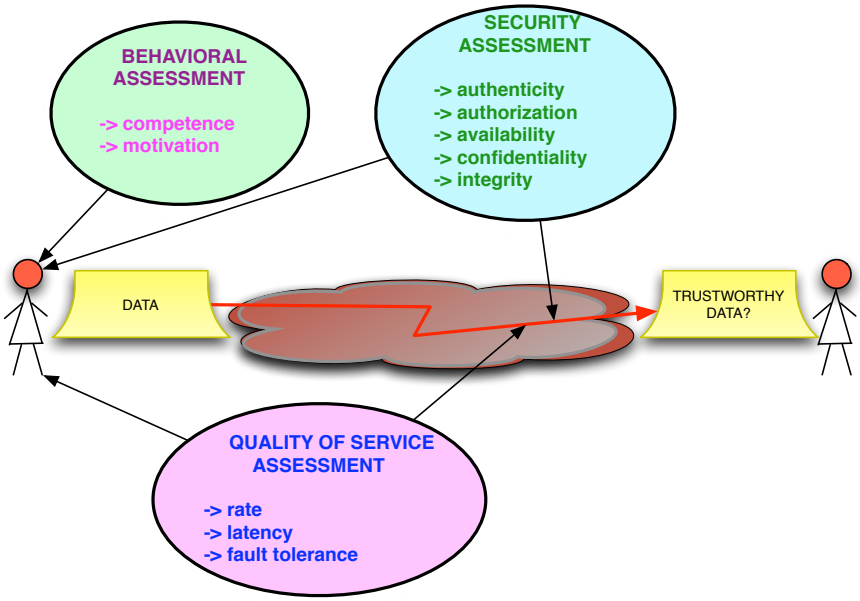

Figure 3.1: Data Trust: Security, Behavioral and QoS Assessments

respect to another set of expectations, depending on who the trustor is. As a complement to competency, motivation assessment accounts for the trustee's intentions. Cooperation and competition bonds are crucial factors in deducing the common interests between a provider and a consumer. Behavioral assessment is not limited to end entities but it extends to intermediate entities as well.

The second service is the security assessment service. Security services such as authenticity, confidentiality, integrity, availability and authorization enhance the security of the transmitted information by safeguarding it against attacks [41]. A number of different security mechanisms exist that provide security services such as encryption, digital signatures and authentication protocols. Entities need the necessary tools to assess a channel's resistance to security attacks, such as interruption, interception, modification and fabrication of an information flow.

Finally, quality of service (QoS) plays an important role in trust evaluation. The QoS requirements of an information flow are satisfied when the information producer and the network resources fulfill their assigned parts of the QoS contract. A QoS violation is a deviation from the expected behavior by the resources disseminating the information. When latency or rate are

unexpectedly poor the information can be out-dated and thus less trustworthy.

While trust is an integral part of decision making in collaborative models, there is no unique way to determine the right level of trust, or which facets to use. Researchers have defined trust concepts for many perspectives, with the result that trust definitions overlap or even contradict each other [3]. The reason is that decisions about how to evaluate each facet lie with the evaluator and can differ substantially from situation to situation. As of today, there is no comprehensive definition that covers the semantics of end-to-end trust for interactions in which information is delivered by intermediaries. End-to-end trust is essential for topologies such as publish-subscribe where interactions are dynamic and they always involve the collaboration of multiple entities to disseminate data from its source to its destinations. We propose using a concept we call *information lifecycle* to analyze the more complex situation of indirect interactions. We define information lifecycle as the interval in which information is created and consumed. In the realm of data trust, the information lifecycle consists of three stages: generation, dissemination, and consumption. At each stage, the entity responsible for the information is the information producer, information dissemination medium, and information consumer, respectively. This decomposition into lifecycle stages allows trust to be examined and evaluated at this finer granularity, namely, each stage in the information lifecyle. Appendix B investigates in detail the applicability of information lifecycle trust on peer-to-peer and publich-subscribe systems.

Building a trust management service for indirect interactions requires defining those issues that are necessary for extending a general trust management system in such a way that dynamic trust assessments of all entities that handle the information, not just the creator or consumer of the data, are supported. Therefore, a trust management system for indirect interactions should address the following objectives:

- **TMS Objective 1: Dynamic Trust** TMS must allow changes to trust relationships during the operational lifetime of the system in such a way that they reflect the actual interactions

between participants. The interactions within large-scale dissemination systems are dynamic for a number of reasons. First, new alliances between participants are formed or existing ones are dissolved, and second, the dynamics of existing coalitions change due to diverse policies, change of leadership, and experience. The underlying trust relationships among the participants should reflect the dynamic nature of the interactions that span these different administration domains. In order to support dynamic trust, current trust for a given entity must be revised based on experience and other information (from both itself and others) that becomes available only after the initial relationship is specified.

- **TMS Objective 2: Composable Trust** Indirect interactions are, by definition, based on the collaboration between multiple entities that cooperate to carry out the task of disseminating information from its source to its intended destination. Obtaining a comprehensive assessment of trust for such interactions requires a trustor to reason about trust not only for the information destination and source, but also for the intermediary entities that act as forwarding servers. In order to support composable trust, a number of pairwise relationships must be identified and synthesized in a way that allows an entity to assess trust for end-to-end indirect interactions. Additionally, TMS must consider that the overall system, and thus the pairwise relationships, could span a number of different administrative domains.

- **TMS Objective 3: Broad Trust Scope** TMS must cover a broad trust scope including the traditional trust usages such as access trust and identity trust as well as non-traditional ones such as fault detection and establishment of data quality.

- **TMS Objective 4: Collaborative Environment** TMS must operate in a collaborative environment. An entity's ability to monitor and manage all interactions in a large-scale distributed system is limited and therefore it needs to rely on other entities' opinions and experience. Users must be able to choose their collaborators and update their collaboration list according to their needs and the specific situation.

3.2 TMS Requirement List

The four activities of a generalized trust management system [26] must be supported for any trust management framework that is designed to meet the trust requirements of indirect interactions. The first activity is the *Trust Evidence Collection*, which is the process of collecting evidence required to make a trust decision. Evidence refers to information that stands as proof of one's behavior, attitude, or external attributes. Evidence must not be confused with forensic evidence, which must adhere to the standards of evidence that is admissible in a court of law. The second activity, *Trust Analysis*, is the process of examining trust relationships to identify implicit relationships. *Trust Evaluation* is the third activity that evaluates evidence in the context of trust relationships. Finally, *Trust Monitoring* is the activity that is responsible for updating trust relationship based on evidence.

Given the four TMS activities and the four TMS objectives, we categorize the requirements for a trust management system that handles indirect interactions in five groups: each of the first four categories represents requirements associated with a TMS activity, and the fifth one discusses requirements on external entities.

R1 Evidence Collection and Distribution

R1.1 Heterogeneous Forms of Evidence

TMS must support multiple types of evidence, including recommendations. Diversity in evidence types allows for a broader assessment of trust because knowledge is obtained from multiple sources.

For example, role-based access control decisions rely on external recommendations and profile information for the intended trustee. Not only the trustee's role must be verified prior to access, but also reputation information is needed to assure that the trustee will not abuse the privileges of the role.

R1.2 Selective Collection and Distribution of Evidence

The collection and dissemination of evidence is not mandatory. A TMS user must be

able to specify the source and frequency of received evidence. Similarly, a TMS user must also be able to restrict dissemination of its own evidence to selected users.

For example, a user should be able to prevent disclosure of sensitive information to business partners that are considered to be competitors.

R1.3 Dynamic Management of Evidence Streams

The evidence streams (both incoming and outgoing) should not be assumed to be static, but they are changing according to the user's policies.

For example, a recommender that acquires a reputation for favoritism or a new business partner may not be a reliable source of recommendations and thus the use of the evidence stream originated at those recommenders should be used cautiously.

R2 Trust Analysis

R2.1 Time-Aware Trust Relationships

TMS must model time as a fundamental quantity that allows reasoning about it during the specification and analysis of trust relationships.

For example, the trust assessment of a path that is composed of three servers can be derived if the intersection of the valid intervals of the three trust relationships is not empty.

R2.2 Composable Trust Constructs

TMS must provide the necessary constructs that will allow composable trust relationships.

Systems based on the publish-subscribe paradigm rely on intermediary entities to forward information from its source to the intended target. A trustor must derive trust assessments for all entities that actively participate in a particular data stream. For example, a secure communication link does not provide guarantees about the quality of the data from an unreliable publisher.

R3 Trust Evaluation

R3.1 Evidence Aggregation

TMS must provide a wide range of mechanisms to aggregate evidence of the same or different type. TMS must support a range of typical voting algorithms but also deal with incorrect and missing inputs to the aggregation.

For example, the feedback from recommenders is weighted by the level of confidence attached to it. All recommendations are combined by an aggregation scheme into a single output. Trustors should be allowed to choose the aggregation scheme that conforms to their policies. A weighted average aggregation scheme may be suitable for one trustor whereas a majority scheme is more appropriate for another.

R3.2 Evidence-to-Expectation Mapping Functions

Expectation is defined as a requirement and its allowed values that a trustor has for a particular interaction with a trustee; these values are constrained by equality and inequality operators. The observed value for an expectation is not necessarily derived directly from a single type of evidence. The TMS must allow a user to express functions that map evidence to expectations. These functions specify the inputs to aggregation methods and may vary depending on the *imputed trust mode* (see R4.2 below).

For example, a trustor may choose to specify the behavior expectation as a function of reputation and reliability. Behavior is not collected directly from evidence streams, but on the contrary, recommendations and reliability measurements are manipulated to derive behavioral values. Aggregated results may be used to evaluate more than one expectations; for example, recommendations affect a trustee's reputation and at the same time they are also used to extract behavioral trends for the particular trustee.

R3.3 Expectation Satisfaction

Expectation satisfaction occurs when the trustee's observed value for a requirement

falls into the range of allowed values for that particular requirement. TMS must provide a wide range of techniques that target expectation satisfaction.

The reasoning behind this requirement is similar to the one explained for the R3.1 Evidence Aggregation requirement.

R4 Trust Monitoring

R4.1 Trust Re-evaluation

TMS must take into consideration the dynamic nature of the network and the dynamic behavior of the network participants whenever a trust assessment is made.

There are a number of situations where the dynamic nature of the network and its participants is observed. For example, new participants join the network and existing ones depart from it. Participant's performance and properties change. Legal contracts may force collaborators to become competitors. Trust relationships should reflect these social, organizational and network changes.

R4.2 Imputed Trust Mode Support

TMS must provide for different operational modes. Policies will have different inputs and rules depending on which mode they are in. These modes are similar to the homeland security alert codes or intrusion detection modes.

For example, during the red alert mode, only predefined trust relationships are active depending on the severity of the situation.

R5 Requirements on Components External to Model

R5.1 Security Services and Certificate Management Tasks

TMS assumes that there are underlying mechanisms that provide basic security services including encryption, digital signing, and certificate management.

As an example, consider certificates as one type of evidence. TMS must have access

to certificate information, either directly from the certificate authority or from another service such as an enhanced directory service that stores certificates as one of the supported attributes. TMS should not support any certificate management activities such as managing certificate revocation lists. These tasks are assumed to be provided by external entities.

3.3 Supporting New Paradigms

In order to meet the TMS requirements, new paradigms are needed. The functionality of the TMS relies on several concepts such as time, trust re-evaluation triggers, expectation satisfaction, trust composition techniques and predefined compound trust relationships. These paradigms are not novel but are new to trust management.

3.3.1 Temporal and Knowledge Awareness

There are two main concepts that are related to dynamic trust: time and knowledge. As time progresses and new evidence is processed into higher-level knowledge, trust relationships must evolve. As a result, the TMS must accommodate some notion of time as well as facilitate the utilization of different kinds of evidence.

Time is paramount for dynamic trust re-evaluation, especially for reasoning about overlapping intervals where multiple trust relationships are considered to be valid. In addition, time must be modeled in a way that time awareness can be embedded into the trust relationships specification, analysis and monitoring mechanisms. Temporal database concepts regarding time are adapted in order to reason about time-aware trust relationships. To be more specific, Allen's algebra [8] is used to reason about relationships between time intervals.

Knowledge awareness is accomplished by identifying evidence types that must be incorporated into the TMS.

3.3.2 Trust Re-evaluation Triggers

Time and knowledge awareness allows multiple states of trust to evolve from the initial state. While time and knowledge enable a change of trust, specific events and conditions are the ones that actually cause re-evaluation of relationships. We refer to these as trust re-evaluation triggers.

The first trust re-evaluation trigger is the *insertion, deletion or modification* of trust relationships. New pairwise trust relationships may lead to the formation of new end-to-end trust relationships. Similarly, removal or modification of trust relationships may lead to the invalidation of existing end-to-end trust relationships. The second trust re-evaluation trigger is the *expiration of the trust relationship's valid time period*, where upon expiration of the specified valid period a trust relationship is re-evaluated. Another trust re-evaluation trigger is the *violations of the expectations for trust relationship(s) that pertain to a specific trustee.* Expectation violation occurs when the trustee does not satisfy its expectations. Violations are detected with the arrival and processing of local and global evidence. Consequences of violations vary in scope including, but not limited to, individual scope and group scope. For the former category, only the trust relationships for the trustee in question are revised whereas the latter category re-evaluates trust relationships for an extended group of trustees. A last trigger is the *upgrading or downgrading of imputed trust mode.* A change in the imputed trust mode may result in using different inputs to the configurable trust policies. That, in turn, could result in relaxing or restricting requirements, and adjusting the permissible deviation between observed and expected values.

3.3.3 Expectation Satisfaction

Expectation satisfaction is the process that determines whether or not the trustee observed value satisfies the allowed values of a given requirement. *Covering techniques* are methods that provide expectation satisfaction. Expectation satisfaction concerns itself with two situations. Given a trustee's observed values and a covering technique, reach a binary decision on whether or not these values satisfy the expectations. And, in the case of a negative outcome, examine whether or not

deviation from the expected values could be tolerated.

Covering techniques are grouped into three categories: strict satisfaction, relaxed satisfaction, and user-defined satisfaction. The strict satisfaction category includes techniques that operate on the premise that the actual values *cover* the range of allowed values. For instance, consider expectation e to be $(cooperation, >, 5, 4, ev)$, with an allowed value 4 and an observed value of cooperation to be 5. The value of 5 satisfies the binary relationship $>$. On the contrary, relaxed satisfaction methods allow actual values to deviate from the allowed ones. An actual value of 3, for example, does not satisfy the binary relationship, however it might be the case that the deviation of 3 from the desired value is permitted. Finally, user-specified techniques are user-defined functions that test expectation satisfaction according to the user's criteria. The various covering techniques are given in more detail below:

- Strict satisfaction
  - Covering relation: as defined in SIENNA [14] but customized to expectation semantics. (note that the covering relation is used by SIENNA to propagate subscriptions and advertisements as well as to perform notification-subscription matching.)
  - Strict covering relation: it is a covering relation with the additional constraint that all trustee actual values must cover at least one expected value.
  - Selective covering: it is possible that not all expectations need to be met by the specific trustee, but instead only a subset of them is applicable.
  - Preferred covering: it is possible that not all expectations carry the same "importance" weight; some expectations are more important than others. In this case, the expectations are evaluated based on their importance factor.
- Relaxed satisfaction
  - Semantic distance: An expectation set with n requirements (variables) is mapped into

an n-dimensional Cartesian system as a single point. In this space, it is possible to evaluate expectation value deviation using a modified k-nearest algorithm [40] (with or without weights) that exploits semantic distance as the indicator of the deviation between any two points (e.g. Euclidean distance). In this case, the equality operator is the only operator that can be used between the actual and observed values. It is possible to have multiple points for expectation sets that use the inequality operator, but the calculation of the points may not always be possible (e.g. $>$ operator for requirements without upper-bound restrictions).

  - Individual deviation: the trustor specifies the range of deviation that can be tolerated for each requirement in the expectation set.
  - Hyperplanes: A hyperplane [31] is a plane that is extended in n dimensions. Similar to the semantic distance technique, an expectation set is modeled as hyperplanes, one for each requirement. The intersection of all the appropriate halfspaces is a polytope (which may be unbounded). Actual values are mapped into the n-dimensional space as a point and the distance between the point and the polytope is calculated.

- User-defined satisfaction
  - Any user-defined function that conforms to the user's criteria could be used as an expectation covering method.

3.3.4 Trust Composition

Trust composition is one of the TMS properties that allows for end-to-end trust management of indirect interactions. Defining and reasoning about trust in such interactions requires the proper management of a number of direct pairwise trust relationships. Trust constructs make explicit the stages in the information lifecycle of indirect data delivery.

Trust composition is also required at the data level. There are cases where data is aggregated

at a particular place and the result is further disseminated to end entities. In this case, the various trust relationships that correspond to each datum must be composed in order to derive the trustworthiness of the aggregation result.

Trust composition is based on the expectation satisfaction covering techniques between an expectation set and a trustee's actual values. These techniques are applied on multiple expectation sets to answer the following questions:

- Given a set of expectation sets $E = \{e_1, e_2, \cdots, e_n\}$, is there an expectation set e that satisfies all members of set E? We call this the *expectation set satisfaction problem*.

  We use the notion of reducibility to show that this problem is NP-complete. A problem Q can be reduced to another problem Q' if any instance of Q can be "easily rephrased" as an instance of Q', the solution to which provides a solution to the instance of Q [17]. This expectation set satisfaction problem is NP-complete, and we can prove that by showing that the problem belongs to the NP class and then show that the well-known problem CIRCUIT-SAT $<=$ SAT. That means, any instance of the CIRCUIT-SAT can be reduced in polynomial time to an instance of the expectation satisfiability problem.

- Given a target expectation set e and a set of expectation sets $E = \{e_1, e_2, \cdots, e_n\}$, does the target expectation set satisfy all members of set E?

  This is a comparison problem between set e and set e_i from E and it is solved in polynomial time, given that a hash table is used to store the requirements and their values. This comparison process is repeated n times to cover all members of E.

3.3.5 Predefined Compound Trust Relationships

Predefined compound trust relationships refer to the grouping of multiple trust relationships that are automatically activated (become valid) when their prespecified mode of operation ("red alert", etc. as evaluated and announced by external mechanisms such as a QuO contract [50]) becomes the operational mode of the system. The imputed trust mode of the trustor could be used to specify

these compound trust relationships. For instance, an operational mode (e.g. red) could be mapped to a specific imputed trust mode (e.g. distrust). In this case, imputed trust modes are differentiated by their inputs to the configurable trust policies that include the specification of expectation sets, the bounds of the acceptable tolerance for value deviations, and the parameters of aggregation algorithms such as weight.

CHAPTER 4

HESTIA TRUST MODEL DESIGN

Chapter 3 identified and documented the TMS requirements that are needed to provide dynamic and composable trust for indirect interactions. This chapter presents the design of a TMS, called Hestia, that meets these requirements. The focal point of the Hestia design is modeling the evidence flow within the system and the effect it has on trust relationships. Section 4.1 describes Hestia at an abstract level, emphasizing the interactions between the system and external entities outside Hestia. The two architectural design choices are presented in Section 4.2. Section 4.3 discusses in more detail the functionality of Hestia's individual components. The operation of Hestia is demonstrated in Section 4.4.

4.1 Hestia Context Diagram

Figure 4.1 illustrates the abstract model of Hestia, which starts with a context diagram showing the system as interacting sets of processes connected to external entities outside of the system boundaries. This process is expanded to more detailed diagrams that divide the system into smaller parts (see section 4.3).

The external entities that interact with Hestia are the following:

- Application Entity, which can be either a trustor or a trustee. Information producer (publisher) and information consumer (subscriber) are such entities.
- Network Entity, which refers to the actual information dissemination medium. This is a network of forwarding servers, usually transparent to the application.
- Network of data flows, which includes application data streams and trust data streams.
- Data Dissemination Management Authority, which controls and manages network resources.

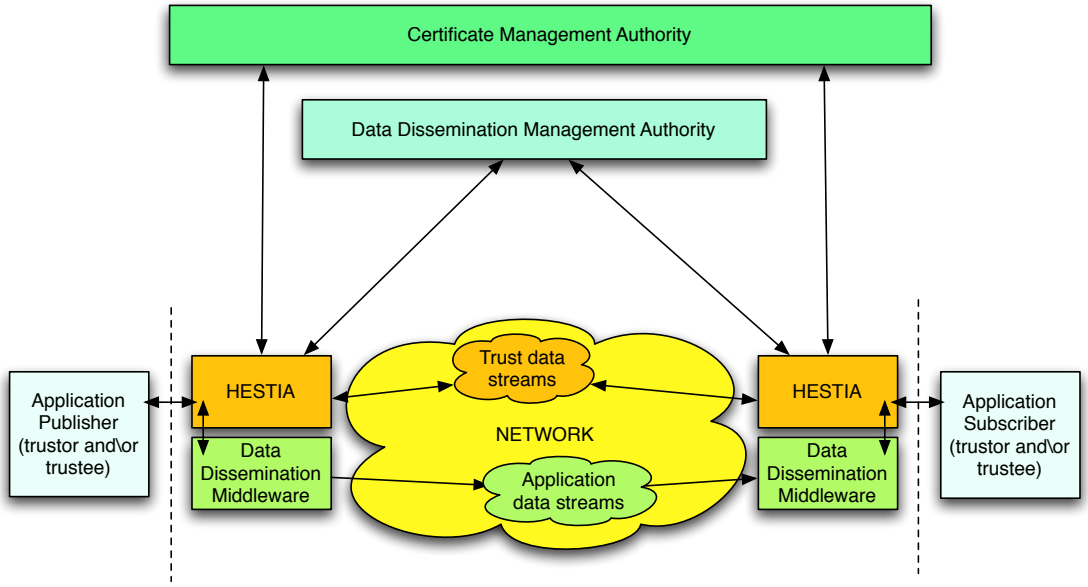

Figure 4.1: Hestia Context Diagram

- Certificate Management Authority, which handles the management of certificate related issues
- Data Dissemination Middleware, which provides an abstraction of a message bus to the application layer.

4.2 Architectural Design Choices

Two architectural design choices were made at the early stages of the design. These choices were made in order to provide a reasonable bound on the scope of the entities which would be incorporated or designed in detail. We believe that these design choices in no way limit the general applicability of the architecture, something we hope to demonstrate in research in the near future.

4.2.1 Architectural Design Choice 1: Trust Service Transparency

First, trust management is provided as a middleware service. However, it could be provided at any level. Hestia can be completely transparent to the application layer, given that the necessary policies are provided as configuration information during installation. On the other hand, with

Hestia, the application can be allowed to customize the policies and be a trust-aware application. There are 16 possible interactions between the application module (AM) and the trust module (TM), shown in Table 4.2, including the following:

- Application is trust-aware and the trust module is not transparent to the application. This is the case for subscribers that are not willing to delegate trust decisions affecting subscription paths to any other module.
- Application is not trust-aware and the trust module is transparent to the application. This is the case for publishers that are devices with limited functionality such as sensor devices.
- Application is limited trust-aware, meaning that it is informed of trust module's decisions and may influence them. This is the case for the subscriber that delegates authority to the trust module, but it is informed (notified) whenever an action is taken by trust module, possibly via callback objects.

4.2.2 Architectural Design Choice 2: Limit the Scope of the Trust Service

The second architectural design choice was to limit the trust service to the data streams as a baseline. The entire problem space motivated and defined by this research work is too broad to cover in a single dissertation. We will not examine the interactions, and thus any related issues, between a trustor and the management authority. For example, a trustor will not make trust assessments for issues like: authentication, authorization, and confidentiality for subscription and publication advertisements, behavior of management authority, fault tolerance of this authority, etc. We must note that Hestia was designed to handle these issues as well.

We will also not going to examine the trustworthiness of the trust service itself. It will be premature to go beyond this baseline. Exploring this is very important future work, with similarities to the well-known question "Who guards the guardians?" In our case, the question becomes "Is the trust management service trustworthy?" The various implementations of the trust model must

	Managing Application Data Streams (no notification)	Managing Application Data Streams (notification)	Managing Trust Data Streams (no notification)	Managing Trust Data Streams (notification)
Fully Aware	AM		AM	
Semi	AM		TM	
	AM			TM
	AM			AM
	TM		AM	
	TM			TM
	TM			AM
		AM	AM	
		AM	TM	
		AM		TM
		AM		AM
		TM	AM	
		TM	TM	
		TM		TM
		TM		AM
Transparent	TM		TM	

Figure 4.2: Interactions between application module and trust module

be checked against the TMS requirements. The reason is that these implementations may collaborate (e.g. exchange recommendations) and may affect the functionality of each other. In order to address the original question of whether or not the trust service is trustworthy, a number of issues must be considered:

- Is there any sort of guarantee that different implementations of the model will conform to the design principles?
- Consider the possibility of having implementations tested against the design principles yielding in certified implementations. How do implementations check the certifications of other implementations?

A technique called “Proof-Carrying Code (PCC)” could be used in some context for safe execution of untrusted code. In a typical instance of PCC, a code receiver establishes a set of safety rules that guarantee safe behavior of programs, and the code producer creates a formal safety proof

that proves, for the untrusted code, compliance to the safety rules. Then the receiver is able to use a simple and fast proof validator to check that the proof is valid and hence the untrusted code is safe to execute. This technique is not applicable in our case because the burden of checking the code of all collaborators is simply not practical (or feasible). However, the management authority may take on this responsibility and provide ways for collaborators to check with the management whether or not implementations are certified.

4.3 Hestia Components and Databases

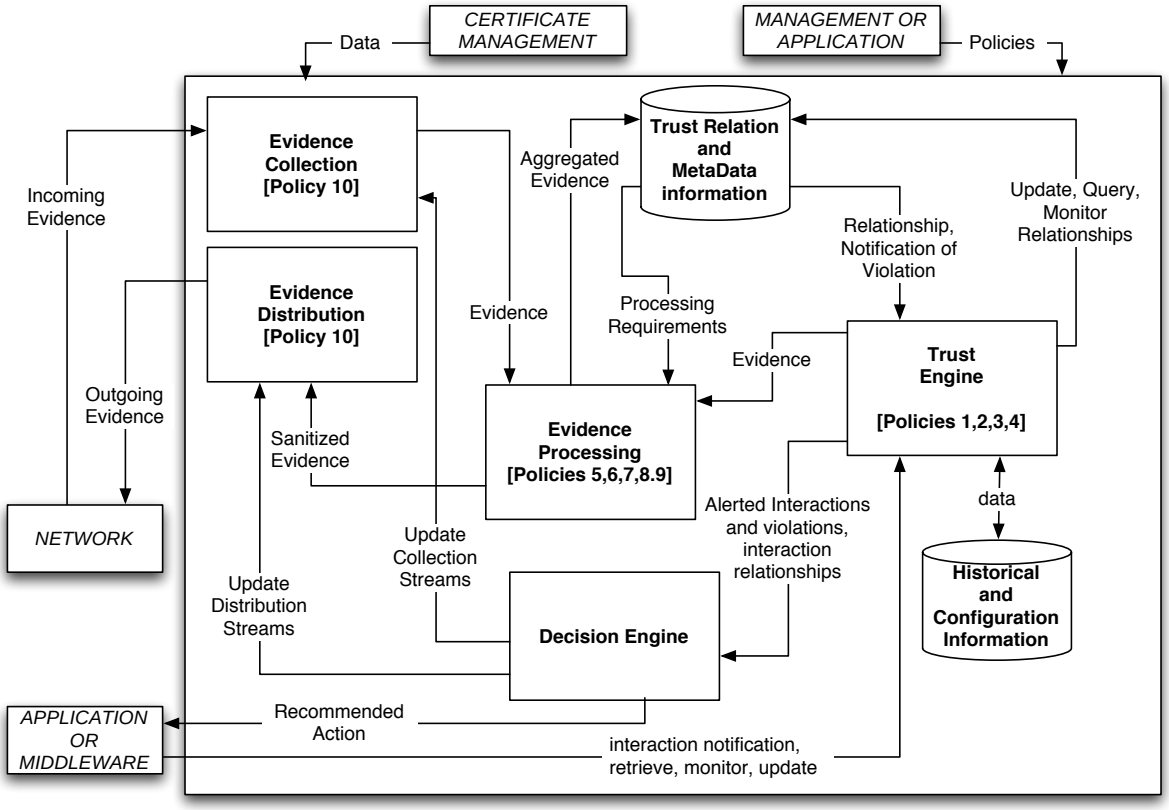

Figure 4.3: Hestia Components and Databases

Hestia is composed of five components: Evidence Collection, Evidence Distribution, Evidence Processing, Trust Engine, and Decision Engine. There are also two databases: the Trust Relation and MetaData Information (TRMI) and Historical and Configuration Information (HCI). Figure

4.3 illustrates the interactions among the components as well as the interactions between the components and the databases. Each component and database is described in more detail in the subsequent subsections. Note that the policies that dictate the functionality and operation of the various components in Figure 4.3 can be found in Section 5.1.

4.3.1 Evidence Collection Component

The Evidence Collection component, illustrated in Figure 4.4, is responsible for collecting evidence such recommendations, QoS values, and complaints from external sources. It also collects evidence locally, such as locally detected faults.

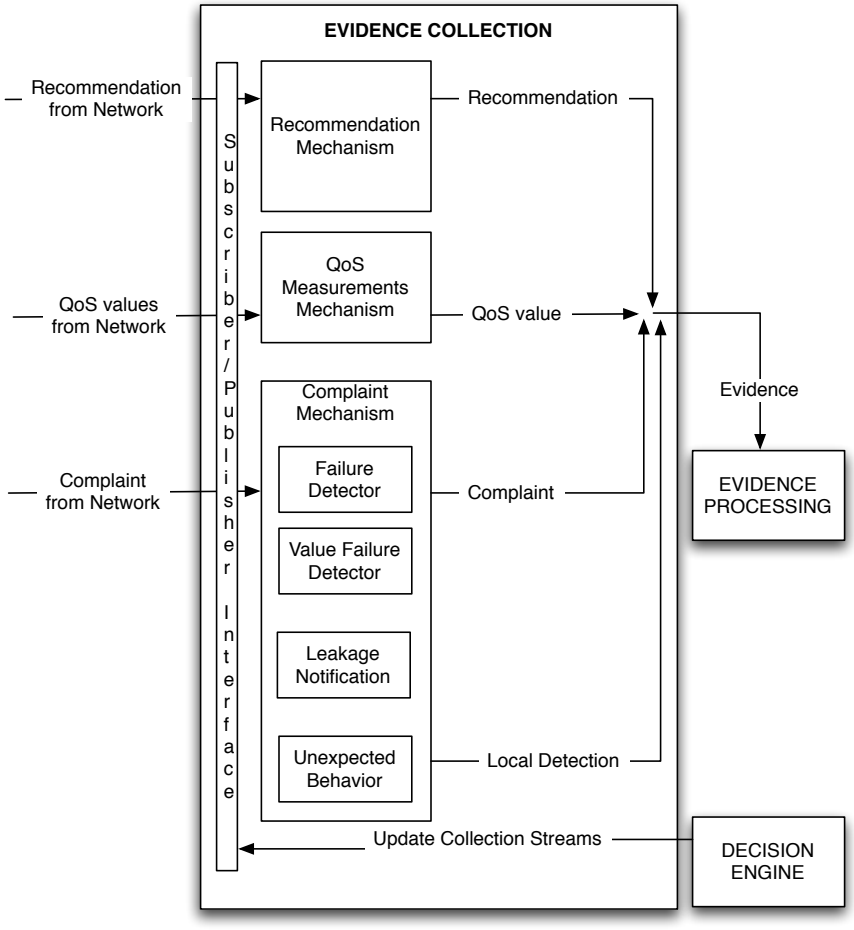

Figure 4.4: Hestia Evidence Collection Component

Evidence is closely related to the three trust facets of behavior, security and QoS. Each facet is refined into more specific ones, called properties, as shown in Figure 4.5 (Note that this is not the entire space of the facets-properties-raw evidence). Behavior is determined by the properties of competence and motivation, where competence is further expressed in terms of reliability, dependability, accuracy, etc. Security properties include authentication mechanisms, encryption algorithms, and digital signing algorithms. Finally QoS is refined into rate, latency, bandwidth and redundancy.

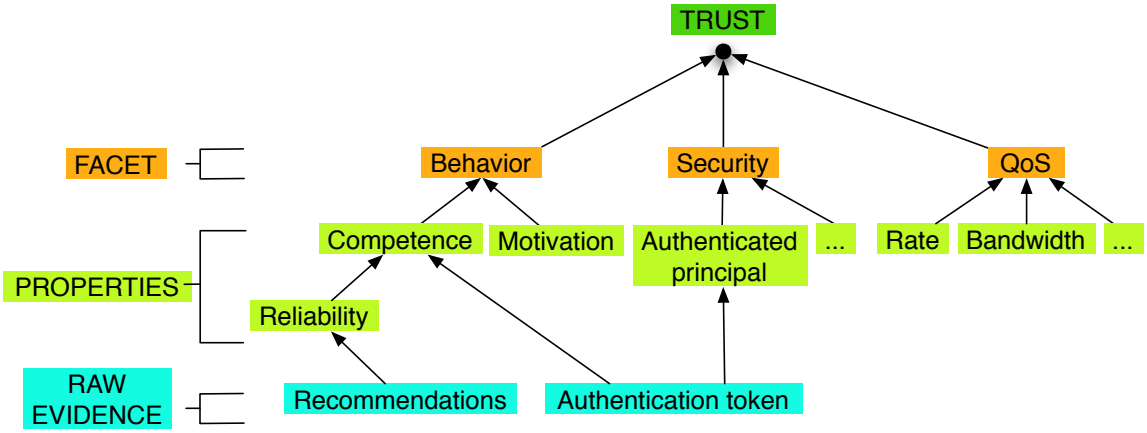

Figure 4.5: Hestia Trust facets, Properties, and Raw Evidence

Each property is evaluated based on the raw evidence that is collected by evidence acquisition mechanisms that are embedded in the Hestia framework. These mechanisms include recommendation, complaint, QoS instrumentation and fault tolerance mechanisms. The kinds of evidence that are currently supported are local evidence and global evidence. Local evidence includes experience, cooperation and competition factors. Global evidence refers to recommendations, complaints (including value omission faults, semantic fault, and leakage notification), legal contracts and agreements, QoS measurements (server and network performance), profile information, authentication tokens, and connection properties.

The various kinds of evidence may be applicable to more than one property as shown in Figure

4.5. An authentication token could be used to prove identity, but it could also be used to predict the expected level of competence based on group memberships.

4.3.2 Evidence Distribution Component

The Evidence Distribution Component, shown in Figure 4.6, is responsible for distributing evidence (local recommendations, complaints and locally detected faults) to other application entities. All evidence is distributed via the same evidence acquisition mechanisms that are utilized by the Evidence Collection component.

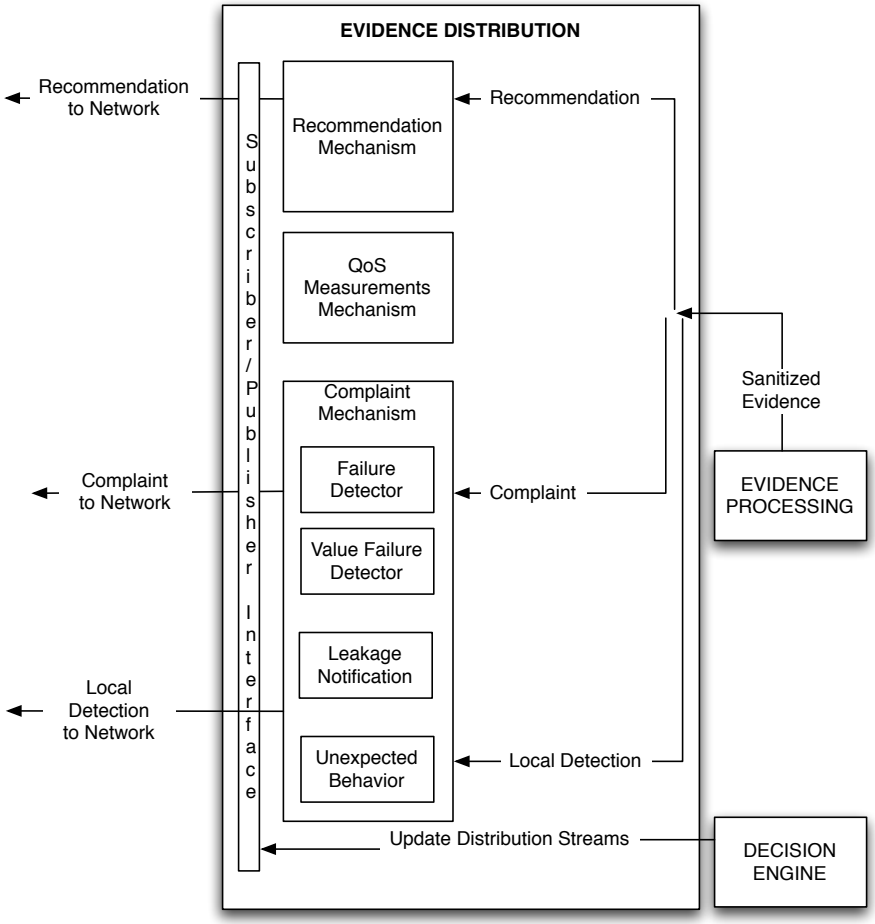

Figure 4.6: Hestia Evidence Distribution Component

4.3.3 Evidence Processing Component

The Evidence Processing component, shown in Figure 4.7, performs two tasks. The first task is accessing the TRMI database to check where and how the received evidence is to be applied. The second task is to process evidence that is to be made available to external entities. To be more specific, the Evidence Processing component is responsible for processing incoming and outgoing messages. The authenticity of the incoming evidence message is verified before sanitization, a form of filtering, and the outgoing evidence message is digitally signed prior to its dispatching into the network. Sanitization occurs for both incoming and outgoing evidence. An example of sanitizing incoming evidence occurs when at certain operating modes, evidence from particular recommenders is discarded, which otherwise would be processed. On the other hand, sanitization of outgoing evidence may occur when the entity has to decide whether or not to share evidence. For example, an entity might wait before disseminating complaints that involve one of its collaborators. An entity is under no obligation to share its own trust assessments. On the contrary, it provides them according to its policies.

Incoming evidence must be evaluated in the context of existing trust relationships. As a result, this component accesses the processing requirements for trust relationships (triggering rules, aggregation methods) with the intention of assessing the new evidence with respect to the particulars of the trust relationships. When triggering conditions are activated, the evidence is aggregated. Aggregating evidence can be seen as a voting mechanism where instances of evidence types are combined by a voting scheme into a single output. Aggregation algorithms can be triggered either at predefined intervals or when a number of instances arrive at the evaluator. The aggregated result is an observed value for an expectation. It is important to note that an observed value is not necessarily related directly to a single evidence type. A user could specify functions that map evidence types to expectation values.

Hestia supports a spectrum of aggregation algorithms targeting the aggregation of evidence

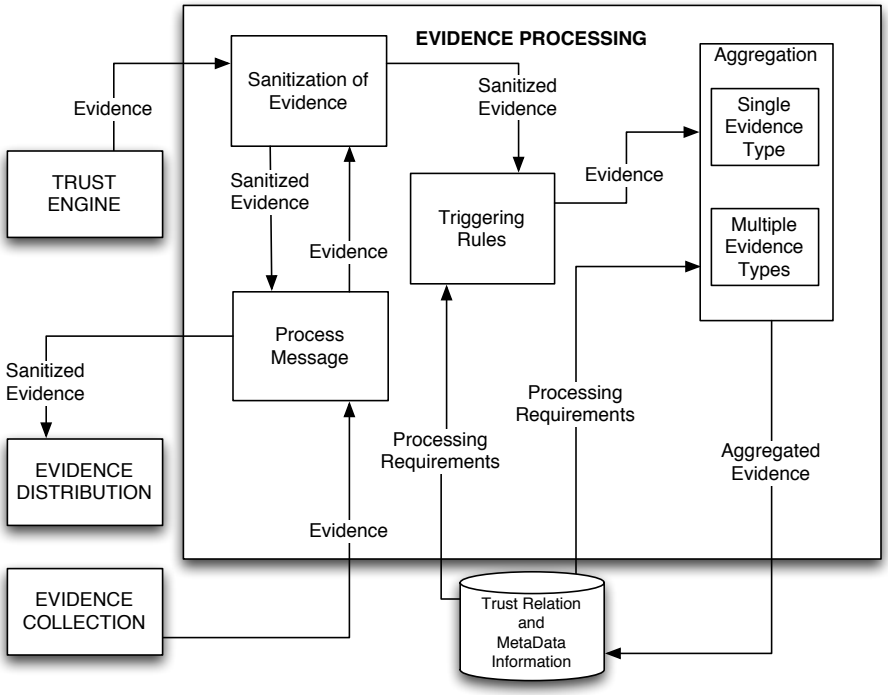

Figure 4.7: Hestia Evidence Processing Component

from multiple instances of a single evidence type and evidence from different evidence types. The aggregation algorithms that are supported by the model include *average*, *weighted average*, *majority*, *m-out-of-n*, *plurality*, and any user-defined aggregation. These user-defined mechanisms can be provided by a virtual machine with a programmable language such as [24, 9]. Consider v_{all}^{j} to be the aggregated evidence value for entity j, v_i^j to be evidence value by external source i for entity j, and τ_i to be the trust relationship between the evaluating trustor k and external entity i. The aggregation algorithms that trustor k can use are shown in Table 4.1.

Table 4.1: Aggregation Algorithms for Single Evidence Type

Average	$v_{all}^{j} = \frac{1}{n}\sum_{i=1}^{n}(v_i^j)$, where n is the number of external sources
Weighted Average	$v_{all}^{j} = \frac{1}{n}\sum_{i=1}^{n} f(\tau_i, v_i^j)$, where f is a function
Weighted Average Combination	$v_{all}^{j} = \frac{u}{x}\sum_{i=1}^{x} f(\tau_i, v_i^j) + \frac{v}{z}\sum_{i=1}^{z} f(\tau_i, v_i^j)$, where $x + z = n$, $u + v = 1$
Majority	v_{all}^{j} is the value that more than $\frac{n}{2}$ external sources agree on (allow for a δ deviation)
m-out-of-n	v_{all}^{j} is the value that at least m out of n external sources agree on
Plurality	v_{all}^{j} is the value that most external sources agree on
User-defined	Any customized aggregation method such as median value, etc

4.3.4 Trust Engine Component

The Trust Engine component is shown in Figure 4.8 and its functionality includes a number of tasks. It monitors existing trust relationships and handles trust relationship violations, which involves updating the relationship's status (see Section 4.3.6, Trust Relation and Metadata Information Database) and checking for trust relationship interdependencies in order to update all relationships affected by the violations. Furthermore, it notifies the decision engine component about interactions that violations are reported for (and the type of the violation). This component is also responsible for handling queries for adding, removing, and updating trust relationships as well as providing trust information about relationships. In addition, it processes interaction notifications, including identifying the pairwise relationships that correspond to the new interaction. Whenever new evidence or queries change the state of the trust relationships, the Trust Engine is performing trust analysis by applying operations such as composition to deduce new relationships. Finally, the Trust Engine monitors the behavior of entities prior to any interactions; this is something that we label as *passive monitoring*.

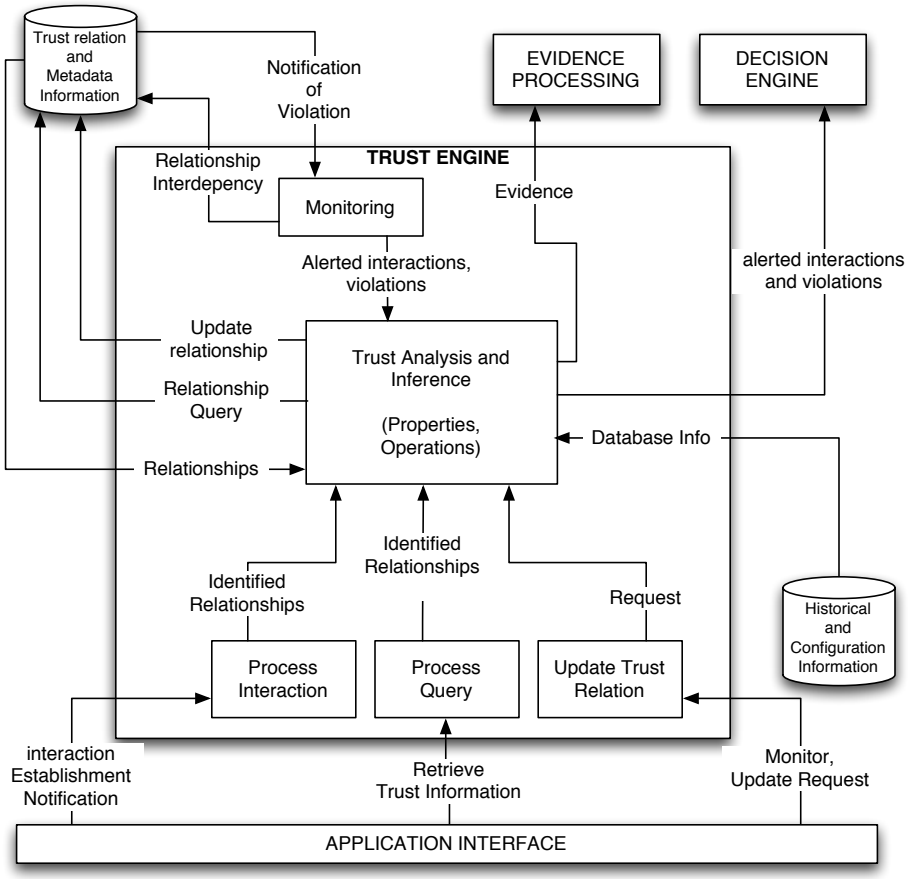

Figure 4.8: Hestia Trust Engine Component

4.3.5 Decision Engine Component

Currently, the Decision Engine component is responsible for recommending actions including adding, removing, or updating subscription paths and establishing the recommender network. For example, in order to limit the risk of information leakage to unauthorized parties, dissemination of sensitive information is suspended if the authenticity of the receiving entity cannot be verified (e.g. its certificate is invalid). However, we envision that the decision engine should make decisions that yield the maximum utility for the trustor.

Consider the case where Hestia is used for authorizing an action. In the case of an information dissemination system, the action set is restricted to four actions: share information, not share information, use information and not use information. Regardless of the action, the result of the authorization must ideally contribute positively in achieving higher-level goals. Thus, trust is a prerequisite of authorization because even though complete (highest possible) trust exists between entities for a specific interaction, the action's authorization still needs to be approved upon a risk-and-benefit evaluation. This evaluation varies with the current state of the network or system as well as with other factors. For example, whats the cost of sharing versus the cost of not sharing?

In general, utility theory represents and reasons with preferences. No trust means that new alternatives must be found. Utility of trust concerns itself with representing and reasoning with preferences related to trust and an overall goal, and it may be outlined as the series of steps below:

1. Establish the high-level goal.

2. Identify the possible actions that trustor can execute to reach the goal. In this case the four general actions are extended to cover a finer granularity (ALL, SOME, NONE) of the amount of information that is involved.

   (a) [Share|Use] ALL

   (b) [Share|Use] SOME

(c) [Share|Use] NONE

(d) [Not Share|Not Use] ALL

(e) [Not Share|Not Use] SOME

(f) [Not Share|Not Use] NONE

Note that actions a,f and c,d are essentially the same. The challenging part is to quantify SOME.

3. Each action has a possible outcome state. Identify those states and assess how close the action is to the overall goal.

4. Assign likelihood to each outcome.

5. Estimate the expected utility of action, given the evidence.

6. Choose an action that maximizes the agents expected utility.

4.3.6 Trust Relation and MetaData Information Database

The Trust Relation and Metadata information (TRMI) database is responsible for storing information about the current state of an entity's trust relationships. Information about the current trust relationships is accessible by any Hestia component. Three sets of information are maintained for each trust relationship (more details in later chapter):

- Trust relationship parameters
  - Context (an action that is performed on the data during its lifecycle)
  - Interval (the time duration that the trust relationship is predicted to be valid)
  - Expectation values (allowed and observed)
  - Trustee trustworthiness

  - Imputed Trust Mode

- Trust relationship metadata
  - Processing requirements for the relationship (aggregation method and triggering rules)
  - Monitoring requirements (covering method)
  - Temporary storage for unprocessed evidence
  - Interaction identifier (unique identifier that is associated with an interaction)
  - Status of relationship
    * OK - valid relationship
    * ALERT - violation(s) occur(s) in this relationship
    * WARNING - violation(s) have occurred for this interaction

- Other information about the specific relationship
  - Reason for termination

A database, by definition, is a collection of information that is organized in a manner that it can be easily and efficiently accessed and managed. Even though the database of Figure 4.9 resembles a "flat" file of records, relational database concepts could be used to decompose the original database into several *relation schemas*. The definition of a relation does not specify any ordering of tuples, however there is an ordering of values within a tuple [20]. Furthermore, all tuples must be distinct, meaning that no two tuples can have the same values for their attributes. In order to enforce this policy, there is a unique key for each tuple. It's beyond the scope of this research work to identify the relation schemas and their respective keys.

Figure 4.9 represents a snapshot of the state of TRMI at a specific time t. The current trustor has 2 trust relationships with trustee Y. For brevity reasons, only the first relationship is described in detail. This trust relationship is established for context *access data1*. As long as trustee Y's

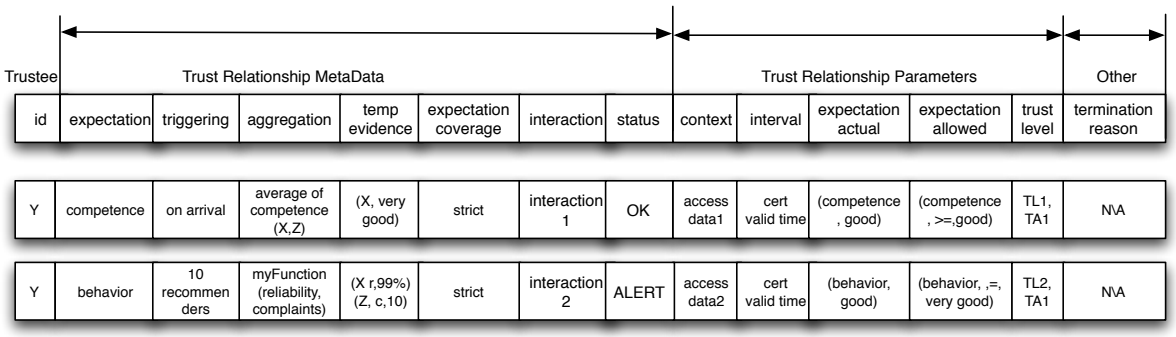

Figure 4.9: Hestia TRMI DataBase

certificate is valid, and the observed value for *competence* expectation is set to *good*, then trustee Y's trustworthiness is *TL1* for imputed trust mode *TA1*.

The competence observed value is the result of aggregating recommendations from recommenders X and Z. Whenever these recommendations arrive, the aggregation method *average* will be applied on the recommender values. At this specific time t, the recommendation from Z hasn't arrived yet, and as a result X's recommendation is temporarily stored until all necessary evidence arrives. The result of the aggregation will replace the value of the *actual* field and this will be compared to the *allowed* field in order to detect violations. At time t, the actual and allowed competence values are the same. A violation depends on the allowable deviation for the current imputed trust mode TA1. If a deviation occurs, the Trust Engine component is notified. For the specific relationship, the expectation coverage is *strict*, meaning that no deviation is allowed. The *status* field is updated by the Trust Engine component (in particular, the Monitoring module) to reflect the current status of the relationship.

4.3.7 Historical and Configuration Information Database

The Historical and Configuration Information (HCI) Database is responsible for maintaining generic information that is used to deduce new trust relationships or update existing ones. More specifically, the HCI Database stores information about the current and past end-to-end interactions that the entity is involved with. An interaction is associated with multiple trust relationships

and thus a trust relationship may be part of multiple interactions. Whenever an interaction is terminated, the trust relationship that is not associated with any other interaction could be considered as a past experience. Storing terminated interactions allows for tracking the history of the interactions and the reason of their termination. The trust relationships that are also associated with these interactions are stored as *experience*.

In addition, HCI stores information regarding legal bindings between actions and expected behavior of trustees. Information about trustees that could be used to deduce behavior trends for trustees, such as competition and cooperation factors, are stored here as profile.

4.4 Hestia Operational Examples

This section demonstrates the operation of Hestia in three situations: arrival of new evidence, self-triggered re-evaluation of trust relationships, and end-to-end trust composition.

4.4.1 Scenario 1: Arrival of Evidence

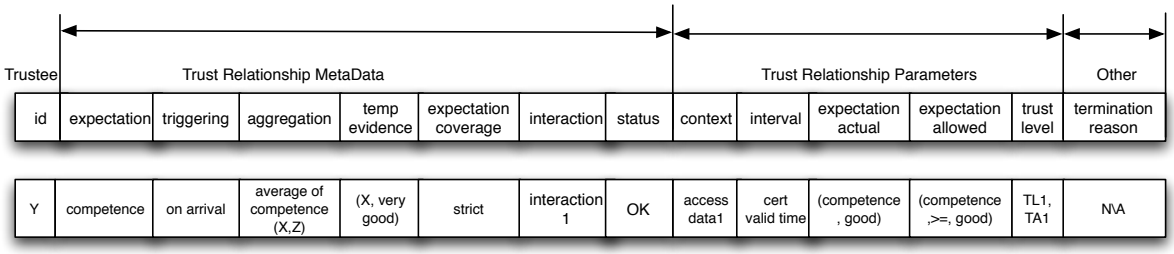

Figure 4.10: Hestia TRMI Database for Scenarios 1 and 2

This scenario describes how trust relationships evolve with the arrival of new evidence (Figures 4.10, 4.11). The description is in the form of steps that are executed sequentially as follows:

1. A recommendation data stream for recommender *Z* has been set up and it is currently active.

2. A recommendation has been received from *Z's* data stream. This is collected at the Evidence Collection component and forwarded to the Evidence Processing component.

3. The recommendation message is processed according to the recommendation format semantics. The recommendation is about the competence level of entity *Y*, which is set to value *very bad*.

4. This recommendation evidence is forwarded to the Triggering Rules module, and the TRMI is searched to locate relationships that this evidence affects. The first relationship in TRMI is affected by this recommendation. The evaluation of the expectation is triggered on the arrival of recommendations from *X* and *Z*.

5. The recommenadation from *X* is already stored and as a result the re-evaluation of the relationship is triggered. The next operation is to lookup how to aggregate the evidence. The method indicated is *average*, and the value of competence becomes *bad*.

6. The monitoring module of the Trust Engine detects that there is a violation of expectations in the updated relationship. The new actual values do not strictly cover the expected ones.

7. The violation is forwarded to the trust analysis module, which simply forwards the violation to the Decision Engine component.

8. The action that is recommended is to prohibit access of *data1* by trustee *Y* and discontinue any recommendation paths associated with trustee *Y*.

9. Trustee *Y's* recommendation path is removed (if any).

10. The application is notified of the removal of trustee *Y* privileges.

4.4.2 Scenario 2: Self-triggered Re-evaluation

This scenario describes how Hestia, and particularly the monitoring component, is self-triggered to discover trust changes (Figures 4.10, 4.12). The description is in the form of steps that are executed sequentially as follows:

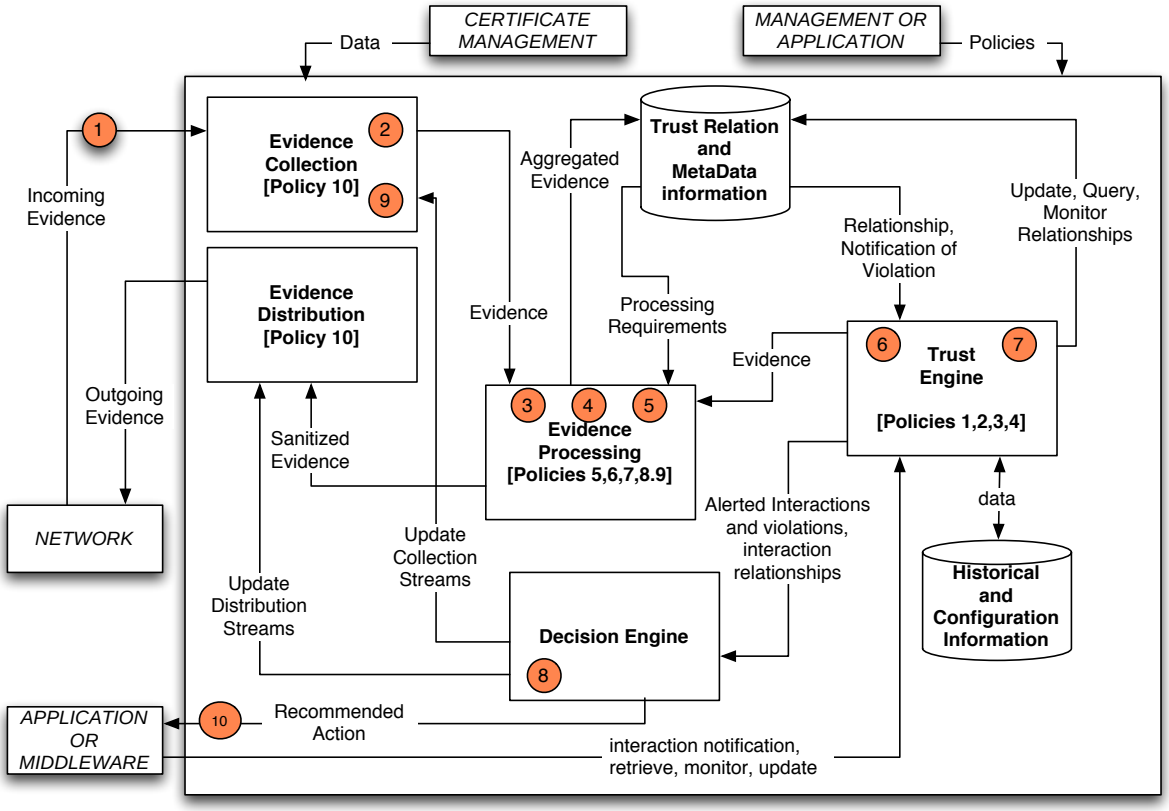

Figure 4.11: Hestia Arrival of Evidence

1. The monitoring module of the Trust Engine component detects that the certificate of trustee *Y* is to expire in *24 hours*.

2. The Trust Engine component forwards a potential violation to the Decision Engine component.

3. The Decision Engine requests from the application to retrieve a new certificate for trustee *Y*.

4. The request is forwarded and processed at the application.

5. The application cannot retrieve a certificate for trustee *Y* (i.e. Y has not renewed its certificate).

6. The Trust Engine component marks trustee *Y* as an unauthenticated entity upon expiration of the certificate. The status of the relationship is set to *ALERT*.

7. The new evidence is forwarded to the Evidence Processing component in order to be distributed to the network.

8. The Evidence Processing forwards it to the Evidence Distribution component.

9. The evidence is labeled as a complaint of "not able to authenticate entity Y".

10. The complaint is distributed to the interested parties.

4.4.3 Scenario 3: End-to-end Trust Composition

The last scenario illustrates how the end-to-end trust for an interaction, which is characterized by a chain of trust relationships, is handled. Consider interaction *interaction1* that allows trustee *Y* access to *data1*, through servers *S1* and *S2* (Figure 4.13). Suppose that the certificate for *S1* gets invalided (look the scenario for self-triggered re-evaluation above). In this case, the status of the relationship between the current trustor and the server *S1* is set to *ALERT*. All other relationships

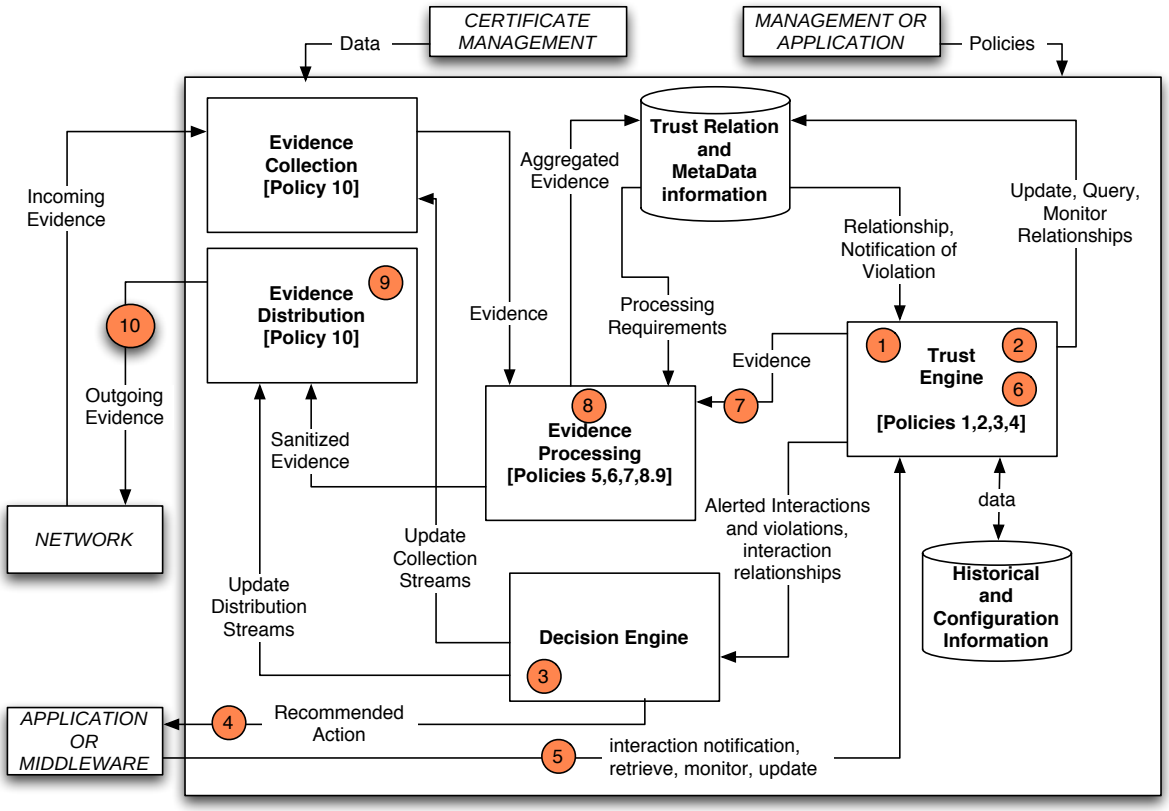

Figure 4.12: Hestia Self-triggered Re-evaluation

Trustee	Trust Relationship MetaData							Trust Relationship Parameters					Other
id	expectation	triggering	aggregation	temp evidence	expectation coverage	interaction	status	context	interval	expectation actual	expectation allowed	trust level	termination reason
Y	competence	on arrival	average of competence (X,Z)	(X, very good)	strict	interaction 1	OK	access data1	cert valid time	(competence , good)	(competence ,>=, good)	TL1, TA1	N\A
S1	authenticity	on arrival	certificate from CA1	none	strict	interaction 1	OK	access data1	cert valid time	(auth. token, certificate)	(auth. token, =, certificate)	TL1, TA1	N\A
S2	authenticity	on arrival	certificate from CA1	none	strict	interaction 1	OK	access data1	cert valid time	(auth. token, certificate)	(auth. token, =,certificate)	TL1, TA1	N\A

Figure 4.13: Hestia TRMI Database for Scenario 3

that are associated with that interaction are set to *WARNING*, a type of indirect alert, because we don't want to trigger the invalidation of other interactions that these relationships are involved with. Assume that the policy dictates that an interaction gets invalided when one or more of its pairwise relationships get invalided.

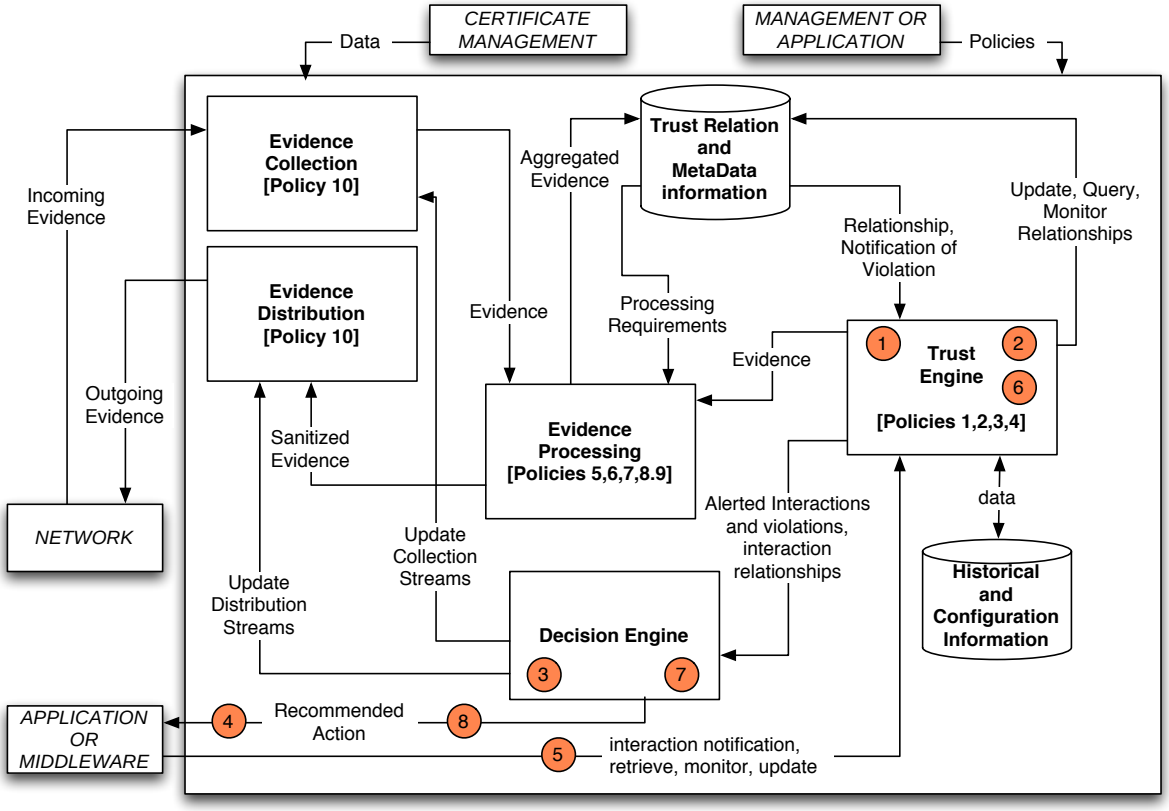

Figure 4.14: Hestia End-to-end Trust Composition

The description is in the form of steps, illustrated in Figure 4.14, that are executed sequentially as follows:

1. The monitoring module of the Trust Engine component detects that the certificate of server *S1* is to expire in *24 hours*.

2. The Trust Engine component forwards a potential violation to the Decision Engine component.

3. The Decision Engine requests from the application to retrieve a new certificate for server *S1*.

4. The request is forwarded and processed at the application.

5. The application cannot retrieve a certificate for server *S1* (S1 did not renew its certificate).

6. The status of the relationship between the current trustor and the server *S1* is set to *ALERT*. All relationships that are involved in *interaction1* are set to *WARNING*.

7. The Trust Engine forwards the alerted interaction to the Decision Engine, which decides to remove the access privileges for trustee *Y*.

8. The decision is forwarded to the application.

CHAPTER 5

HESTIA POLICIES

Hestia supports low-level configurable policies that determine component functionality. These policies are organized in a hierarchical manner for efficiency reasons. This chapter explains the hierarchical policy space as well as the specification of these low-level policies.

5.1 Hestia Components and Sample Policies

Chapter 4 discussed the Hestia components and the need for configurable policies that determine the functionality of these components. Below, is the textual description of such policies (note that the policy number corresponds to the policy number illustrated in Figure 4.3):

Policy 1 Expectation Set Specification

Description Specify expectation sets for trust relationships.

General Policy Accessing non-sensitive data requires authentication.

Specialized Policy Accessing non-senstivive data by INTER domain entities requires authentication and behavior set to "good".

Policy 2 Expectation Covering Technique

Description Specify the expectation covering technique that is to be applied on an expectation set.

General Policy Strict satisfaction must be applied on expectation sets for all INTRA domain entities.

Specialized Policy Relaxed satisfaction must be applied on all expectation sets for entity Y.

Policy 3 Expectation Violation Handling

Description Specify the action that is taken whenever an expectation violation occurs.

General Policy Whenever an expectation violation occurs, re-evaluate the trust relationship immediately.

Specialized Policy Whenever an expectation violation occurs for non-sensitive data, keep monitoring for X minutes before re-evaluation.

Policy 4 Trust Relation Operations

Description Specify rules that govern the operations that the trust engine performs on trust relationships.

General Policy A terminated relationship residing in the Trust relation and MetaData Information Database becomes experience after a predefined interval X.

General Policy A relationship is valid as long as the trustee's certificate is valid.

Policy 5 Evidence Mapped to Expectation

Description Specify the rules that map evidence to expectation.

General Policy Behavior is a function of reliability and cooperation indicator.

Specialized Policy Behavior for any LAN entity is a function of reliability.

Policy 6 Evidence Aggregation

Description Specify the aggregation method for evidence such as recommendations.

General Policy An entity's behavior is the result of aggregating feedback from at least 10 INTRA recommenders using the average function.

Specialized Policy A LAN entity's behavior is the result of aggregating feedback from at least 10 LAN recommenders using the ma jority function.

Policy 7 Triggering Rules

Description Specify the triggering algorithm for the evidence aggregation process.

General Policy The average aggregation algorithm is triggered at predefined interval X.

Specialized Policy The average aggregation algorithm is triggered either at predefined interval X or when at least one sample has been received from all recommenders.

Policy 8 Incoming Evidence Sanitization

Description Discard evidence prior to processing

General Policy Whenever the system operates in attack mode, use only evidence from INTRA recommenders.

Specialized Policy Whenever the system operates under attack mode, discard all evidence from all recommenders for relationships that involve sensitive data.

Policy 9 Outgoing Evidence Sanitization

Description Filter evidence (generated at the local trust module) that is to be disseminated to network entities.

General Policy Disseminate recommendations about the behavior of any INTRA entity.

Specialized Policy Do not disseminate recommendations about the behavior of entity Y.

Policy 10 Recommenders Identification

Description Identify the recommenders that provide feedback for trust relationships.

General Policy Any LAN entity may serve as a recommender for any INTRA entity.

Specialized Policy Entity Y may serve as a recommender for any INTRA or INTER entity.

5.2 Space of Hestia Policies

Hestia's provision of configurable trust policies allows entities to set up policies for the functionality of Hestia components, such as evidence sanitization and aggregation. These low-level trust policies are organized in a four-dimensional space, which is bounded by the trustee location dimension, the context dimension, the task dimension and lastly the imputed trust mode dimension, as shown in the Figure 5.1. For illustrational purposes, the 4-D space is mapped to a 3-D space by keeping imputed trust mode a constant value throughout the entire space.

The *location* dimension categorizes trustees based on their group memberships in relation to the trustor. There are three default groups:

- LAN: refers to trustees belonging to the same LAN as the trustor
- INTRA: refers to the entities belonging to the same administrative domain
- INTER: refers to the entities belonging to any other domain.

Entities may further be organized in various groups within a LAN, or an administrative domain. The degenerated form of a group is, naturally, an individual entity.

The *context* dimension represents the permissible operations performed on data. These will be explained in detail in a later chapter.

The *task* dimension represents various tasks that are necessary for the correct functionality of Hestia components. These tasks include, but are not limited to expectation specification, expectation covering technique, expectation violation handling, trust relation operations, evidence mapped to expectation functions, evidence aggregation technique, triggering rules, incoming evidence sanitization, outgoing evidence sanitization, and recommenders identification.

Finally, the *imputed trust mode* dimension represents the various operational modes of the system.

This trust policy space is populated by high-level policies, which are policies that dictate the

proper usage of an organization's information. High-level policies are essentially translated into practice via the low-level policies.

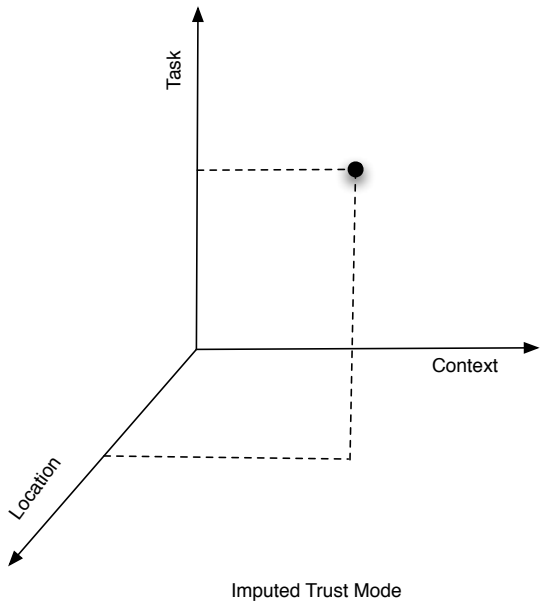

Figure 5.1: Hestia Trust Policy Space

Each point in this space represents a policy for the particular dimension values. These are default policies, which can be overridden to apply to specific groups of trustees, specific contexts and specific subtasks. As a result, a policy may be expanded to cover more specific points in all dimensions to reflect restrictions to the more general default policy. The evaluation of the policies is performed in a bottom-up manner.

Figure 5.2 illustrates how default policies can be overridden so that they are applied to specific groups of trustees. Consider the policy for the evidence aggregation method that is applicable to LAN entities that access sensitive data. For any LAN entity that accesses sensitive data, the default aggregation method is *function0*. This default policy is overridden for entities that belong to either *workgroup1* or *workgroup2*, where the aggregation methods are *function1* and *function2* respectively. These restricted policies are overridden for specific members of *workgroup1*. As a result, entities *P1* and *P2* are subject to aggregation methods *functionP1* and *functionP2* respectively.

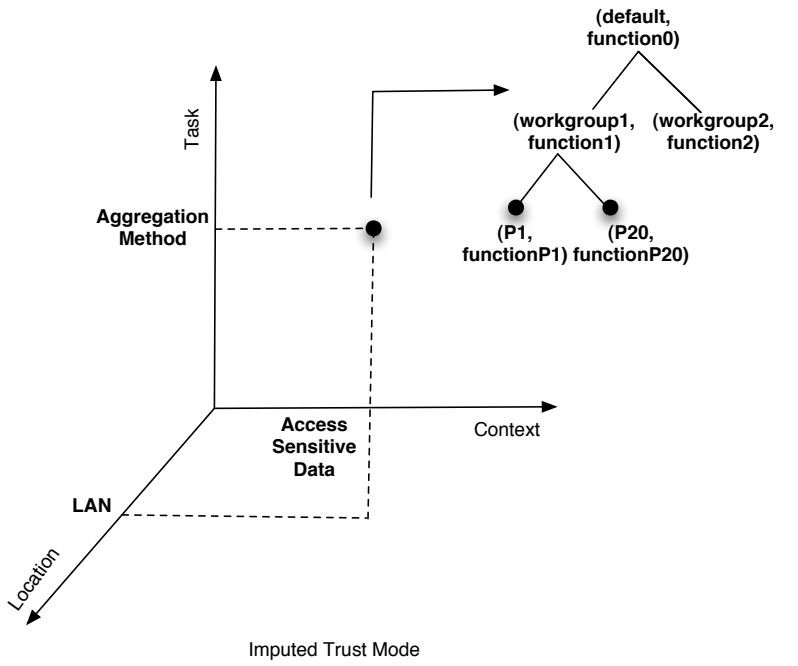

Figure 5.2: Hestia Trust Policy Space Example 1

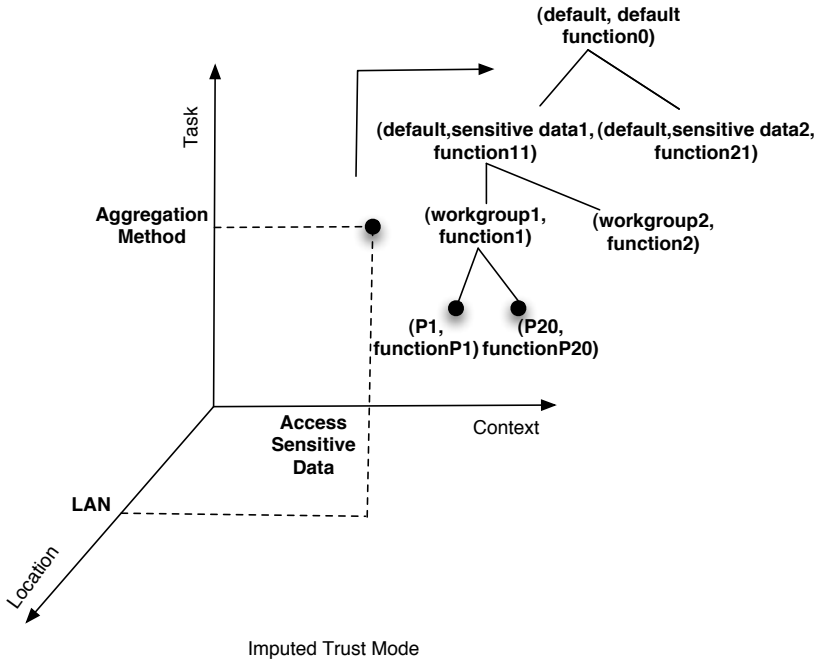

Figure 5.3: Hestia Trust Policy Space Example 2

Consider Figure 5.3, which again illustrates the trust policy space for the evidence aggregation method regarding LAN entities that access sensitive data. In this example, the policies are restricted not only by varying the trustee group membership, but by considering particular sensitive data as well. For any LAN entity that accesses any sensitive data, the default aggregation method is *function0*. For any LAN entity that accesses sensitive *data1*, then the aggregation method becomes *function11*. The interpretation of the subtree rooted at this point is the same as the one of the previous example.

5.3 Hestia Policy Specification

In order to have a better understanding of the Hestia trust policies, sample policies are described in this section. It is assumed that policies are bundled together for a specific imputed trust mode. First, a textual description summary of the policy is provided, followed by the XML schema that captures the semantics of the policy and an example XML policy. For clarity reasons, each policy is specified in its own schema, but with very slight modifications a single policy schema could be obtained. These policies are provided at the initial phase of initialization of the system but it could also be possible to support policy adaptability during runtime.

The policy specification is at the initial stages and it is out of the scope of this dissertation work to verify that there no conflicts between the policies. It is still open research how to define policies for appropriate use of an organization's information resources. It is also nontrivial to translate these policies into practice and provide guarantees that there are no inconsistent policies.

5.3.1 Expectation Specification Policy

The Expectation Specification policy shown in Figure 5.4, as the name implies, specifies expectation sets for particular operations on data.

A general policy could be:

"Accessing public data requires authentication"

and a specialized policy could be:

"Accessing public data by INTER domain entities that belong to the competitors group requires authentication and a behavior indicator of grater than good".

The two policies are illustrated in Figure 5.5.

```
<?xml version="1.0" encoding='us-ascii'?>
<!ELEMENT xml_policy_1 ANY>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT policyname EMPTY>
<!ATTLIST policyname
        name            CDATA                   #REQUIRED
        trustmode       (red | orange | green)  #REQUIRED
>

<!-- Information about the requirements. -->
<!ELEMENT requirementList (requirements*) >
<!ELEMENT requirements (requirement+)>
<!ATTLIST requirements
        location        (lan | inter | intra | any_location )                   #REQUIRED
        trusteeid       CDATA                                                   " "
        type            (sensitive | public | private | evidence | alert | all ) #REQUIRED
        operation       (use | access | all)                                    #REQUIRED
>

<!ELEMENT requirement EMPTY >
<!ATTLIST requirement
        requirementname     CDATA                                                       #REQUIRED
        operand             (lessthan | greaterthan | eq | lessoreq | grateroreq | noteq) #REQUIRED
        value               CDATA                                                       #REQUIRED
>
```

Figure 5.4: XML Schema for Hestia Expectation Specification Policy

5.3.2 Expectation Covering Technique Policy

The Expectation Covering Technique policy shown in Figure 5.6 specifies the expectation covering technique that is to be applied on an expectation set.

A general policy could be:

"Strict satisfaction is to be applied on expectation sets for all INTER domain entities that access sensitive data"

and a specialized policy could be:

"Relaxed satisfaction is applied on all expectation sets for entity Y for accessing all data types. The permissible deviation is [-2,2] for all requirements in these sets".

The two policies are illustrated in Figure 5.7.

```
<?xml version="1.0"?>
<xml_policy_1>
<policyname>
        <name="expectation specification" trustmode = yellow>
</policyname>

</requirementList>
        <requirements>
                <location =  any_location type = public operation = access >
                <requirement>
                        <requirementname = "authentication" operand = "eq" value = "1">
                </requirement>
        </requirements>

        <requirements>
                <location =  inter trusteeid = "competitor" type = public operation = access >
                <requirement>
                        <requirementname = "authentication" operand = eq value = "1">
                </requirement>
                <requirement>
                        <requirementname = "behavior" operand = greaterthan value ="good">
                </requirement>
        </requirements>
</requirementList>

</xml_policy_1>
```

Figure 5.5: Hestia Expectation Specification Policy

5.3.3 Trust Relationship Violation Handling Policy

The Expectation Violation Handling policy shown in Figure 5.8 specifies the action that is taken whenever there is either time violation or expectation violation for a trust relationship.

A general policy could be:

"Whenever an expectation violation occurs for accessing sensitive data by any entity, re-evaluate the particular trust relationship immediately".

and a specialized policy could be:

"Whenever a violation occurs for accessing any data by INTRA entities, keep monitoring for X minutes".

The two policies are illustrated in Figure 5.9.

```
<?xml version="1.0" encoding='us-ascii'?>
<!ELEMENT xml_policy_2 ANY>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT policyname EMPTY>
<!ATTLIST policyname
        name            CDATA                   #REQUIRED
        trustmode       (red | orange | green)  #REQUIRED
>

<!-- Information about the satisfaction covering technique. -->
<!ELEMENT expectationcovering (covering*) >
<!ELEMENT covering (deviationlist*)>
<!ATTLIST covering
        location        (lan | inter | intra | any_location )                       #REQUIRED
        trusteeid       CDATA                                                       " "
        type            (sensitive | public | private | evidence | alert | all )     #REQUIRED
        operation       (use | access)                                               #REQUIRED
        technique        (covering | strict_covering | selective_covering
                        | preferred_covering | relaxed | relaxed_individual)       #REQUIRED
>

<!ELEMENT deviationlist EMPTY>
<!ATTLIST deviationlist
        requirement     CDATA       #REQUIRED
        low_bound       CDATA       #REQUIRED
        upper_bound     CDATA       #REQUIRED
>
```

Figure 5.6: XML Schema for Hestia Expectation Covering Technique Policy

```
<?xml version="1.0"?>
<xml_policy_2>
<policyname>
        <name="expectation covering technique" trustmode = yellow>
</policyname>


<expecationcovering>
        <covering>
                <location = inter type = sensitive operation = access technique =strict_covering>
        </covering>
        <covering>
                <location = inter trusteeid = "Y" type = all operation = access technique =
relaxed_covering>
                        <deviationlist>
                                < requirement = "all" low_bound = "-2" upper_bound = "2" >
                        </deviationlist>
        </covering>

</expectationcovering>

</xml_policy_2>
```

Figure 5.7: Hestia Expectation Covering Technique Policy

```
<?xml version="1.0" encoding='us-ascii'?>
<!ELEMENT xml_policy_3 ANY>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT policyname EMPTY>
<!ATTLIST policyname
        name            CDATA                   #REQUIRED
        trustmode       (red | orange | green)  #REQUIRED
>

<!-- Information about the violation handling technique. -->
<!ELEMENT violationhandling (handling*) >
<!ELEMENT handling EMPTY>
<!ATTLIST handling
        location        (lan | inter | intra | any_location )                   #REQUIRED
        trusteeid       CDATA                                                   " "
        type            (sensitive | public | private | evidence | alert | all ) #REQUIRED
        operation       (use | access)                                          #REQUIRED
        violationtype   (interval_expiration | expectation_violation | all)     #REQUIRED
        action          (re_evaluate | monitor | ignore)                        #REQUIRED
>
```

Figure 5.8: XML Schema for Hestia Trust Relationship Violation Handling Policy

```
<?xml version="1.0"?>
<xml_policy_3>
<policyname>
        <name="trust relationship violation handling" trustmode = yellow>
</policyname>

<violationhandling>
        <handling>
                <location = any_location type = sensitive operation = access violationtype = expectation_violation
                 action = re_evaluate >
        </handling>
        <handling>
                <location = intra type = all  operation = access violationtype = all action =monitor >
        </handling>
</violationhandling>

</xml_policy_3>
```

Figure 5.9: Hestia Trust Relationship Violation Handling Policy

5.3.4 Evidence Mapped to Expectation Policy

The Evidence Mapped to Expectation policy shown in Figure 5.10 specifies the rule that dictates how raw evidence is applied to expectations. Note that trust has properties, but a trustor has expectations. The collected evidence is not meaningful unless a trustor decides how it could be applied to its expectations.

A general policy could be:

"For all LAN entities, regardless of data operation and type, behavior is a function of recommendations only".

and a specialized policy could be:

"For all INTER entities, regardless of data operation and type, behavior is a function of recommendations and cooperation indicator ".

The two policies are illustrated in Figure 5.11.

```
<?xml version="1.0" encoding='us-ascii'?>
<!ELEMENT xml_policy_4 ANY>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT policyname EMPTY>
<!ATTLIST policyname
        name            CDATA                   #REQUIRED
        trustmode       (red | orange | green)  #REQUIRED
>

<!-- Information about the evidence mapped to expectation technique. -->
<!ELEMENT evidencemapping (mapping*) >
<!ELEMENT mapping (evidence+)>
<!ATTLIST mapping
        location        (lan | inter | intra | any_location )                   #REQUIRED
        trusteeid       CDATA                                                   " "
        type            (sensitive | public | private | evidence | alert | all ) #REQUIRED
        operation       (use | access | all)                                    #REQUIRED
        expectation     CDATA                                                   #REQUIRED
        mapfunction     CDATA                                                   #REQUIRED
>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT evidence EMPTY>
<!ATTLIST evidence
        evidencename  CDATA                     #REQUIRED
>
```

Figure 5.10: XML Schema for Hestia Evidence Mapped to Expectation Policy

```
<?xml version="1.0"?>
<xml_policy_4>
<policyname>
        <name="evidence mapped to expectation" trustmode = yellow>
</policyname>

<evidencemapping>

        <mapping>
                <location = lan type =all operation = all expectation = "behavior" mapfunction = "function1">
                <evidence>
                        <name = recommendations>
                </evidence>
        </mapping>
        <mapping>
                <location = inter type =all operation = all expectation = "behavior" mapfunction = "function2">
                <evidence>
                        <name = "recommendations">
                </evidence>
                <evidence>
                        <name = "cooperation_indicator">
                </evidence>
        </mapping>

</evidencemapping>

</xml_policy_4>
```

Figure 5.11: Hestia Evidence Mapped to Expectation Policy

5.3.5 Triggering Rules Policy

The Triggering Rules policy shown in Figure 5.12 specifies the event that triggers evidence aggregation.

A general policy could be:

"The aggregation algorithm is triggered at predefined interval X".

and a specialized policy could be:

"For all INTER entities, the aggregation algorithm is triggered at predefined interval X or when at least one sample has been received by Y recommenders".

The two policies are illustrated in Figure 5.13.

```
<?xml version="1.0" encoding='us-ascii'?>
<!ELEMENT xml_policy_5 ANY>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT policyname EMPTY>
<!ATTLIST policyname
        name            CDATA                   #REQUIRED
        trustmode       (red | orange | green)  #REQUIRED
>

<!-- Information about the evidence mapped to expectation technique. -->
<!ELEMENT triggeringList (triggering*) >
<!ELEMENT triggering (parameters+)>
<!ATTLIST triggering
        location        (lan | inter | intra | any_location )                   #REQUIRED
        trusteeid       CDATA                                                   " "
        type            (sensitive | public | private | evidence | alert | all ) #REQUIRED
        operation       (use | access | all)                                    #REQUIRED
>

<!-- Specifies the name of the policy. This is essentially the task. -->
<!ELEMENT parameters EMPTY>
<!ATTLIST parameters
        triggering              (predefined | arrival_of_samples)               #REQUIRED
        intervalORsamples       CDATA                                           #REQUIRED
>
```

Figure 5.12: XML Schema for Hestia Triggering Rules Policy

```
<?xml version="1.0"?>
<xml_policy_5>
<policyname>
        <name="triggering rules" trustmode = yellow>
</policyname>

<triggeringList>

        <triggering>
                <location = any_location type =all operation = all>
                <parameters>
                        <triggering = predefined intervalORsamples = "X">
                </parameters>
        </triggering>
        <triggering>
                <location = inter type =all operation = all>
                <parameters>
                        <triggering = predefined intervalORsamples = "X">
                </parameters>
                <parameters>
                        <triggering = arrival_of_samples intervalORsamples = "Y">
                </parameters>
        </triggering>

</triggeringList>

</xml_policy_5>
```

Figure 5.13: Hestia Triggering Rules Policy

CHAPTER 6

TRUST FORMALISM

This chapter presents the formal specification for trust relationships for indirect interactions using the theory of sets and relations. Formal specifications use mathematical notation to describe in a precise manner the properties which a system must have. They describe what the system must do without saying how it is to be done (without the fine details of data structures and algorithms.) This approach eliminates the details of programming languages but at the same time it offers enough precision to describe how the trust tasks are accomplished. Due to the fact that it is independent of the program code, a formal specification of trust can be completed early in the development.

6.1 Trust Ontology

The theory of sets and relations is used to represent the trust relationships between trustors and trustees. In particular, this chapter formally defines *"trust between trustors and trustees"* as a relation τ and examines the relation's properties, operations and any other observations. Trust relation has static aspects, such as properties and attributes (along withe their properties). The dynamic properties include operations and the changes of state that happen.

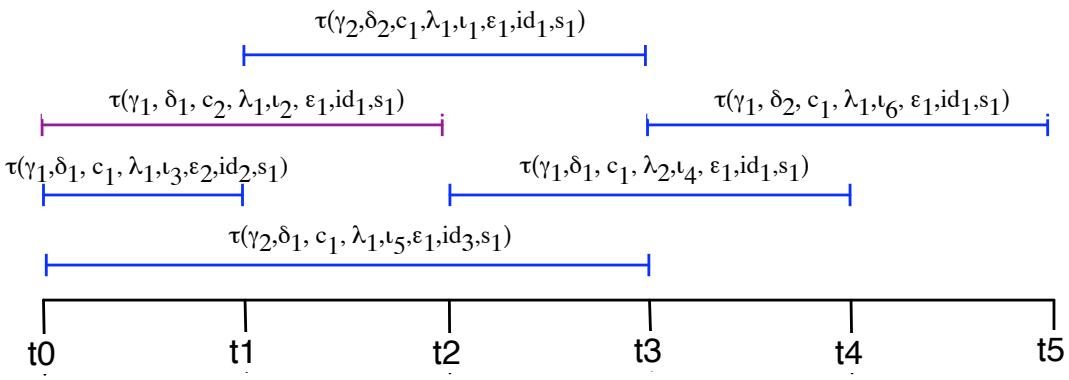

Figure 6.1: Trust Relationships Timeline

The attributes of the trust relation τ are trustor γ, trustee δ, context c, levels λ, time ι, expectations ϵ, interaction identifier id, and status s. Figure 6.1 illustrates six trust relationships which hold true in τ. In order to understand the figure, it is sufficient, for now, to know that $\tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ is a trust relationship between two entities and it is interpreted as "*trustor γ, based on the current imputed trust mode, believes that the extent to which trustee δ will act as expected for context c during time interval ι is λ, and this belief is subject to the satisfaction of expectation set ϵ. This relationship is valid for a specific lifecycle stage and interaction id and its status is indicated by s.*" The current imputed trust is included in λ.

The following subsections explain in detail the relation's attributes. To be more specific, a formal definition is given for each attribute along with complementary definitions, properties, and operations related with that attribute. The relation's properties are described next, followed by operations that change the state of the relation and observations. Two theorems are presented, followed by operations that affect the relation indirtectly. Figure 6.2 is an informal overview of the formalisms presented in this chapter.

6.1.1 Trustors and Trustees

The first two attributes γ, δ of trust relation τ represent the trustor and trustee respectively. These are entities with unique identifiers. It is assumed that there is a naming and discovery service that returns unique identifiers for all Hestia entities. Since the trust relation is maintained locally by each Hestia entity (which is the trustor), the trustor attribute γ might be considered redundant. However,we chose to include it as a trust parameter for completeness reasons. In addition, there might be cases where all local trust relations will be merged to a central place in order to obtain a snapshot of trust relationships at a wide-area level.

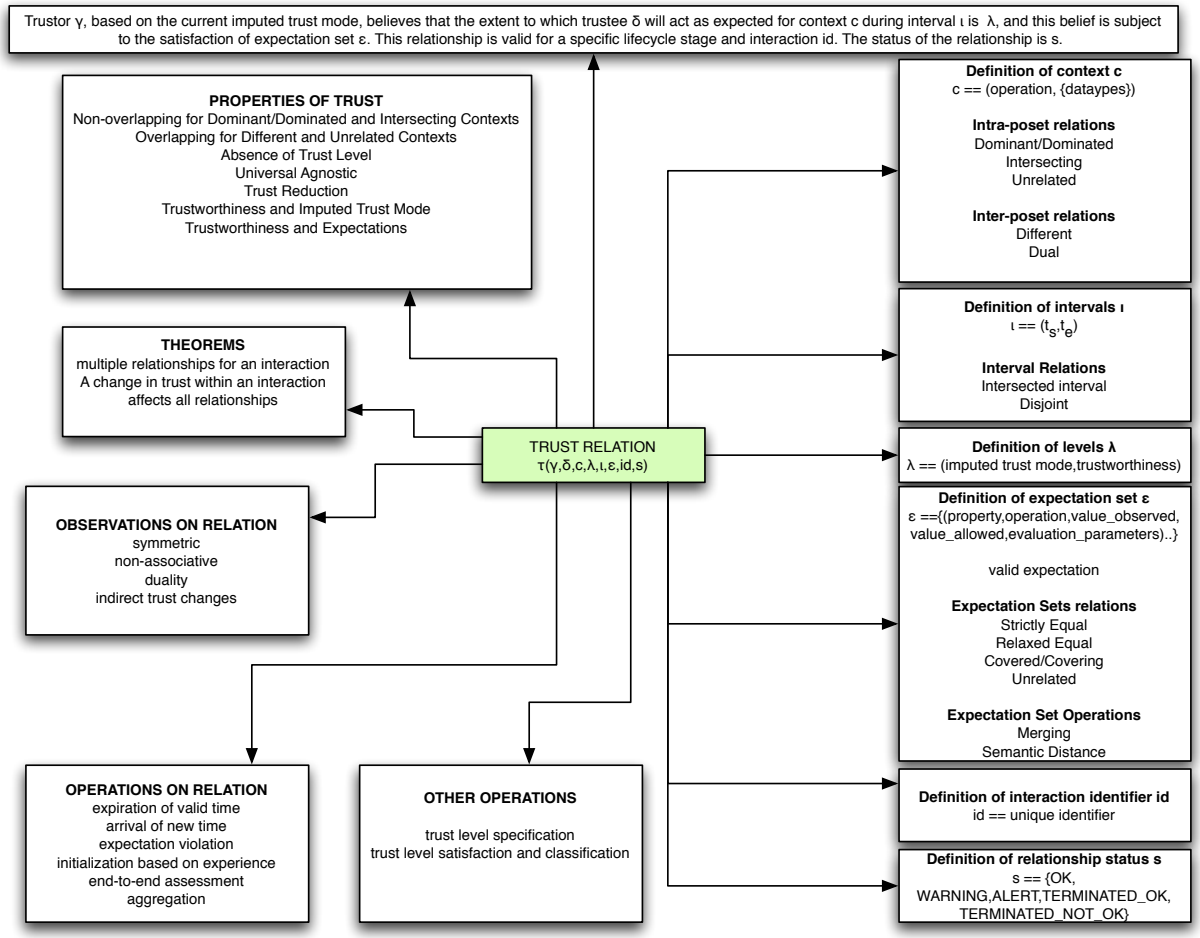

Figure 6.2: Trust Formalism Overview

6.1.2 Context

The scope of the trust relationship is narrowed to a specific activity or environment called context. Our model portrays context c as an action or operation performed on data during its lifecycle. Consider A to be the set of operations performed on data and D to be the set of data types. $DataTypes$ is the power set of D. Context c is defined as follows:

Definition 6.1.1 Context *Context c is an ordered pair $(operation, datatypes)$, with $operation \in A$ and $datatypes \in DataTypes$. $DataTypes \in \wp D$, with D to be the set of valid data types.*

For example, consider the case where A is the set $\{consume, provide\}$ and D consists elements $\{normal, alert, recommendation\}$. Example contexts include the pairs $(provide, \{normal\}), (provide, \{alert\})$, and $(consume, \{normal, alert\})$.

The set of all contexts is denoted by C and it is defined as a subset of the cartesian product $A \times DataTypes$. Set C is organized in partially ordered sets (posets), one for each operation, under the relation of set inclusion $\subseteq$ on $DataTypes$. Figure 6.3 illustrates two posets for C, given the declarations of A and D above.

We distinguish context relations within a poset (intra-poset) and between different posets (inter-poset). Contexts belonging to the same poset are associated via dominating, intersecting or unrelated relations. On the other hand, contexts from two different posets are related if their respective actions are considered to be *complimentary*. For example, *produce* and *consume* are complementary actions because they are associated with two different information lifecycle stages.

Starting with intra-poset relations, two contexts c_1 and c_2 may be related, by definition, under $\subseteq$. In that case, context c_1 is a superset of c_2 and c_1 is labeled as *dominant* context. Similarly, c_2 is the dominated context. As an example, consider contexts $c_1 = (provide, \{normal, alert\})$ and $c_2 = (provide, \{normal\})$ with the former being the dominant context and the latter being the dominated one.

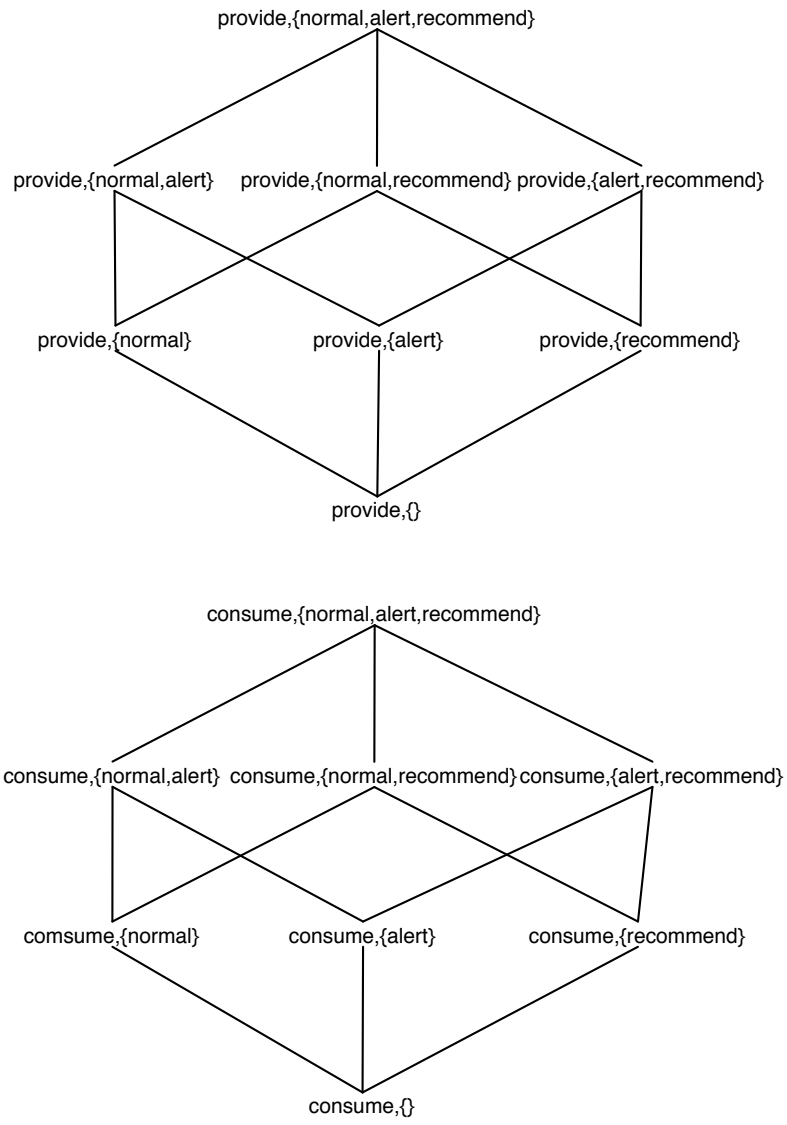

Figure 6.3: Posets For Context Set C

Definition 6.1.2 Dominant/Dominated Contexts *Context c_1 dominates context c_2 if and only if $c_2 \subset c_1$ in the same poset. In this case c_1 is the dominant context and c_2 is the dominated context.*

Contexts c_1 and c_2 may be unrelated in a poset under the $\subseteq$ relation. Take as an example contexts $(provide, \{normal, alert\})$, $(provide, \{alert, recommendation\})$. However, the intersection of their respective datatypes sets outputs the singleton set $\{alert\}$. From this fact, it can be deduced that unrelated contexts may be related under a secondary relation of intersection $\bigcap$ within the poset.

Definition 6.1.3 Intersecting Contexts *Context c_1 intersects with context c_2 if and only if $c_1 \subset c_2$ and $c_2 \subset c_1$ do not hold in the same poset but $c_1 \bigcap c_2 \neq \emptyset$.*

It is trivial to observe that Dominant/Dominated contexts are also Intersecting contexts but not vice versa. Finally, there is the case where contexts are simply unrelated in a poset. Any two singleton contexts fall into this category.

Definition 6.1.4 Singleton Context *Context c is a singleton context if and only if its datatype set is singleton.*

Definition 6.1.5 Unrelated Contexts *Context c_1 and context c_2 are unrelated if and only if they are not related via Dominant/Dominated relation or Intersecting relation.*

Inter-poset relations categorize contexts as different or dual. Contexts that are members of different posets are automatically labeled as different. Contexts $(provide, \{normal, alert\})$ and $(consume, \{normal\})$ are two different contexts.

Definition 6.1.6 Different Contexts *Different contexts refer to contexts c_1 and c_2 from different posets.*

A specialization of the above inter-poset relation is the case where operations are considered to be paired. For example, *provide* and *consume* constitute a pair. Dual contexts are different contexts whose actions are paired for the intersection of their datatypes sets.

Definition 6.1.7 Dual Contexts *Dual contexts refer to contexts c_1 and c_2 from different posets whose first coordinates are paired actions. Duality is valid on the set produced by $c_1 \bigcap c_2$, provided that the intersection is not the empty set $\emptyset$.*

6.1.3 Trust Levels

The next trust relation attribute is the trust levels λ, which is an ordered pair of two levels $(\lambda_{im}, \lambda_t)$. Trust is subjective for a couple of reasons: a trustor's requirements are not met by trustees at the same degree and a trustor's expectations for a trustee differ based on the imputed trust mode. The current literature provides a number of ways to denote how much an entity is worthy of trust. Trust values [7], degrees [6] and levels [26] are all used by the trustor to categorize trustees based on their perceived trustworthiness. Our model adopts the term *trust level* and it is defined below:

Definition 6.1.8 Level *Trust level is a label supported by a trust classification system that is used for labeling trustworthiness extent and trustfulness extent (imputed trust mode).*

Definition 6.1.9 Trust Levels *Trust levels λ is an ordered pair $(\lambda_{im}, \lambda_t)$ with $\lambda_{im} \in \Lambda_{im}$, $\lambda_t \in \Lambda_t$, and its coordinates represent trustfulness extent and trustworthiness extent respectively. Λ_{im} is the set of trust levels values for trustfulness extent whereas Λ_t is the set of trust levels values for trustworthiness extent. Sets Λ_{im} and Λ_t could be equal sets.*

Current trust level classifications, shown in Figure 6.4(a), resemble a chain with total ordering under the $\leqslant$ operator. A chain structure allows a hierarchical ordering of trust levels such that there is always a dominance relation between any two levels. For example, a *partial trust* level dominates a *partial distrust* level.

Our trust level organization does not follow the total ordering scheme. According to Ullmann-Margalit [44], trust and distrust are mutually exclusive but are not mutually exhaustive as they don't complement each other. There is a state of being agnostic in the matter of trusting or distrusting. The new trust level classification is based on two observations that are derived from the above claim:

1. The agnositc level is portrayed as the lack of evidence that prohibits the proper classification of a trust relationship. It is not portrayed as the deliberate decision to ignore a trust relationship. The former case corresponds to the non-intentional ignorance (agnostic) whereas the latter deals with the intentional ignoring, or neutral. Neutrality is not a synonym for distrust. It only indicates an entity's unwillingness to form any type of trust relationships with a trustee for a number of reasons; i.e there is no gain from the interaction or the other entity will benefit from the interaction but harmful consequences for members of the community.

2. The second observation is that the agnostic state serves as the starting point of a bi-directional system leading to either trust or distrust (Figure 6.4(b)).

The new trust classification system of Figure 6.4(c) incorporates the two observations mentioned above. It is a partially ordered lattice where not all levels are comparable. A trust relationship between two entities could be initially assigned a trust level of agnostic without making any unsafe assumptions. From there, there are two directions to follow; the trust or distrust paths with intermediate levels in between. No matter the direction, the dominant level is neutrality. The main difference between our trust labeling system and others is that we don't consider distrust as being dominated by trust because such an ordering is situation-dependent and based on the evaluator's policies. In addition, distrust protects an entity from potential dangers and is not always associated with a breach of trust.

Trust levels are closely related to two important concepts: trustfulness of the trustor and trustworthiness of the trustee [13]. Trustfulness is defined as the extent to which the trustor is willing to

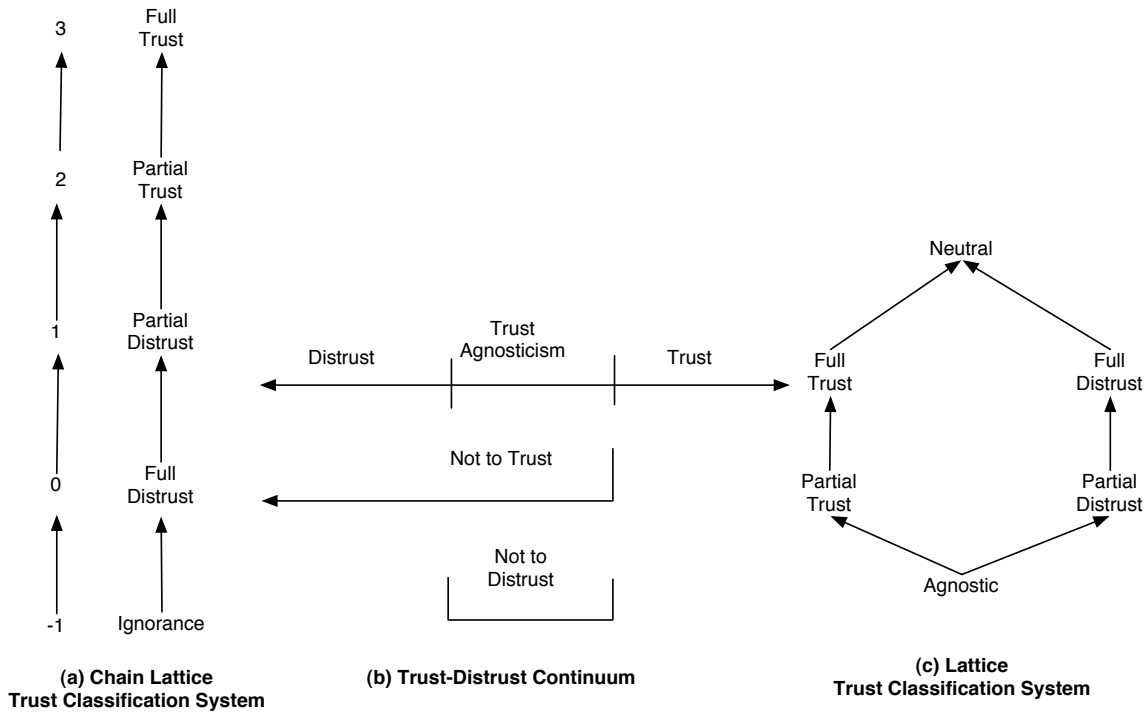

Figure 6.4: Trust Classification Systems

take the risk of trust being abused by the trustee. On the other hand, trustworthiness is the extent to which the trustee honors trust, if trust is placed. Trustfulness is the trait of trusting whereas trustworthiness is the trait of deserving trust. The trusting attitude of the trustor affects its judgment regarding the trustworthiness of a trustee (and vice versa). The trustfulness of the trustor varies and it depends on the trustor's willingness to trust. In this model, the trustfulness of the trustor is a synonym of imputed trust mode.

A trust level can take either continuous or discrete values. The fine-grained nature of the continuous labeling makes it difficult to map trust into a value that is uniquely distinguished from other values. Consider the trust levels of 2.12 and 2.22. The difference of 0.1 translates into infinite trust levels between the two levels. As a result, the evaluator of the trust relationship might find it non-trivial to choose a different course of action corresponding to each level. If the same actions are taken for all levels, then it is pointless to use any levels at all. Grouping of continuous values

into discrete sets that are identified by a single label is a course-grained approach that allows clearer distinction between trust levels.

6.1.4 Time

A trust relation consists of trust relationships that are valid for a period of time. Therefore, the trust formalism must model the abstract concept of time.

The temporal database interpretation of time is chosen for modeling time. Temporal databases keep track of past and future database states by attaching a time period to the data. There are two different notions of time: valid time and transaction time. Valid time is the period which a fact is true with respect the real world and transaction time is the time when the fact is actually stored in the database. The two different notions of time allow the distinction of different forms of temporal databases:

- Valid Time Database: stores data with respect to valid time. A valid time database allows three types of updates: proactive, retroactive and simultaneous. The categorization depends on the value of the valid end time with respect the time that the fact becomes effective in the real world. A proactive update refers to an update that is applied to the database before it becomes effective in the real world. A retroactive update refers to an update that is applied to the database after it became effective in the real world. Lastly, an update that is applied at the dame time when it becomes effective is a simultaneous update. Due to the fact that updates may be applied retroactively or proactively, there is no record of the actual database state at any point in time.

- Transaction Time Database: stores data with respect to transaction time. The actual timestamp of the transaction that applied the change is recorded. Transaction databases are also called rollback databases because users can logically roll back to actual database states at any past point.

- Bitemporal Database: stores data with respect to both valid time and transaction time.

In temporal databases, time domain T is considered to be an ordered sequence of points t_i in some application-dependent granularity [20]. An interval is an ordered pair of time points, and for our purposes it is an anchored time duration with a fixed starting point. In temporal databases, there is a distinction between interval and period. In a sense, our definition of interval corresponds to what period means in temporal databases. The reason is because the term interval is commonly used in trust literature, in contrast with the term period.

Definition 6.1.10 Interval *An interval ι is an ordered pair of time points (t_s, t_e) with the first point appearing before the second point in the timeline. The set of all intervals I is the cartesian product $T \times T$, with ι representing an element from I.*

According to Allen's algebra [8], there are 13 possible relationships between two intervals ι_1 and ι_2 as shown in Table 6.1. The relationships between the intervals are illustrated in Figure 6.5.

Table 6.1: Interval Relationships

Relationship	Notation	Illustration
ι_1 before ι_2 (and its inverse)	$(\iota_1 < \iota_2)$	XXX YYY
ι_1 equal ι_2	$(\iota_1 = \iota_2)$	XXX YYY
ι_1 meets ι_2 (and its inverse)	$(\iota_1$ m $\iota_2)$	XXXYYY
ι_1 during ι_2 (and its inverse)	$(\iota_1$ d $\iota_2)$	XXXX YYYYYYYYY
ι_1 overlaps ι_2 (and its inverse)	$(\iota_1$ o $\iota_2)$	XXXXXXX YYYYY
ι_1 starts ι_2 (and its inverse)	$(\iota_1$ s $\iota_2)$	XXX YYYYYYYY
ι_1 finishes ι_2 (and its inverse)	$(\iota_1$ f $\iota_2)$	XXX YYYYYY

As it was mentioned earlier, the trust relation is dynamic and changes over time. *Transaction time* is the time point when the current state of the relation changes.

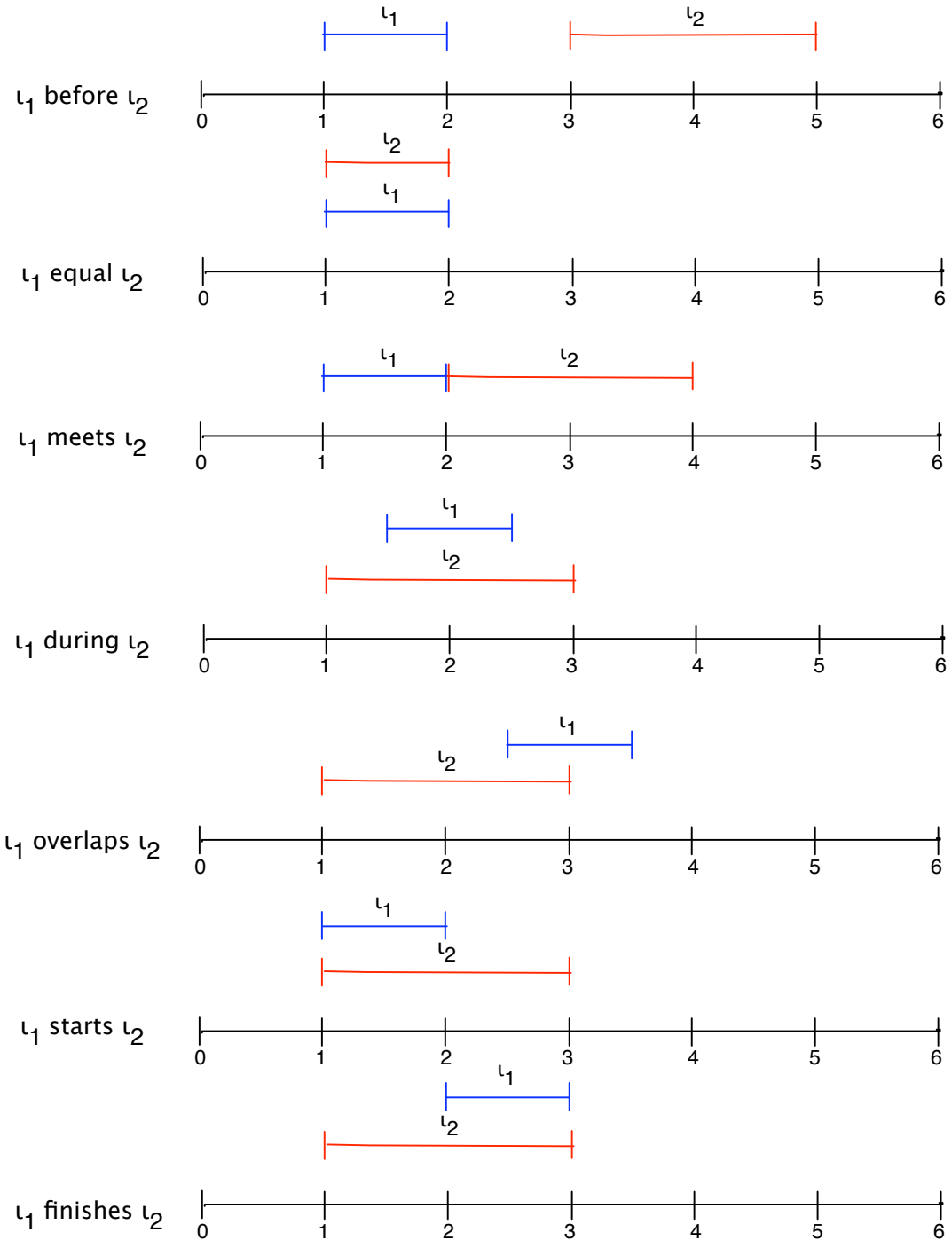

Figure 6.5: Interval Relationships Diagrammatically

Definition 6.1.11 Transaction Time *A transaction time t is a time point at which the state of the trust relation changes.*

Similar to temporal databases, a trust relation stores observed and anticipated trust relationships. Each trust relationship is timestamped with time intervals. The granularity of time does not affect the logic of the trust relation if it is consistent throughout the model. There are two types of intervals: *expected* interval and *actual* interval. The expected interval indicates the anchored time duration in which a trust relationship is predicted to be valid. The actual interval is the one that the trust relationship was observed to be valid in the real world. Note that the actual trust relation τ only keeps track of the expected interval, but when a trust relationship is terminated, then the actual interval is recorded in the *experience* database.

Definition 6.1.12 Expected Interval *An expected interval is the interval that a particular trust relationship is expected to be valid in the real world.*

Definition 6.1.13 Actual Interval *An actual interval is the interval that a particular trust relationship was observed to be valid in the real world.*

The reason for differentiating between the expected and actual intervals is twofold. First, the expected interval is a prediction, an estimation. Hestia estimates expected intervals based on its current knowledge. As time progresses, additional information generates new knowledge that may affect the already predicted expected intervals. In a sense, actual intervals represent the history of trust relationships, which is also known as *experiences*. Second, it is important to record both the expected and actual intervals because the deviation between these two provides feedback to the inference engine. Ideally, the expected and actual intervals should be the same. However, there are scenarios that don't correspond to the ideal case. For example, the predicted interval for a server's availability is [today,today+4 days]. The email announcement of a scheduled server maintenance for tomorrow affects the server's functionality and thus availability. As a result, the trust on that

server's availability changes and possibly gets restored after the maintenance. The assumption is that a new trust relationship is formed when the expected interval $\iota = (t_s, t_e)$ is interrupted at $t > t_e$.

There are some additional restrictions on how to assign intervals for trust relationships:

- The starting time is either the present time or a time point in the future. The transaction time is not necessarily the same as the starting time. A trust relationship may be created proactively because the future is not observed but predicted.
- The ending time is a time point in the future.
- A trust relationship with a recorded actual interval cannot be changed retroactively because it is an observed experience that cannot and must not be changed. Since no retroactive transactions are allowed, past states of the trust relation knowledge base are known.

In addition to the interval relationships presented earlier, the model must also account for the intersected time interval ι_s between intervals related with the equal, during, overlaps, starts and finishes relationships.

Definition 6.1.14 Intersected Intervals *Intervals $\iota_1 = (t_{s1}, t_{e1})$ and $\iota_2 = (t_{s2}, t_{e2})$ are intersected if and only if $(\iota_1 = \iota_2) \vee (\iota_1\ d\ \iota_2) \vee (\iota_1\ s\ \iota_2) \vee (\iota_1\ f\ \iota_2) \vee (\iota_1\ o\ \iota_2)$. The intersected interval ι_s is obtained as follows:*

1. *If $(\iota_1 = \iota_2) \vee (\iota_1\ d\ \iota_2) \vee (\ \iota_1\ s\ \iota_2) \vee (\iota_1\ f\ \iota_2)$, then $\iota_s = (t_{s1}, t_{e1})$*
2. *If $(\iota_1\ o\ \iota_2)$, then $\iota_s = (t_{s1}, t_{e2})$*

It is also important to be able to deduce whether two intervals are disjoint or not.

Definition 6.1.15 Disjoint Intervals *Intervals ι_1 and ι_2 are disjoint intervals if and only if $(\iota_1 < \iota_2) \vee (\iota_1\ m\ \iota_2)$.*

The usefulness of intervals is twofold. First, an interval represents the valid time duration of a trust relationship. This allows for deducing chronological ordering of trust relationships. For example, in Figure 6.1 $\tau(\gamma_2, \delta_2, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ and $\tau(\gamma_1, \delta_1, c_2, \lambda_1, \iota_2, \epsilon_1, id_1, s_1)$ are associated with intervals ι_1 and ι_2 respectively. Since ι_2 o ι_1, it is inferred that $\tau(\gamma_1, \delta_1, c_2, \lambda_1, \iota_2, \epsilon_1, id_1, s_1)$ o $\tau(\gamma_2, \delta_2, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$. Second, intervals are used to model the lifecycle of an information flow, which is central to the trust model. Information flow lifecycle consists of three stages; generation, distribution and consumption. Each stage is associated with an interval that indicates the time that information resides in that stage. Let's call the intervals ι_g, ι_d, ι_c. For simplicity reasons, for a specific information flow, its intervals intersect in time since trust is assessed for a flow rather than for a packet basis.

6.1.5 Expectations

At it was stated earlier, trust is an abstraction of individual beliefs and requirements that an entity has for specific situations and interactions. An expectation is defined as a requirement, or trust property, and its allowed values that a trustor has for a particular interaction with a trustee. The allowed values are constraint by equality and inequality operators. An expectation is a tuple $(\pi, o, \nu_o, \nu_a, ev)$, where π is a trust property, o is a standard equality or inequality operator, ν_o is the observed or actual value for the property, ν_a is the allowed value for that property, and ev represents the evaluation information for the specific property.

The first expectation attribute is the trust property π. Trust properties are organized in a hierarchical manner illustrated in Figure 4.5. Properties are grouped in their respective behavioral (β) (including competence (cp) and motivation (μ)), security (σ), and QoS (ϕ) categories. Consider facet $\kappa \in \{\beta, \sigma, \phi\}$. Properties $\pi_\beta \in \Pi_\beta$, with Π_β to be the set of valid properties for β facet. Properties $\pi_\sigma \in \Pi_\sigma$, with Π_σ to be the set of valid properties for σ facet. Similarly, properties $\pi_\phi \in \Pi_\phi$, with Π_ϕ to be the set of valid properties for ϕ facet. Since the three sets Π_β, Π_σ, and Π_ϕ are disjoint sets and their respective properties are unique, facets are not included in the expectation definition.

Without loss of generality, trust property $\pi \in \Pi$, with $\Pi = \Pi_\beta \cup \Pi_\sigma \cup \Pi_\phi$.

The second expectation attribute is the arithmetic equality or inequality operator o. Operator o takes values from the set O that includes the arithmetic operators $=$, $<$, $\leqslant$, $>$, $\geqslant$, and $\neq$.

The third and fourth attributes are the observed and allowed values respectively. Value $\nu \in V$, with set V consisting of values that are assigned to properties. Set V is partitioned into subsets V_i. A partition of a set S is a collection of nonempty disjoint subsets of S whose union equals S. The subsets making up the partition are formed by grouping together related elements, in this case values that correspond to a particular property π. The set partitioning is achieved by defining equivalence relation ρ on V as $\nu_1 \rho \nu_2$ where ν_1 belongs to the same property as ν_2. Note that elements of set V must be of the same type. Since the subsets are disjoint, values have to be unique. There are cases where two properties are mapped to the same values. In this case, the elements are uniquely distinguished by replacing single value ν with a pair (π, ν). For brevity reasons, we assume that values are unique. The reason for partitioning set V is for determining the minimal value related to a property.

In order to reason between expectations the values must be ordered. Orders are special binary relations, and in our case this relation is the $\leqslant$. Consider set V_i and the relation $\leqslant$ on V_i. The $\leqslant$ is a total order if it is reflexive, antisymmetric, transitive and total. For example, any set of ordinal numbers (numbers used to denote the position in an ordered sequence) is a total-ordered set. Given set $V_i = \{a, b, c, d\}$, the total order is defined by assuming $a \leqslant b \leqslant c \leqslant d$. Without loss of generality, assume that every element of each subset of V is related to every other element using the relation $\leqslant$ on the subsets, yielding a total ordering (chain). Each set has one maximal element and one minimal element. Furthermore, it is also assumed that operators $o \in O$ are already defined for all $\nu \in V$.

Finally, the fifth attribute ev represents the evaluation parameters for the particular property. An element ev is a tuple $(covering, triggering, aggregation)$ that describes the covering method, triggering rule and aggregation scheme for property π. The first attribute provides the terms under

an expectation is consider valid and the remaining two characterize the "when" and "how" the actual value is updated. All three attributes conform to the principles of covering techniques, triggering schemes and aggregation algorithms as explained in Chapter 3. The covering techniques covered in this chapter are *strict* and *relaxed*, where in the former case the observed value satisfies the allowed value under operation o and in the latter case a deviation is allowed.

An expectation set ϵ describes the requirements that a trustor has for a trustee. Formally, such a set is defined in Definition 6.1.16.

Definition 6.1.16 Expectation Set *Expectation set ϵ is subset of the cartesian product $\Pi \times O \times V \times V \times EV$. A member of the expectation set is tuple $(\pi, o, \nu_o, \nu_a, ev)$, which is referred to as **expectation**. There is a unique tuple $(\pi, o, \nu_o, \nu_a, ev) \in \epsilon$ that corresponds to any given π.*

Expectations sets are expressed in a manner that allows comparisons between any two sets of expectations. Expectation sets are examined to determine relationships between their members.

Expectation Properties

As it was mentioned earlier, the expectation semantics dictate that an expectation is valid if and only if the relationship between the observed value ν_o and the actual value ν_a under operation o and covering method $covering$ is also valid. Otherwise, a *violation* occurs. Table 6.2 illustrates valid expectations and violations.

Table 6.2: Valid Expectations and Violations

Operator o	ν_o	ν_a	Covering	Valid or Violation
$=$	3	5	strict	violation
$=$	5	3	d=2	valid
$\leqslant$	6	5	strict	violation
$\leqslant$	6	5	d=4	valid

The property of valid expectation is given in Property 6.1.17.

Property 6.1.17 Valid Expectation *Expectation (π, o, ν_o, ν_a, ev) is valid if and only if ν_o o ($\nu_a + d$) for $o \in \{=, <, <=\}$ and ν_o o ($\nu_a - d$) for $o \in \{>, >=\}$ is true under the ordering of the values of set V. Value d is set to 0 for strict covering and set to the permissible deviation for relaxed covering technique.*

Note that according to the semantics of threshold, a lower bound set by the operator $>$ is shifted down by d whereas the upper bound set by operator $<$ is shifted up by d.

Expectation Set Comparisons

By itself an expectation set is not interesting unless operations are performed on its elements. However, prior to defining these operations we must first define the primitive comparison relationships between its elements. The relationships between expectation tuples determine the relationships between expectation sets. These binary relationships include the standard *equality* ($=$) and *less than or equal* ($\leqslant$) relationships as well as the redefinition of *not equal* $\neq$. In addition, a new relationship called *relaxed equal* is defined. Based on the binary relationships, expectation sets when compared fall in one of the four categories: strictly-equal, relaxed-equal, covered/covering and unrelated.

Starting with the relationships between expectation tuples, the equality relationship is given in Definition 6.1.18. Two expectation tuples are equal if their respective trust properties π_1, π_2, observed values ν_{o_1}, ν_{o_2}, allowed values ν_{a_1}, ν_{a_2}, and covering methods $covering_1, covering_2$ are the same. The triggering and aggregation methods do not need to be the same since they don't affect the semantics of the expectation tuples. Those attributes merely affect the when and how the observed value changes.

Definition 6.1.18 Equal Expectations *Expectation ($\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1$) is equal with expectation ($\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2$) if and only if $\pi_1 = \pi_2$ and $o_1 = o_2$ and $\nu_{o_1} = \nu_{o_2}$ and $\nu_{a_1} = \nu_{a_2}$ and $covering_1 \in ev_1 = covering_2 \in ev_2$.*

($\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1$) = ($\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2$) $\Leftrightarrow$

$\exists\, (\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)\,, (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon,$

$(covering_1, triggering_1, aggregation_1), (covering_2, triggering_2, aggregation_2) \in ev\bullet$

$(\pi_1 = \pi_2) \wedge (o_1 = o_2) \wedge (\nu_{o_1} = \nu_{o_2}) \wedge (\nu_{a_1} = \nu_{a_2}) \wedge (covering_1 = covering_2)$

One has to be careful with the semantics of not equal $\neq$. Using the definition 6.1.18, $(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \neq (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \Leftrightarrow \neg\, (\, \pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge \nu_{a_1} = \nu_{a_2} \wedge covering_1 = covering_2\,)$. However, this definition is incomplete when one takes into account the semantics of an expectation tuple. There are 2^7 (7 different expectation attributes) possible combinations that relate two expectation tuples under the $\neq$ operator. Due to brevity reasons, we only list a subset of these combinations in Table 6.3

Table 6.3: Expectation Combinations

Name	Notation	Expression
Equal	$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)$ = $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$	$\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge$ $\nu_{a_1} = \nu_{a_2} \wedge covering_1 = covering_2$
Relaxed Equal	$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)$ $\approx (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$	$(\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} \neq \nu_{o_2} \wedge$ $\nu_{a_1} \neq \nu_{a_2} \wedge covering_1 = covering_2)$ $\vee$ $(\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge$ $\nu_{a_1} \neq \nu_{a_2} \wedge covering_1 = covering_2)$
Not Equal	$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)$ $\neq (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$	$(\pi_1 \neq \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge$ $\nu_{a_1} = \nu_{a_2} \wedge covering_1 = covering_2\,)$ $\vee$ $(\pi_1 \neq \pi_2 \wedge o_1 \neq o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge$ $\nu_{a_1} = \nu_{a_2} \wedge covering_1 = covering_2\,)$ $\vee \cdots$

The *not equal* is redefined to cover only the combinations such that $\pi_1 \neq \pi_2$. The reason for excluding combinations where $\pi_1 = \pi_2 \wedge \nu_{a_1} \neq \nu_{a_2}$ lies on the semantics of an expectation; these combinations suggest that both expectation tuples refer to the same requirement but with different

values. As a result, the *relaxed equal* relationship relates two expectations that refer to the same property mapped to different values.

Definition 6.1.19 Relaxed Equal Expectations *An expectation* $(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)$ *is relaxed equal with another expectation* $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ *if and only if* $\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} \neq \nu_{o_2} \wedge \nu_{a_1} \neq \nu_{a_2} \wedge covering_1 = covering_2$ *or* $\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge \nu_{a_1} \neq \nu_{a_2} \wedge covering_1 = covering_2$

$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \approx (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \Leftrightarrow$

$\exists\, (\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)\,, (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon\,,$

$(covering_1, triggering_1, aggregation_1), (covering_2, triggering_2, aggregation_2) \in ev\bullet$

$(\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} \neq \nu_{o_2} \wedge \nu_{a_1} \neq \nu_{a_2} \wedge covering_1 = covering_2) \vee$

$(\pi_1 = \pi_2 \wedge o_1 = o_2 \wedge \nu_{o_1} = \nu_{o_2} \wedge \nu_{a_1} \neq \nu_{a_2} \wedge covering_1 = covering_2)$

Relaxed equal expectations are related by the $\leqslant$. The following definition combines equality and relaxed equality:

Definition 6.1.20 Less Than or Equal ($\leqslant$) Expectations *Expectation i =* $(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)$ *is less than or equal to expectation j=*$(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ *if and only if* $i \approx j$ *and* $\nu_{a_1} \leqslant \nu_{a_2}$.

$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \leqslant (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \Leftrightarrow$

$\exists\, i \in \epsilon_1,\, j \in \epsilon_2 \bullet i \approx j \wedge \nu_{a_1} \leqslant \nu_{a_2}$

Definitions 6.1.18 through 6.1.20 are used for determining relationships between any two expectation sets. Below, are 4 ways that expectation sets are related. Each relationship is illustrated with an example.

Starting with the equality relationship, two expectation sets are strictly equal if they contain the same elements. Since equality between members is already defined, the definition of improper

set needs not to be redefined. Expectation sets $\epsilon_1 = \{(cooperation, =, 1, 1, ev), (reliability, >, 0.98, 0.97, ev2)\}$ and $\epsilon_2 = \{(cooperation, =, 1, 1, ev), (reliability, >, 0.98, 0.97, ev2)\}$ are strictly equal.

Definition 6.1.21 Strictly Equal Expectation Sets *Expectation set ϵ_1 is strictly equal to expectation set ϵ_2 if and only if ϵ_1 is an improper*[1] *set of ϵ_2, under the equality definition 6.1.18.*

$\epsilon_1 = \epsilon_2 \Leftrightarrow \forall\, i \in \epsilon_1\ \exists\, j \in \epsilon_2 \bullet i = j \wedge |\epsilon_1| = |\epsilon_2|$

Consider expectation sets $\epsilon_1 = \{(cooperation, >, 3, 1, ev), (reliability, >, 0.98, 0.97, ev2)\}$ and $\epsilon_2 = \{(cooperation, >, 2, 1, ev), (reliability, >, 0.98, 0.97, ev2)\}$ These two sets don't have the same elements, but they both contain values for the same properties: cooperation and reliability. These two sets are called relaxed equal. The relaxed equal comparison is a generalization of the strictly equal comparison.

Definition 6.1.22 Relaxed Equal Expectation Sets *Expectation set ϵ_1 is relaxed-equal to expectation set ϵ_2 if and only if for all tuples i=$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \in \epsilon_1$ there is tuple j = $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon_2$ such as $|\epsilon_1| = |\epsilon_2|$ and ($i \approx j$ or $i = j$).*

$\epsilon_1 \approx \epsilon_2 \Leftrightarrow \forall\, i \in \epsilon_1\ \exists\, j \in \epsilon_2 \bullet ((i = j) \vee (i \approx j)) \wedge |\epsilon_1| = |\epsilon_2|$

An interesting relationship between two expectation sets is the covering relationship. An expectation set is covered by another one if all its elements are related with an element from another expectation set under the relations $=$ or $\leqslant$.

Consider expectation sets $\epsilon_1 = \{(cooperation, >, 3, 1, ev), (reliability, >, 0.98, 0.97, ev2)\}$ and $\epsilon_2 = \{(cooperation, >, 4, 2, ev), (reliability, >, 0.98, 0.97, ev2),$ $(authentication, =, 1, 1, ev)\}$. ϵ_1 is covered by ϵ_2 and this relationship is denoted as $\epsilon_1 \leqslant \epsilon_2$.

[1]The symbol $\subseteq$ admits the possibility that a subset concerned maybe in the fact the whole of the set. The symbol $\subset$ rejects the idea. The symbol for improper set is therefore not defined.

Definition 6.1.23 Covered/Covering Expectation Sets *Expectation set ϵ_1 is covered by expectation set ϵ_2 if and only if for all tuples i=($\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1$) $\in \epsilon_1$, there is tuple j = ($\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2$) $\in \epsilon_2$ such i $\leqslant$ j. Analogously, expectation set ϵ_2 is covering expectation set ϵ_1.*

$\epsilon_1 \leqslant \epsilon_2 \Leftrightarrow \forall\, i \in \epsilon_1\, \exists\, j \in \epsilon_2 \bullet (i \leqslant j)$

Finally, there is the case where the elements of an expectation set are not related to the elements of another set with only one relationship or the elements are simply not equal. Expectation sets $\epsilon_1 = \{(cooperation, >, 2, 1, ev)\}$ and $\epsilon_2 = \{(reliability, >, 0.98, 0.97, ev2), (authentication, =, 1, 1, ev)\}$ are unrelated.

Definition 6.1.24 Unrelated Expectation Sets *Expectation set ϵ_1 is unrelated with expectation set ϵ_2 if and only if the two sets are not strictly equal or relaxed equal or associated by the covered/covering relation.*

$\epsilon_1 \neq \epsilon_2 \Leftrightarrow \neg\, (\epsilon_1 = \epsilon_2 \vee \epsilon_1 \approx \epsilon_2 \vee \epsilon_1 \leqslant \epsilon_2 \vee \epsilon_2 \leqslant \epsilon_1)$

Recall, that in Section 3.3.4 there was a question that involved expectation set satisfaction. The question was phrased as "Given a target set ϵ and a set of expectation sets $E = \{\epsilon_1, \epsilon_2, \cdots, \epsilon_n\}$, does the target expectation set satisfy all members of set E?" The solution to this question is given below:

$\epsilon\ satisfies\ E \Leftrightarrow \forall \epsilon_i \in E \bullet (\epsilon = \epsilon_i \vee \epsilon_i \leqslant \epsilon)$

Expectation Set Operations

A new set of expectations can be defined from existing expectation sets. There are two operations that result in the creation of a new set: merging and semantic distance.

Merging is an operation on two expectation sets and its purpose is to provide a unified view of expectations, which answers queries like "What is the expectation set for a path that starts from X

and terminates at Y?" A general way to accomplish merging would be to apply a function f_π on the values of a property.

Property 6.1.25 Merging of Expectation Sets (General Function) *Consider expectation sets ϵ_1 and ϵ_2. The merging of the two expectation sets results in a new expectation set ϵ_{merge} that is constructed as follows:*

1. *Initialize $\epsilon_{merge} = \emptyset$*
2. *If $\epsilon_1 = \epsilon_2$, then $\epsilon_{merge} \cup \epsilon_1$.*
3. *if $\epsilon_1 \approx \epsilon_2$, then $\forall i$:$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \in \epsilon_1$, $\exists j$:$(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon_2$ such that $i \approx j$ and $\epsilon_{merge} \cup \{((\pi_1, o_1, \nu_{f\pi_{(o_2,o_1)}}, \nu_{f\pi_{(a_2,a_1)}}, ev_1))\}$.*
4. *if $\epsilon_1 \leqslant \epsilon_2$, then $\forall i$:$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \in \epsilon_1$, $\exists j$:$(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon_2$ such that $i \leqslant j$ and $\epsilon_{merge} \cup \{((\pi_1, o_1, \nu_{f\pi_{(o_2,o_1)}}, \nu_{f\pi_{(a_2,a_1)}}, ev_1))\}$. . Furthermore, for all tuples $(\pi_d, o_d, \nu_d, \nu_d, ev_d)$ in $\epsilon_2 - \epsilon_1$, $\epsilon_{merge} \cup \{((\pi_d, o_d, \nu_{f\pi_{(o_d)}}, \nu_{f\pi_{(a_d)}}, ev_d))\}$.*
5. *if $\epsilon_2 \leqslant \epsilon_1$, then $\forall i$:$(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon_2$, $\exists j$:$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \in \epsilon_1$ such that $i \leqslant j$ and $\epsilon_{merge} \cup \{((\pi_2, o_2, \nu_{f\pi_{(o_2,o_1)}}, \nu_{f\pi_{(a_2,a_1)}}, ev_2))\}$. . Furthermore, for all tuples $(\pi_d, o_d, \nu_d, \nu_d, ev_d)$ in $\epsilon_1 - \epsilon_2$, $\epsilon_{merge} \cup \{((\pi_d, o_d, \nu_{f\pi_{(o_d)}}, \nu_{f\pi_{(a_d)}}, ev_d))\}$.*
6. *if $\epsilon_2 \neq \epsilon_1$, then $\epsilon_{merge} \cup \epsilon_1$. For all $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ in ϵ_2, if there is tuple $(\pi_m, o_m, \nu_{o_m}, \nu_{a_m}, ev_m) \in \epsilon_{merge} \leqslant (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \vee (\pi_m, o_m, \nu_{o_m}, \nu_{a_m}, ev_m) \approx (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \vee (\pi_m, o_m, \nu_{o_m}, \nu_{a_m}, ev_m) = (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ then $\epsilon_{merge} \cup \{((\pi_m, o_m, \nu_{f\pi_{(o_2,o_m)}}, \nu_{f\pi_{(a_2,a_m)}}, ev_m))\}$ and $\epsilon_2 - (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$. Afterwards, if $|\epsilon_2| \neq 0$, for all tuples $(\pi_d, o_d, \nu_d, \nu_d, ev_d)$ in ϵ_2, $\epsilon_{merge} \cup \{((\pi_d, o_d, \nu_{f\pi_{(o_d)}}, \nu_{f\pi_{(a_d)}}, ev_d))\}$.*

Merging expectations based on the weakest link theory in security chains is another way to unify expectations. A security chain is as strong as its weakest link. Consequently, the aggregation

of expectation sets reflects the weakest expectations from both sets, meaning that the lower values are chosen. This is a pessimistic approach for combining expectations, but nevertheless viewing a group of entities with diverse expectations as a single entity assumes that a low value dominates a high value. Viewing a chain as a single link means to find the achilles heel, the most vulnerable point, its most damaging weakness. A future work on this is to detect the weakest link and devise ways to toughen that link in the chain. The new set is called the merging of expectation sets ϵ_1 and ϵ_2.

Property 6.1.26 Merging of Expectation Sets (Weakest expectation) *Consider expectation sets ϵ_1 and ϵ_2. The merging of the two expectation sets results in a new expectation set ϵ_{merge} that is constructed as follows:*

1. *Initialize $\epsilon_{merge} = \emptyset$*
2. *If $\epsilon_1 = \epsilon_2$, then $\epsilon_{merge} \cup \epsilon_1$.*
3. *if $\epsilon_1 \approx \epsilon_2$, then $\forall(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \in \epsilon_1$, $\epsilon_{merge} \cup \{(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1)\}$ if and only if $(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \leqslant (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon_2$. Otherwise, $\epsilon_{merge} \cup \{ (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \}$.*
4. *if $\epsilon_1 \leqslant \epsilon_2$, then $\epsilon_{merge} \cup \epsilon_1$. Furthermore, for all tuples $(\pi_d, o_d, \nu_{o_d}, \nu_{a_d}, ev_d)$ in $\epsilon_2 - \epsilon_1$, replace ν_{a_d} and ν_{o_d} by the minimum value ν_{min} from the value set V, and $\epsilon_{merge} \cup \{ (\pi_d, o_d, \nu_{o_{min}}, \nu_{a_{min}}, ev_d) \}$.*
5. *if $\epsilon_2 \leqslant \epsilon_1$, then $\epsilon_{merge} \cup \epsilon_2$. Furthermore, for all tuples $(\pi_d, o_d, \nu_{o_d}, \nu_{a_d}, ev_d)$ in $\epsilon_1 - \epsilon_2$, replace ν_{a_d} and ν_{o_d} by the minimum value ν_{min} from the value set V, and $\epsilon_{merge} \cup \{ (\pi_d, o_d, \nu_{o_{min}}, \nu_{a_{min}}, ev_d) \}$.*
6. *if $\epsilon_2 \neq \epsilon_1$, then $\epsilon_{merge} \cup \epsilon_1$. For all $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ in ϵ_2, if there is tuple*

$(\pi_m, o_m, \nu_{o_m}, \nu_{a_m}, ev_m) \leqslant (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$, then $\epsilon_2 - (\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$. Afterwards, if $|\epsilon_2| \neq 0$, for all set members replace ν_{a_2} and ν_{o_2} in $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ by the minimum value ν_{min} from the value set V and $\epsilon_{merge} \cup \{(\pi_2, o_2, \nu_{o_{min}}, \nu_{a_{min}}, ev_2)\}$

The second operation is to determine the *distance* between two expectation sets. Given an ideal expectation set ϵ_{ideal}, expectation set $\epsilon_{current}$ meets, under-qualifies or over-qualifies the ideal set's requirements. Semantic distance is an indicator of the deviation between a given expectation set and an ideal one, when operands for all expectation tuples is the equality operand. This is an alternative method of deciding whether a group of entities undertaking the same task meets particular expectations. In order to estimate the distance between any two sets, we use a variant of the k-nearest neighbor algorithm, which classifies new data points based on a similarity measure. That measure, in our case, is the Euclidean distance. An expectation set with n attributes is mapped into an n-dimensional cartesian system as a single point. In this n-dimensional space, the distance between two points (two expectation sets) is calculated as follows:

$$d(\epsilon_1, \epsilon_2) = \sqrt{\sum_{k=1}^{n} (\nu_{1k} - \nu_{2k})^2} \tag{6.1}$$

In the above equation all attributes carry the same importance. However, there are cases where an attribute affects trust more than the remaining attributes; for example, in ecommerce applications authentication of the server is far more important than the competition factor. Attribute weights are used to affect the distance metric. In particular, the Euclidean metric is modified by a set of weights that represent the information content or importance of each requirement as follows:

$$d_w(\epsilon_1, \epsilon_2) = \sqrt{\sum_{k=1}^{n} w(k)(\nu_{1k} - \nu_{2k})^2} \tag{6.2}$$

Selecting a subset of attributes is a special case of feature weighting where the weights can only take binary values.

Property 6.1.27 Semantic Distance Between Expectation Sets *Consider expectation sets ϵ_{ideal} and $\epsilon_{current}$. The semantic distance between the two sets is estimated as follows:*

1. *Initialize variable sum to 0*
2. *$\forall i$:$(\pi_1, o_1, \nu_{o_1}, \nu_{a_1}, ev_1) \in \epsilon_{ideal}$ do one of the following in the order given:*
   (a) *if $\exists j$:$(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2) \in \epsilon_{current}$ such that $i = j \vee i \approx j$ then* add*(sum, $(\nu_{a_1} - \nu_{a_2})^2$).*
   (b) *if the above is not true, then* add*(sum, $(\nu_{a_1})^2$)*
3. *if $|\epsilon_{ideal}| < |\epsilon_{current}|$, then* add*(sum, $(\nu_{a_2})^2$) for all $(\pi_2, o_2, \nu_{o_2}, \nu_{a_2}, ev_2)$ not in ϵ_{ideal}*
4. $d(\epsilon_1, \epsilon_2) = sqrt(sum)$

The weighted semantic distance between sets is similar to operation 6.1.27, with the difference that weight w is attached to the difference between two values for the same property.

6.1.6 Interaction id

Another trust relation attribute is the interaction identifier id. There is a unique identifier for each indirect interaction, and there are at least two trust relationships for such interaction.

6.1.7 Status

The last attribute is the status of a trust relationship. Status takes values from the set $S = \{OK, WARNING, ALERT, TERMINATED - OK, TERMINATED - NOT - OK\}$. The first three elements of S are used when the relationship is valid in τ and the remaining two are used when the relationship does not belong to τ. In the latter case, the relationship is added as experience along with the appropriate status element that represents the reason of termination.

6.2 Formal Definition of Trust Relation

Trustors, trustees, contexts, levels, intervals, expectations, interaction identifiers and status are combined together to represent trust as a relation. Let Γ, Δ, C, Λ, I, E, ID, and S be the sets

of trustors, trustees, contexts, intervals, levels, expectations, identifiers and status respectively. A trust relation between the eight sets is a collection of ordered tuples $(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ such that $\gamma \in \Gamma$, $\delta \in \Delta$, $c \in C$, $\lambda \in \Lambda$, $\iota \in I$, $\epsilon \in E$, $id \in ID$ and $s \in S$. Let's call this 8-ary trust relation τ. The formal definition of trust as a relation has as follows:

Definition 6.2.1 Trust *Trust is represented as a trust relation τ and the $\tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ tuple valid in τ represents a trust relationship between a trustor γ and a trustee δ. Trustor γ, based on the current imputed trust mode λ_{im} (with $\lambda = (\lambda_{im}, \lambda_t)$, believes that the extent to which trustee δ will act as expected for context c during time interval ι is λ_t, and this belief is subject to the satisfaction of expectation set ϵ. This relationship is valid for a specific lifecycle stage and interaction id. Status s represents the current status of the relationship.*

6.3 Representation of Trust Relation

A trust relation, being a set of ordered tuples, is represented by listing its members using set notation or using the more informal form of a table. In the first case, Ω is a subset of $\Gamma \times \Delta \times C \times \Lambda \times I \times E \times ID \times S$ that consists of those ordered tuples $(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ for which $\tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ is true. Formally, $\Omega = \{ (\gamma, \delta, c, \lambda, \iota, \epsilon, id, s) ; \gamma \in \Gamma, \delta \in \Delta, c \in C, \lambda \in \Lambda, \iota \in I, \epsilon \in E, id \in ID, s \in S; \tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s) \}$.

The Ω set that corresponds to Figure 6.1 has as follows: $\Omega = \{ (\gamma_2, \delta_2, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$, $(\gamma_1, \delta_1, c_2, \lambda_1, \iota_2, \epsilon_1, id_1, s_1)$, $(\gamma_1, \delta_2, c_1, \lambda_1, \iota_6, \epsilon_1, id_1, s_1)$, $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_3, \epsilon_2, id_1, s_1)$, $(\gamma_1, \delta_1, c_2, \lambda_4, \iota_1, \epsilon_1, id_1, s_1)$, $(\gamma_2, \delta_1, c_1, \lambda_1, \iota_5, \epsilon_1, id_1, s_1)$ $\}$.

The second method for representing τ is the tabular representation shown in Table 6.4. This scheme, even though is intuitively satisfying, it consumes a lot of space.

6.4 Trust Relation Visualizations

There are a number of ways to visualize relations graphically and those include using the cartesian coordinates, mapping diagrams and directed graphs. Starting with the cartesian coordinate system,

Table 6.4: Tabular Representation of Trust Relation of Figure 6.1

Trustor	Trustee	Context	Level	Interval	Expectation	Interaction Id	Status
γ_2	δ_2	c_1	λ_1	$\iota_1 = [t_1, t_3]$	ϵ_1	id_1	s_1
γ_1	δ_1	c_2	λ_1	$\iota_2 = [t_0, t_2]$	ϵ_1	id_1	s_1
γ_1	δ_2	c_1	λ_1	$\iota_6 = [t_3, t_5]$	ϵ_1	id_1	s_1
γ_1	δ_1	c_1	λ_1	$\iota_3 = [t_0, t_1]$	ϵ_2	id_1	s_1
γ_1	δ_2	c_1	λ_2	$\iota_4 = [t_2, t_4]$	ϵ_1	id_1	s_1
γ_2	δ_1	c_1	λ_1	$\iota_5 = [t_0, t_3]$	ϵ_1	id_1	s_1

the visualization of higher dimensions is not trivial. Since the focus of this paper is not to devise an 8-dimensional cartesian coordinate system, we are using the 3-dimensional system to capture a snapshot of the relation at a specific time, trustor, level and status. Figure 6.6 illustrates such a snapshot for the trust relation of Figure 6.1. The utility of such a representation is the ability to decompose the universal relation τ into groups with narrower focus. Such decomposition is based on grouping tuples specific for a trustor, time, level and status. We define such a group as *a given trustor's trust relation plane for level λ, status s at time t.* A trustor has multiple trust relation planes that change over time (Sliding trust planes).

Definition 6.4.1 Trust Relation Plane *Trust Relation Plane is a collection of tuples ($\gamma, \delta, c, \lambda, \iota$) belonging to τ for a specific γ, λ and s at a specific time point t. The trust relation plane dynamically changes to reflect the intervals that those tuples are valid in τ.*

The second method for relation visualization is a graph. Graphs focus on binary relations where hypergraphs focus on n-ary relations.. A hypergraph is a graph structure where an edge links more than 2 vertices. The complexity of hypergraphs led us in tailoring graphs to accommodate the 8-ary trust relation. A node represents either a trustor or a trustee. Nodes are connected by directed edges that start at a trustor node and end at a trustee node. Each edge carries the tuple that describes the relationships between the two connecting nodes. An edge may carry multiple tuples. Figure 6.7 is a graph for trust relation τ of Figure 6.1.

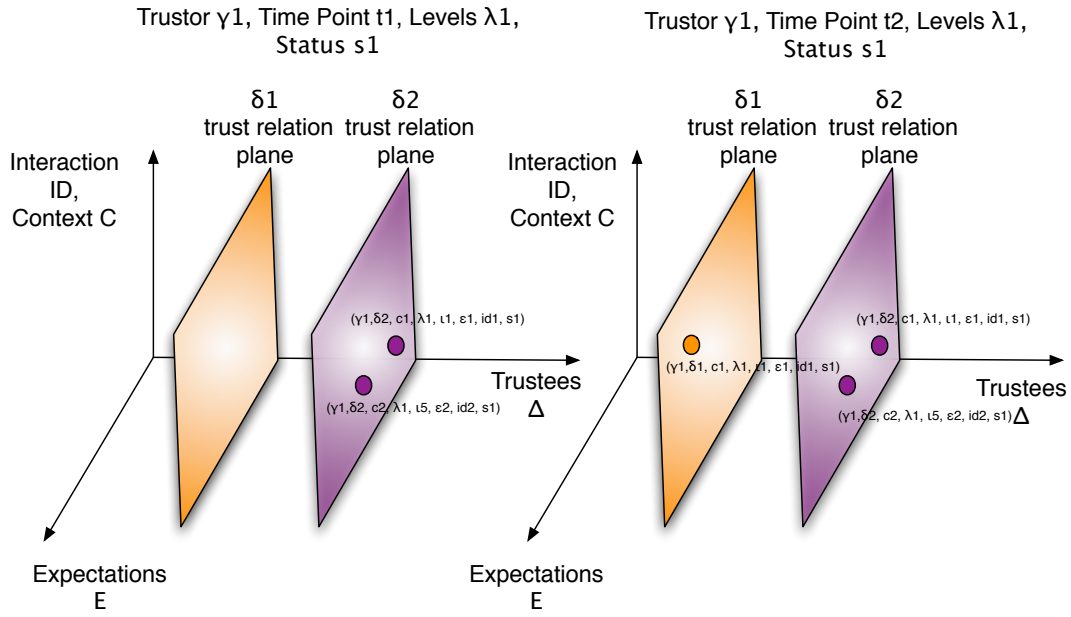

Figure 6.6: Trust Relation Visualization in Cartesian Coordinate System at selected time points

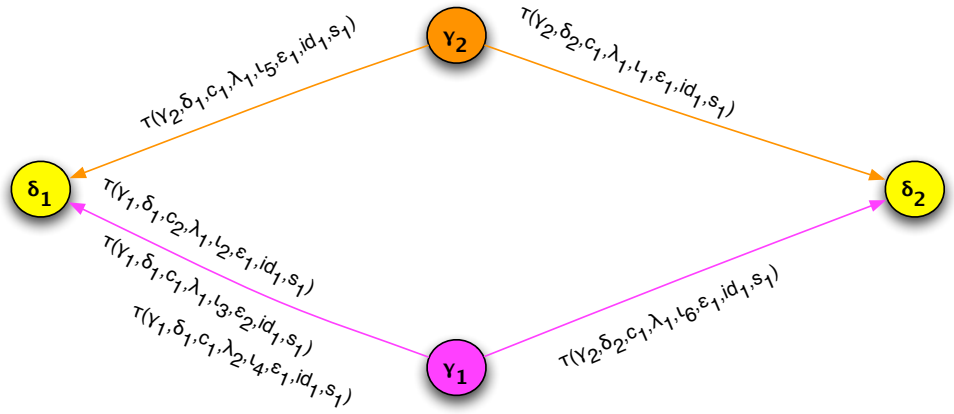

Figure 6.7: Trust Relation Graph

6.5 Trust Relation Properties

After formally defining trust, its properties discussion naturally comes next. For all definitions, operations, and observations that follow, the trust level set Λ is the set {A (agnostic), PT (partial trust), PD (partial distrust), FT (full trust), FD (full distrust), N (neutral)} whose elements are ordered as the lattice of Figure 6.4(c). Throughout this discussion, references to Figures 6.8,6.9 as an example trust relation graph corresponding to a specific network topology will be made. This will help clarify and demonstrate concepts where needed.

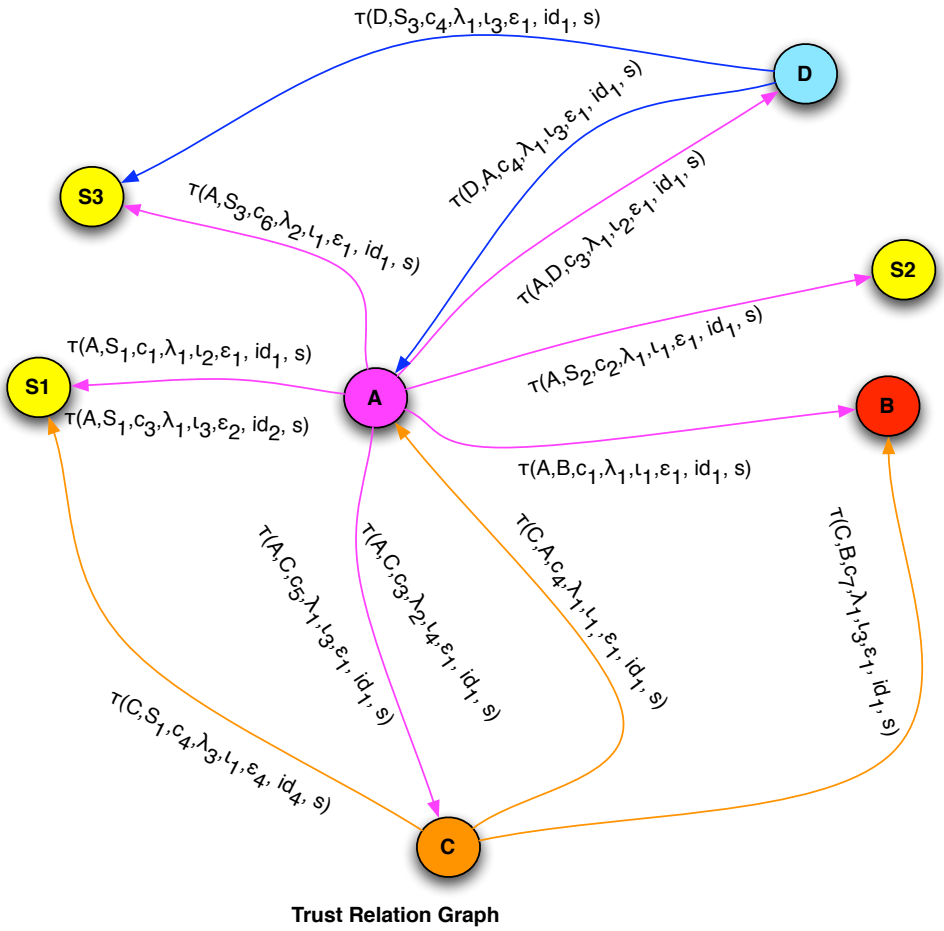

Figure 6.8: Trust Graph for Example Network Topology

The next step in formalizing trust is the definition of its properties. The standard properties

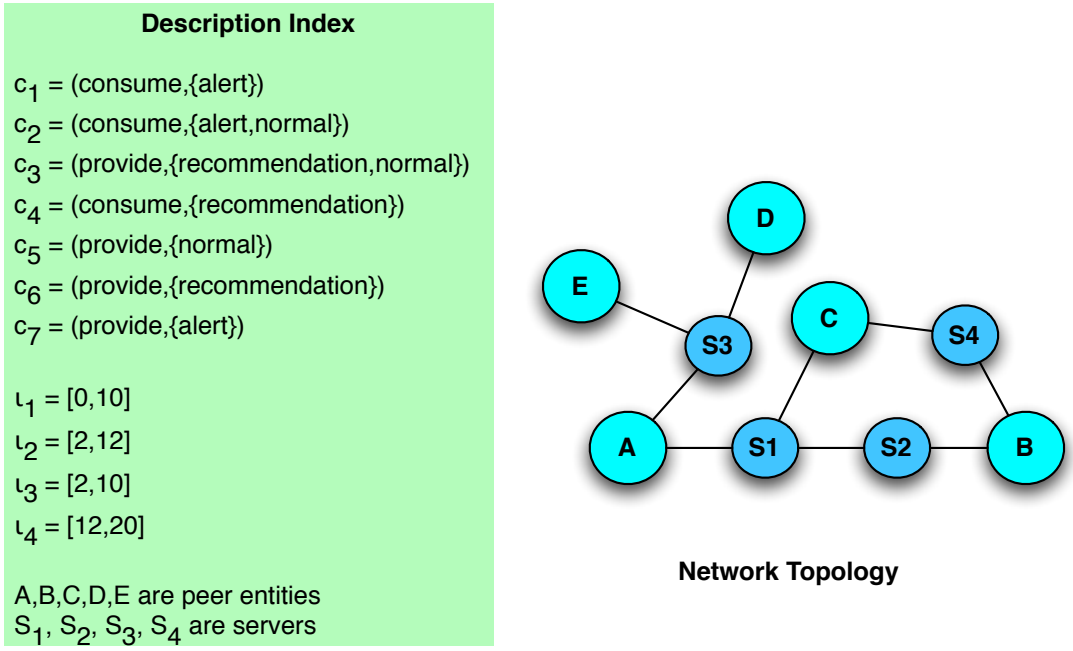

Figure 6.9: Network Topology for Figure 6.8

of any n-ary relation include the reflexive, irreflexive, symmetric, antisymmetric, transitive, and equivalence properties. In the literature, there are claims stating that trust is not transitive and not symmetric. A careful examination of the implications of these statements disproves those claims. First, let us define what a property is. If condition X is true about a relation R, then we say that relation R has the X property. The property is either valid or invalid for a specific relation. Having that said, our position is that the standard properties for trust relations are difficult to prove due to the non-absolute characteristics of trust relationships and their time dependance. These are limitations of expressing trust using the formalisms of sets and relations.

In order to support the above claims, we examine whether or not the antisymmetric property holds for trust relations. An ordered tuple (a,b,c) is also represented as an ordered pair ((a,b),c). Our claim is that the antisymmetric property is tested on the first coordinate of the ordered pair. Our assumption is based on the semantics of the property within the trust relation. In this case, antisymmetric property states that ordered tuple ((b,a),c) does not exist. Antisymmetry is true if

and only if for every tuple $(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ the tuple$(\delta, \gamma, c, \lambda, \iota, \epsilon, id, s)$ does not exist in τ. This condition is time-independent. Consider a simple two-node peer-to-peer network that facilitates the exchange of data generated by temperature sensors. Suppose γ_1 has complete trust in using the received data from node δ_1 for a period of 30 days. The trust relation $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ is valid for $[t, t+30]$. At this point of time $t = today$, the antisymmetric property holds. If node δ_1 decides to establish a reciprocal relationship with γ_1, then τ will seize to be antisymmetric because the tuple $(\delta_1, \gamma_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ belongs to the relation.

All standard properties face the same issue described above and as a result we have to seek other properties unique to trust relations. Trust relation properties are time-dependent and usually valid on restrictions of trust relations.

At a specific time point and particular imputed trust model, there is only one trustworthiness level associated with a context for a specific interaction. For example trustor A has set two trust relationships $\tau(\gamma_A, \delta_C, c_3, \lambda_2, \iota_3, \epsilon_1, id_1, s_1)$ and $\tau(\gamma_A, \delta_C, c_5, \lambda_2, \iota_4, \epsilon_1, id_1, s_1)$ with trustee C for contexts c_3 and c_5 that fall into the Dominant/Dominating category. In this case, the second relationship is valid only after the first one becomes invalid. On the other hand, trustor A allows tuples $\tau(\gamma_A, \delta_{S1}, c_1, \lambda_2, \iota_2, \epsilon_1, id_1, s_1)$ and $\tau(\gamma_A, \delta_{S1}, c_3, \lambda_1, \iota_3, \epsilon_2, id_2, s_1)$ to be valid for an intersected period of time because the two contexts and their associated interactions are different. The requirement of non-overlapping trust tuples for dominant/dominated or intersecting contexts is set so that the formalism will be deterministic in assigning a trustworthiness level for a given context at any time point. Allowing conflicting trust relationships would have required devising ways to resolve the conflicts.

Property 6.5.1 Non-overlapping for Dominant/Dominated and Intersecting Contexts *If there exist ordered tuples $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ and $(\gamma_1, \delta_1, c_2, \lambda_2, \iota_2, \epsilon_2, id_1, s_1)$ that satisfy τ such that c_1 and c_2 are either Dominant/Dominated or Intersecting contexts, then $\iota_1 < \iota_2$ (or its inverse) or ι_1 m ι_2 (or its inverse) must hold.*

Property 6.5.2 Overlapping for Different and Unrelated contexts *If there exist ordered tuples* $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ *and* $(\gamma_1, \delta_1, c_2, \lambda_2, \iota_2, \epsilon_2, id_1, s_1)$ *that satisfy* τ *such that* c_1 *and* c_2 *are either Different or Unrelated contexts, then any valid relationship may hold between their respective intervals* ι_1 *and* ι_2.

The model's supported trust level classification entails that absence of trust does not equal with the presence of distrust. In the contrary, when no information is available to evaluate a trust relationship, *Agnostic* is automatically assigned as the trust level of the tuple in question. The network topology of Figure 6.8 includes entity E whom A has no prior or current trust relationships with. The lack of experience, recommendations and any other factors that might affect the trust relation suggests that A may add tuples $(\gamma_A, \delta_E, c, \lambda_{agnostic}, \iota, \epsilon_{agnostic}, id, s)$ for all valid contexts c.. The valid interval of each tuple is application-specific.

Property 6.5.3 Absence of Trust Level *If at time* t *there is no matching tuple for trustor* γ*, trustee* δ *and context* c *in relation* τ *, then it is not implied that* $\tau(\gamma, \delta, c, \lambda_{Full-distrust}, \iota, \epsilon_{Full-distrust}, id, s)$ *is true.*

Property 6.5.4 Universal Non-agnostic (Closure property) *If at time* t *there is no matching tuple for trustor* γ*, trustee* δ *and context* c *in relation* τ *, then it is implied that* $\tau(\gamma_A, \delta_E, c, \lambda_{agnostic}, \iota, \epsilon_{agnostic}, id, s)$ *is true.*

Property 6.5.5 Trust Reduction *If at time* t *there are tuples* $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ *and* $(\gamma_1, \delta_1, c_2, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ *in* τ *with* c_1 *and* c_2 *be intersecting or unrelated contexts may be reduced to a single tuple* $(\gamma_1, \delta_1, c_3, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ *, where* c_3 *is* $c_1 \bigcup c_2$.

Property 6.5.6 Unique Imputed Trust Mode *At time* t*, there is only one active imputed trust mode* λ_{im} *for all tuples in* τ.

Property 6.5.7 Trustworthiness Levels and Imputed Trust Mode *For each imputed trust mode* λ_{im}*, there are* m *trustworthiness levels* λ_t*, where* m *depends on the level classification system.*

Property 6.5.8 Trustworthiness Levels and Expectations *For each imputed trust mode λ_{im}, there are m expectation sets that are associated with the m trustworthiness levels λ_t, where m depends on the level classification system.*

6.6 Trust Relation Operations

Relation R may be defined by describing the relation or listing ordered pairs that satisfy R. There are a number of ways to view a relation:

- Boolean function: A relation ρ on a set A can be viewed as a function of type $A \rightarrow A \rightarrow Boolean$.
- Set of ordered pairs: A relation ρ on a set A can be viewed as a the set $\{(a, b) \in A \times A|$ $\rho(a, b)$ is true$\}$.

The statement $\rho(a, b)$ *is true* can be replaced by either $\rho(a, b)$ or $a\rho b$. Once you choose a particular a, b pair, statement $\rho(a, b)$ is treated as a simple proposition that returns either true or false. In other words, the ordered pair (a,b) satisfies relation ρ. Relation ρ may be defined by describing (characteristic property of elements of the relation) the relation or by listing the ordered pairs that satisfy ρ. As an example, consider the definition of the transitivity property. A relation ρ on A is transitive if for all $a, b, c \in A$ when $a\rho b$ and $b\rho c$, then $a\rho c$ is implied. Another example is relation ρ defined on a set A by having $\rho = \{(1, 1), (2, 1)\}$. In this case the relation seems to have no obvious description other than $1\rho 1$ and $2\rho 1$ hold.

One of the characteristics of trust relation τ is its dynamic nature; that means a tuple $\tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ that is valid in time t_1 becomes invalid in t_2 with $t_2 > t_1$. There are several ways that members of τ get added or removed. Trust relation τ is affected by time, arrival of new evidence, violation of expectations either direct or indirect, initialization of new relationships and end-to-end assessment for an interaction or multiple interactions.

6.6.1 Expiration of Valid Time

A trust relationship $(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ does not hold in relation τ if its valid interval time expires. A terminated trust relationship is merged with the experience database, *Exp*, which has the same attributes as τ with an additional attribute ι_{actual}. This field stores the actual interval that the relationship was valid.

Operation 6.6.1 Expiration of Valid Time *A trust relationship $\tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ becomes false if and only if the current time t does not fall within the valid interval ι.*

$\forall\ (\gamma, \delta, c, \lambda, \iota, \epsilon, id, s) \in \tau,\ (t_s, t_f) \in \iota,\ t \in T,\ Exp \in EX \bullet (t_f \leqslant t) \Rightarrow \tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ = *false* $\wedge\ \{(\gamma, \delta, c, \lambda, \iota, (t_s, t), \epsilon, id, TERMINATED)\} \cup EX \wedge (\forall\ (\gamma, \delta_i, c, \lambda, \iota, \epsilon, id_i, OK) \in \tau$ $\bullet\ (\gamma, \delta_i, c, \lambda, \iota, \epsilon, id, OK)$ = *false* $\wedge\ (id = id_i) \wedge \{(\gamma, \delta_1, c, \lambda, \iota, \epsilon, id, WARNING)\} \cup \tau)$

6.6.2 Arrival of New Evidence

Suppose that new evidence arrives from a recommender regarding a particular trustee and context. Then, the new value will be applied to the appropriate trust property according to the trustee evaluation information for the trust property.

Operation 6.6.2 Arrival of New Evidence *Suppose that new evidence arrives at trustor γ from trustee r, which is a recommender for trustee δ regarding context c. The new evidence includes the trust property π_r and the recommended value ν_r. All i trust relationships $(\gamma, \delta, c, \lambda_i, \iota_i, \epsilon_i, id_i, s_i)$ are updated to reflect the application of the new evidence on ν.*

$\exists \pi_r \in \Pi, \nu_r \in V, c_r \in C, \delta_r \in \Delta\ \exists\ (\gamma, \delta, c, \lambda, \iota, \epsilon, id, s) \in \epsilon_1,\ \epsilon_1 \bullet \pi = \pi_r \wedge \delta = \delta_r \wedge c = c_r$ $\wedge$ *triggering* = *true* $\Rightarrow \epsilon$ - $\{\ (\pi, o, \nu_o, \nu_a, ev)\ \} \wedge \{\ (\pi, o, \nu_{aggregated}, \nu_a, ev)\ \} \cup \epsilon$

6.6.3 Violation of Expectations

Whenever new evidence arrives, the observed value changes according to the aggregation scheme for the specific property. An update in the observed value may lead into expectation violation. In this case, the respective trust relationship's status is set to ALERT. The relationship does not necessarily becomes false in τ; according to policies it might be the case that a trustor wants to monitor the relationship before terminating it. However, all other trust relationships that are associated with the alerted relationship's interaction identifier have their status set to WARNING.

Operation 6.6.3 Expectation Violation *In a case that an expectation* $(\pi, o, \nu_o, \nu_a, ev) \in \epsilon$ *is not valid for* $\tau(\gamma, \delta, c, \lambda, \iota, \epsilon, id, s)$ *, the respective relationship's status* s *becomes ALERT and all tuples associated with the same interaction identifier* id *have their status set to WARNING.*

$\exists\ (\pi, o, \nu_o, \nu_a, ev) \in \epsilon \bullet (\nu_o\ o\ (\nu_a + d)) = false \Rightarrow \tau\ (\gamma, \delta, c, \lambda, \iota, \epsilon, id, s) = false \wedge \{(\gamma, \delta, c, \lambda, \iota, \epsilon, id, ALERT)\} \cup \tau \wedge (\forall\ (\gamma, \delta_i, c, \lambda, \iota, \epsilon, id_i, OK) \bullet (\gamma, \delta_i, c, \lambda, \iota, \epsilon, id, OK) = false \wedge (id = id_i) \wedge \{(\gamma, \delta_1, c, \lambda, \iota, \epsilon, id, WARNING)\} \cup \tau)$

However, a relationship's status is restored whenever the violation gets corrected during the monitoring interval. In this case, the ALERT gets replaced by OK, and all WARNING are replaced with OK. Note that if there are multiple violations for a specific interaction, then WARNINGS remain as they are until all ALERTS become OK.

6.6.4 Initialization of Trust Relationship

Prior establishing any trust relationships, there is the possibility of passive monitoring. What that means, is that for a designated time period, a trustee's behavior and properties are monitored without modifying the trust relation τ. After the passive monitoring terminates, a trust relationship for a trustee δ is valid in τ as follows:

- Apply Property 6.5.4 : in this case the tuples $(\gamma, \delta, c_i, \lambda_{agnostic}, \iota_{agnostic}, \epsilon_{agnostic}, id_{null}, s_s)$ are valid in τ, where c_i is the context that trustee was monitored for.

- Examine Experience Database: If trustee δ was previously involved with the trustor for the same context and the status is TERMINATED-OK, then add the same trust relationship in τ with valid interval $\iota = \iota_{actual}$, where ι_{actual} is the interval that the relationship was observed to be valid previously.

- Trustee Level Classification: Apply the operation 6.9.2 and the appropriate classification scheme to determine the trustworthiness level λ_t for context c . If found, then add $(\gamma, \delta, c, \lambda_{t,im}, \iota_t, \epsilon_t, id_{null}, s)$

6.6.5 End-to-End Trust Assessment and Aggregation of Data

End-to-end assessment is possible for a specific interaction id, and trust level λ. Our model does not have the capability to know exactly the duration of ι_g, ι_d, and ι_c and the consequence of this limitation is that trust relationships have to cover the entire information lifecycle and not the corresponding lifecycle that their context refers to.

Operation 6.6.4 End-to-End Trust Assessment for An Interaction *Consider tuples $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ and $(\gamma_1, \delta_2, c_2, \lambda_1, \iota_2, \epsilon_2, id_1, s_1)$ in τ where contexts c_1 and c_2 are characterized as Dominant/Dominated or Intersected. Trustor γ_1 may synthesize the two tuples to derive an end-to-end trust assessment for context c_i (the intersection of c_1 and c_2) during interval ι_i (the intersection of ι_1 and ι_2) by applying Operation 6.1.25 or 6.1.26 or 6.1.27 on the expectation sets ϵ_1 and ϵ_2 to derive the aggregated expectation set ϵ_i. Expectation set ϵ_i has to be checked, according to Operation 6.9.2, against the various level specifications in order to assign the trustworthiness level for the new tuple λ_i. The resultant tuple is $(\gamma_1, \delta_{1,2}, c_i, \lambda_i, \iota_i, \epsilon_i, id_1, s_1)$.*

Aggregation of data from multiple Interactions is also possible. First, end-to-end assessment is performed on each interaction and the resultant tuples are then synthesized to derive the aggregated result.

Operation 6.6.5 End-to-End Trust Assessment for Multiple Interactions *Consider tuples* $(\gamma_1, \delta_{1,n}, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ *and* $(\gamma_1, \delta_{1,m}, c_2, \lambda_1, \iota_2, \epsilon_2, id_2, s_1)$ *to be the resultant tuples after applying Operation 6.6.5 on the interactions* id_1, id_2 *respectively. Trustor* γ_1 *may synthesize the two resultant tuples to derive an aggregated end-to-end trust assessment for context* c_i *(the intersection of* c_1 *and* c_2*) during interval* ι_i *(the intersection of* ι_1 *and* ι_2*) by applying Operation 6.1.25 or 6.1.26 or 6.1.27 on the expectation sets* ϵ_1 *and* ϵ_2 *to derive the aggregated expectation set* ϵ_i*. Expectation set* ϵ_i *has to be checked, according to Operation 6.9.2, against the various level specifications in order to assign the trustworthiness level for the new tuple* λ_i*. The resultant tuple is* $(\gamma_1, \delta_{1,k}, c_i, \lambda_i, \iota_i, \epsilon_i, id_{1,2}, s_1)$ *, with* k *to be the total number of trustees involved in both interactions.*

6.7 Trust Relation Observations

There are several observations that can be made from the trust relationships semantics. These are outlined in this section.

Observation 6.7.1 *A trust relationship need not be symmetric for the same context but there may be two relationships between two entities for contexts that are related in some manner.*

Observation 6.7.2 *The non-associative nature of trust is observed when causality affects the formulation of trust relationships. Consider trust relationships* $\tau(\gamma_1, \delta_1, c_2, \lambda_1, \iota_2, \epsilon_1, id_1, s_1)$ *and* $\tau(\gamma_1, \delta_2, c_1, \lambda_1, \iota_6, \epsilon_1, id_1, s_1)$ *as illustrated in Figure 6.1. If the establishment of trust between* γ_1 *and* δ_2 *preceded in time the one between* γ_1 *and* δ_1*, then trustor* γ_1*'s trust in trustee* δ_1 *might be affected (negatively or positively). This scenario is possible if trustee* δ_2 *is a recommender whose opinions are of high importance to trustor* δ_1*.*

Observation 6.7.3 *A trust relationship is directed and may be one-to-one-to-one-..., one-to-one-to-...-to-many, etc. There are strong suggestions that trust relationships should not exhibit any*

transitivity but some researchers try to relax the transitivity property by allowing partial (or conditional) transitivity.

There are cases where symmetry is observed between two entities that hold dual roles as both a trustor and a trustee. In the example trust relation graph, trustor A places trust in trustee D for context c_3 with D acting on the placement of trust by forming a reciprocal trust relationship with A for the dual context of c_3. As a result, A and D trust each other for providing and consuming data of type *recommendation*. The concept of duality is very important because trust without acting on it has minimal usability in the overall trust network. Forming reciprocal relationships is the first step that leads into the final step of authorizing the exchange of information between two entities. Mutual duality is observed when the dual contexts have the same datatype set. In the previous example, there is no mutual duality between A and D.

Observation 6.7.4 Duality *Consider dual contexts* c_1, c_2. *Also, consider that* γ_1, δ_1 *refer to the same entity with the same assumption holding for* γ_2, δ_2 . *If there exist tuples* $(\gamma_1, \delta_2, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ *and* $(\gamma_2, \delta_1, c_2, \lambda_2, \iota_2, \epsilon_2, id_2, s_2)$ *in* τ *such that* $\iota_1 = \iota_2$ *or* ι_1 *o* ι_2 *(or its inverse) or* ι_1 *d* ι_2 *(or its inverse), or* ι_1 *s* ι_2 *(or its inverse), or* ι_1 *f* ι_2 *(or its inverse), then duality is observed between entities* γ_1 *and* γ_2 *regarding contexts* c_1 *and* c_2 *for the intersected interval* ι_s *between* ι_1 *and* ι_2.

A follow-up concept of relationship reciprocity is the non-associative nature of trust. Prior knowledge might change a trust relationship that otherwise could have been established with certain parameters of context, level and duration. Non-associativity is witnessed within a single trustor or between trustors. In the first case, recommendations for an entity and the time of arrival of such recommendations affect trust for a given trustee. At time $t = 1$, trustor A's experience includes $(\gamma_A, \delta_B, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ that indicates the establishment of trust with trustee B for receiving critical information such as alerts. Ar $t = 2$, A decides to place trust on D for receiving recommendations and as a result $(\gamma_A, \delta_D, c_3, \lambda_1, \iota_2, \epsilon_1, id_1, s_1)$ becomes valid in τ. One

of the received recommendations concerns trustee B and its involvement with leaking of confidential information to unauthorized parties. If $(\gamma_A, \delta_D, c_3, \lambda_1, \iota_2, \epsilon_1, id_1, s_1)$ preceded in time $(\gamma_A, \delta_B, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$, then A might have chosen not to allow access to any of its information by B.

Observation 6.7.5 Non-Associative Trust Scenario 1 *If there exist ordered tuples $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ and $(\gamma_1, \delta_2, c_2, \lambda_2, \iota_2, \epsilon_2, id_2, s_2)$ that satisfy τ such that $\iota_1 < \iota_2$ or ι_1 o ι_2 or ι_1 m ι_2 or ι_1 s ι_2, then the tuples might not exist for the inverse of the interval relationships. Trustor γ_1 might decide against establishing a trust relationship with δ_1 if it had prior knowledge of δ_2's trust recommendations.*

There is another form of non-associativity within a trust relation and it involves two different trustors and their relationships with the same trustee. At time $t = 1$, trustor A trusts B with consuming confidential alerts $(\gamma_A, \delta_B, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ but doesn't trust trustee C to gain access to those alerts. At a later time $t = 2$, C forms a trust relationship with B for receiving alerts $(\gamma_C, \delta_B, c_7, \lambda_1, \iota_3, \epsilon_1, id_2, s_1)$. Even though trust establishment does not necessarily mean that the specific action takes place, it is still an indication of intent. In case that B authorizes the propagation of alerts to C, then this becomes an indirect channel for C to receive indirectly alerts from A. If the trust bond between C and B occur prior to A's bond with B, then A might have decided on another course of action regarding B.

Observation 6.7.6 Non-Associative Trust Scenario 2 *If there exist ordered tuples $(\gamma_1, \delta_1, c_1, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ and $(\gamma_2, \delta_1, c_2, \lambda_2, \iota_2, \epsilon_2, id_1, s_1)$ that satisfy τ such that $\iota_1 < \iota_2$ or ι_1 o ι_2 or ι_1 m ι_2 or ι_1 s ι_2, then the tuples might not exist for the inverse of the interval relationships. Trustor γ_1 might decide against establishing a trust relationship with δ_1 if it had prior knowledge of the existence of the trust relationship between γ_2 and δ_1.*

Observation 6.7.7 Indirect Trust Changes *If the trust relationship with a recommender changes, then a number of relationships may change as well. However, the changes will be observed only when the relationships are re-evaluated according to their triggering rules.*

6.8 Trust Relation Theorems

There are two theorems for the trust relation τ. These theorems are related to the dynamic and composable features of trust, which are necessary for any indirect interaction.

Theorem 6.8.1 *A change in a relationship's current status affects all relationships for that particular interaction.*

There are only two cases when the status of a relationship changes: when valid time expires and when an expectation violation occurs. According to Operations 6.6.1 and 6.6.3 timing and expectation violations affect all relationships that are associated with the interaction identifier of the terminated of alerted relationship.

Theorem 6.8.2 *At least 2 trust relationships are needed for an interaction.*

Consider the case where there is only entity that produces and consumes the generated data. An interaction covers the information lifecycle from the producer to the consumer. In this case, the producer and consumer are the same entity. However, the contexts are different. Therefore, two relationships are needed as follows:

$(\gamma_1, \delta_1, c_{(produce,\{normal\})}, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$ and $(\gamma_1, \delta_1, c_{(consume,\{normal\})}, \lambda_1, \iota_1, \epsilon_1, id_1, s_1)$

For n entities involved in an interactions, then n relationships are needed, where $n < 1$.

6.9 Other Operations

There are operations that even though are not operating on the trust relation directly, they change the state of its attributes. These include the level satisfaction and classification for trustee.

6.9.1 *Trust Level Specification*

Our model uses discrete labeling, where each level is specified by a set of properties, their allowed values and a threshold value for each property. Since it is very unlikely that a trustee satisfies completely a level's specifications, a threshold is set to determine the flexibility allowed around the level's expectation set. It might be the case that there are multiple tuples associated with a property π (unlike an expectation set ϵ where a property only appears once in the set). Suppose there is a range for a specific property. In this case, the inequality operators $<, >$ generate two tuples for the property.

Operation 6.9.1 Level Specification *A trust level is specified as a pair* (c, p)*, where* $c \in C$ *and* p *is a set of tuples* $(\pi, o, \nu_a, \nu_{threshold})$ *, where* $\pi \in \Pi$*,* $o \in O$*, and* $\nu_a, \nu_{threshold} \in V$*.*

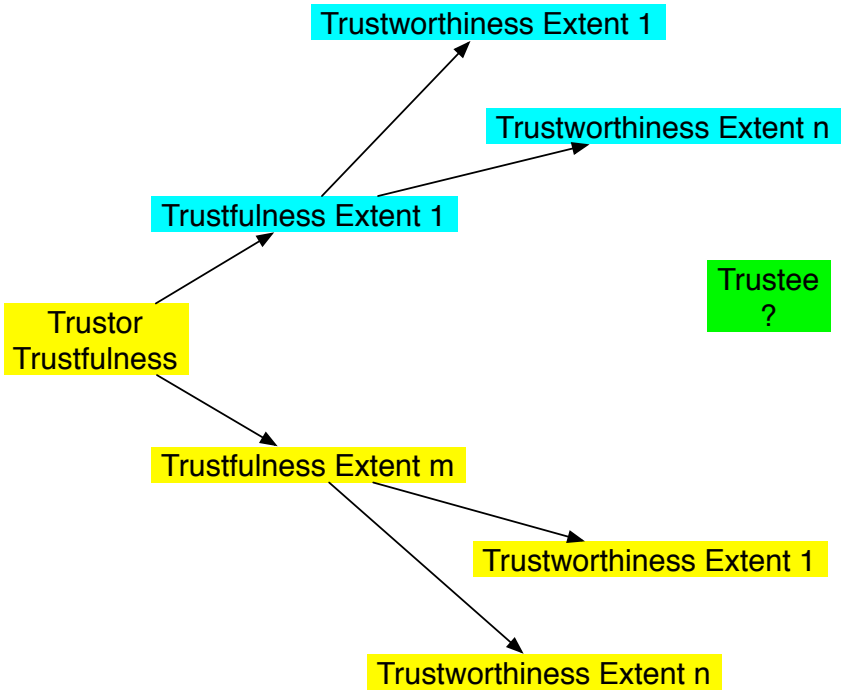

Figure 6.10: Trustor Trustfulness and Trustee Trustworthiness

This kind of level specification is applicable for defining trustworthiness extent for an operation on data. The imputed trust mode affects trustworthiness levels. For example, the threshold could

decrease when a trustor is in *red* imputed trust mode and thus narrowing the deviation from the desired expectations. As a result, distinct level specifications are assigned for trustworthiness extents that correspond to the same trustfulness extent (see Figure 6.10). In other formal models, there is no direct reference to trustfulness as a concept. However, our model explicitly represents trustfulness in order to link expectations to trustees in an adaptive manner. Whenever a trustor's trustfulness changes, its expectations change and the trustee's trustworthiness may change as well. Without loss of generality, it is assumed that a trustor exhibits the same trustfulness extent for all trustees at a given time.

6.9.2 Trust Level Satisfaction and Classification

In order to check whether or not a trustee satisfies a level's requirements, all properties for the level are tested against the trustee's observed values. As an example, consider the level specification for $\lambda_1 = \{(\pi_1, >, 5, 0), (\pi_2, =, 5, 2)\}$. Trustee δ_1, after X time of monitoring, exhibits the observed values for these properties $\{(\pi_1, 6), (\pi_2, 5)\}$. A simple testing of the observed values against the allowed ones concludes that trustee δ_1 could be assigned trustworthiness level λ_1.

Operation 6.9.2 Level Satisfaction *A trustee δ meets level's λ specification for context c if and only if for all properties π_i of λ, the trustee's observed value $\nu_{observed}$ satisfies the $\nu_{allowed}$ under operation o and threshold $\nu_{threshold}$.*

However, there are cases that level satisfaction does not necessarily lead into a unique level assignment for a trustee. There are cases where a trustee satisfies more than one trustworthiness levels for the same imputed trust mode. Continuing the example above, consider the case where there is one more level specification such as $\lambda_2 = \{(\pi_1, =, 6, 0), (\pi_2, =, 5, 0)\}$. The question now becomes how to choose which level specification matches better the trustee's observed behavior up to this point.

A solution to the above problem is to use classification techniques to determine the most appropriate level for the trustee. One technique is the use of the Euclidean distance to classify a trustee

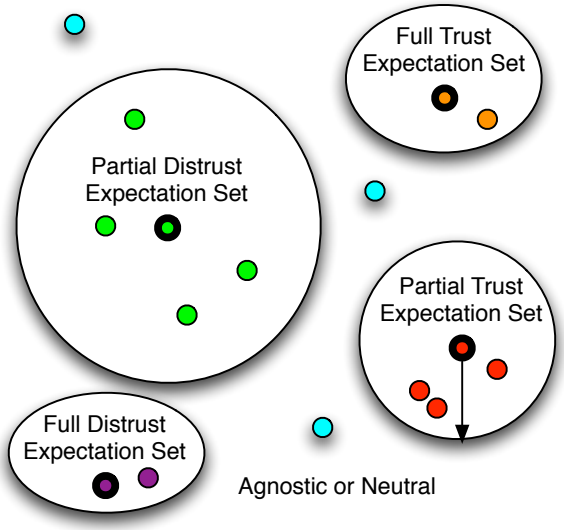

Figure 6.11: Trustee Level Classification in n-dimensional Space

to the appropriate level, as shown in Figure 6.11. This scheme can only be used if the operation for all properties is equality. Using this technique, trustee δ will be assigned trustworthiness level λ_2 since the semantic distance between the observed values and allowed values is 0.

In the case of inequality operators, the valid space for each level is bounded by hyperplanes. Operations theory could be used to devise membership tests i n this case. These tests are beyond the scope of this research work.

CHAPTER 7

RELATED WORK

There is no trust management research to date, that we are aware of, that has attempted to address the problem space outlined in Chapter 3 or define the issues, algorithms and mechanisms for trust management of indirect interactions. First, we give a brief overview of well-known trust management systems, and second, their capabilities and limitations are discussed in terms of those requirements.

7.1 State of the Art in Trust Management

The discussion of each system covers the following topics of interest:

- Description of the system: brief description of the system functionality and its main features
- Trust Scope: trust issue that the system addresses, trust specification, trust properties and requirements
- Collaborative environment capabilities: support for evidence aggregation and recommendation schemes
- Dynamic trust feature: trust re-evaluation triggers
- Composable trust constructs: provisions, mechanisms, concepts that could be used to support composable trust.

The trust management systems examined are organized into categories based on the trust issue that they address:

- Identity Trust: Public Key Certificates [45, 49]
- Resource Trust: PolicyMaker [12], KeyNote [11], REFEREE [15]

- Identity Trust and Resource Trust: Trust Establishment (TE) [30], TrustBuilder [47]
- Content Trust: PICS [39], Poblano [42]
- Behavior Trust: Trust-Aware Multicast (TAM) [34], TCPA [28]
- General Trust: SULTAN [27], Hestia.

The terms *trust*, *trusted*, *trustworthy* that are used to describe the systems below are consistent with the system's specific terminology. In most cases, a system's terminology is not consistent with ours. Appendix A outlines the trust terminology used in this research work.

7.1.1 Identity Trust: PUBLIC KEY CERTIFICATES

Description of the system

A public key certificate is a data structure that uses a digital signature to bind together a public key with an identity. Services based on public key certificates rely on a third party to sign a certificate certifying the identity associated with a public key. Two of the best-known certificate systems dealing with authentication, and thus identity trust, are Pretty Good Privacy (PGP) and X.509.

Pretty Good Privacy (PGP) was created primarily for encrypting email messages using both symmetric and asymmetric cryptography to prevent changes during transmission. The intention was to be a "cryptographic tool for the masses", which bypasses the traditional hierarchical trust architecture by adopting the "web of trust" approach. As a matter of fact, the first version of the PGP system is generally known as a web of trust where there is no central trusted authority. Users sign each other's public keys and that leads into progressively forming webs of individual keys interconnected by these signatures. Trust webs are formed within virtual communities and between communities.

The X.509 authentication framework attempts to solve the same part of the trust management problem that PGP introducer mechanism attempts to solve, namely the need to find a trustworthy copy of the public key of someone with whom one wants to communicate. X.509 certificates

contain more information than PGP certificates (e.g. time interval that they are valid) but their basic functionality is still the binding of users to keys. However, X.509 differs from PGP in its level of centralization of the information. While any entity may act as an introducer in the PGP model, the X.509 framework requires that all certificates will be acquired from an official certificate authority (CA). Certificate authorities may be linked in a hierarchical or a mesh manner. Most public key infrastructure systems rely on certificate chains to establish a users identity, as a certificate may have been issued by a higher-level certificate authority. A PKI is typically composed of many certificate authorities linked by trust paths. A trust path links a relying party with one or more trusted third parties, such that the relying party can have confidence in the validity of the certificate of the user. Recipients of a signed message who do not have a relationship with the certificate authority that issued the certificate for the sender of the message can still validate the senders certificate by finding a path between their certificate authority and the one that issued the senders certificate.

Trust Scope

A certificate authority (either a user or a CA) issues digital certificates that are used in authenticating an owner's identity. This is necessary to establish a resource access or service provision trust relationship and may implicitly reduce the trustor's risk in dealing with the trustee. However, the policy governing what resources or services the trustee is permitted to access is not handled by the certificate infrastructure, but is left up to the application. The certification authority does not vouch for the trustworthiness of the owner, but it simply authenticates its identity.

Trust is specified in the form of certificates (or a chain of certificates, trusted path). In PGP, a public key is considered to be valid, and therefore trusted , if a certificate path exists between the recipient and a trusted introducer. Multiple paths may exist between the recipient and potential introducers. In X.509, the trustworthiness of the key is implied if there exists a certification path between the recipient and a trusted certificate authority. It is the responsibility of each entity to

query the system and to acquire keys as needed. In PGP what is more relevant is what makes an introducer a trusted introducer whereas in X.509 is what makes a CA a trusted CA.

There are two areas where trust is explicit in PGP as follows:

- Trustworthiness of public key certificate: Is a PGP public key certificate reliable or not? What is the degree of confidence that the binding between the ID and the key itself? There are three degrees of confidence that are attached to a certificates validity (undefined, marginal, and complete). The trustworthiness of the key is public.

- Trust to introduce key: How much a public key (indirectly referring to the owner) is trusted to be a competent signer of another PGP certificate? There are four levels (full, marginal, untrustworthy and don't know) that correspond to how much the user thinks the owner of this public key can be trusted to be introducer to another key. It is important to note that the actual meaning of these levels is not explicit. How the user decides what level to assign to an introducer is left up to that user. Each user manually assigns introducer trusts and the information is secret.

PGP allows users to adjust two parameters COMPLETESNEEDED and MARGINALSNEEDED which define the number of valid signatures needed to make a certificate valid. A certificate becomes completely valid if either one of these parameters are met. The user makes its own judgment on how much to trust the public key that is introduced and that is based on how much it trusts the introducer. When a user places trust in an introducer, implicitly it means that the user places some degree of confidence in the introducers capability to introduce valid certificates. For an introduced certificate to be valid, the user must directly trust its introducers. But given two marginally trusted introducers, there is no mechanism to treat one of them to be trustworthier than the other.

Collaborative environment capabilities

Both certificate systems can be viewed as recommendation systems for valid bindings between identities and public keys. Introducers, for example, are recommenders, however there is no mechanism for propagating introducer trust assessments within the PGP web of trust. We believe that by keeping the trust level of the introducers private, each user operates in seclusion without getting feedback from peers.

In PGP, it is possible to adjust two parameters COMPLETESNEEDED and MARGINALSNEEDED to specify criteria about the trustworthiness of keys. For example, a user can specify that it only completely trusts a key if it is marginally trusted by a meta-introducer and completely trusted by an introducer. But given two marginally trusted introducers, there is no mechanism to treat one of them differently than the other. The specification of criteria that assess the trustworthiness of keys relies only on introducer recommendations, but there are no mechanisms for re-evaluating the trustworthiness of registered keys or introducers once a certificate is added to the certificate ring chain.

In the X.509 model, users usually obtain a certificate from a single CA and not from multiple ones. Therefore, the application that examines certificates does not have the necessary inputs to compare multiple certificates for the same public key and owner signed by different CAs.

Dynamic trust features

PGP supports no re-evaluation triggers. Once a key is added to the PGP system, it remains valid at the user's discretion. Furthermore, the various trust levels for introducers remain unchanged, unless the user manually updates the values. As a matter of fact, due to PGP's lack of official mechanisms for the creation, acquisition and distribution of certificates it is considered appropriate only for personal communication.

X.509 maintains a validity period for certificates and there is a number of methods for responding to a request about the validity of an individual certificate. The most well known method

requires the retrieval of a lengthy list of invalid certificates, the Certificate Revocation List (CRL), to check the validity of a single certificate. The effectiveness as well as the efficiency of CRLs are still issues open to debate.

Composable trust constructs

Composable trust is not supported by either system, however transitivity is inherent in both models, without the possibility of opting out of supporting it.

7.1.2 Resource Access Trust: POLICYMAKER

System Description

PolicyMaker is a trust management system, developed at ATT Research Laboratories that integrates the specification of policy with the binding of public keys to the actions that are trusted to perform. Certificate frameworks such as PGP and X.509 described above do not bind access rights to the owner of the public key within the certificate framework. If access rights were to be associated with public keys, then two steps had to be taken. First, a public key is bound to its owner, and this task occurs within the certificate framework. Second, the identified owner is bound to access rights, and this task occurs outside the certificate framework. PolicyMaker combines authentication and authorization into a single system by using certificates to authorize their legitimate owner to perform specific actions.

The PolicyMaker appears to applications as a query engine. It accepts as inputs a set of local policy statements, a collection of credentials, and a string describing an action. PolicyMaker evaluates proposed actions by interpreting the policy statements and credentials. Depending on the credentials and the form of query, it can return a yes/no answer or additional restrictions that would make the action acceptable.

The basic function of PolicyMaker is to process queries. A query is a request to determine whether a particular *key* (or sequence of keys) is permitted to perform particular actions according to a local policy. Queries are of the form:

Key1, key2, REQUESTS ActionStrings

ActionStrings are application-specific messages that describe a trusted action requested by keys. The semantics of action strings are determined by the applications that generate and interpret them; they are not part of PolicyMaker. PolicyMaker processes queries based on trust information contained in *assertions*. An assertion binds a predicate, called a *filter*, to a sequence of public keys called an authority structure. Filters accept or reject actions based on what the holders of the keys are trusted to do. Assertions are of the form:

Source ASSERTS AuthorityStruct WHERE Filter

Source is the source of the assertion and *AuthorityStruct* contains the public key(s) to which the assertion is applicable. Finally, *Filter* is the predicate that action strings must satisfy in order for the assertion to be valid. In other words, PolicyMaker uses credentials to prove that the requested action complies with the policy, and this type of checking is known as compliance checking.

Trust Scope

PolicyMaker addresses authorization based on public keys and thus it focuses on establishing resource access trust and service provision trust. It is not clear if authentication is still possible in the absence of the external source with the functionality to perform identity-based certificate authentication.

Trust is specified as assertions. Each assertion states that the assertion source places trust in the public key that is contained in the authority structure to be bound to action strings that satisfy the filter. There are two types of assertions: certificates and policies. A certificate (also called signed assertion) is a signed message that binds a particular authority structure to a filter. A policy also binds a particular authority structure to a filter, however, policies are not signed and are unconditionally trusted.

A local policy may defer to third parties who are trusted to issue credentials for others, and it may also use filters that limit the extent to which these third parties are trusted. These third

parties may defer trust to others as well. Trust is considered to be monotonic in the sense that each policy or credential can only increase the capabilities granted. There is no revocation of existing capabilities.

Credentials is one kind of trust input used in PolicyMaker framework. PolicyMaker itself does not verify signatures on singed certificates. Instead, signatures are verified by some external program or function (e.g. PGP). The external program guarantees that the signature was valid for the identified public key. Furthermore, PolicyMaker does not implement certificate distribution or revocation services.

Collaborative environment capabilities

There is no support for recommendation schemes. However, it is possible to set up a policy that involves reputation information, but the recommendation semantics are application-specific. There is no support for feedback aggregation algorithms.

Dynamic trust features

As it was mentioned earlier, trust never decreases but it only increases.

Composable trust constructs

There are no explicit provisions for composable trust.

7.1.3 Resource Access Trust: KEYNOTE

KeyNote, the successor to PolicyMaker, was developed to improve on the weaknesses of PolicyMaker. It is based on the same design paradigms of assertions and queries with a couple of slight modifications. First, signature verification is integrated into the trust engine and second there is no support of multiple assertion languages. KeyNote provides a predefined assertion language.

Despite the additional functionality of signature verification and the elimination of choosing assertion languages, KeyNotes evaluation with respect to trust scope, collaborative features, dynamic properties and composable constructs is the same as PolicyMakers evaluation presented above.

7.1.4 Resource Access Trust: REFEREE

System Description

REFEREE (Rule-controlled Environment For Evaluation of Rules and Everything Else) is a trust management system for reaching access decisions that are related to Web documents. REFEREE uses PICS labels in the same theoretical framework as PolicyMaker to interpret trust policies. It allows both policies and credentials to return answers (which is a tri-value) and statement lists. A statement list is a collection of assertions, which is the justification for the answer. There are three possible values for the answer as follows:

- True, meaning that sufficient credentials exist to approve the action
- False, meaning that sufficient credentials exist to deny the action
- Unknown, meaning that the system was unable to find sufficient credentials to approve or deny the action.

The REFEREE system is a set of software modules, and each module handles a particular policy decision. All modules accept inputs and return a tri-value and a statement-list. The inputs are an action name and other arguments that provide information about the action and form the module's trust database. During the bootstrap phase, the application supplies REFEREE the unconditionally trusted assertions and a module database. A module database consists of action names, similar to a DNS server in that it allows a module to be referred to by an action name. During the query phase, the application supplies the action and other arguments such as credentials. The latter are passed onto the appropriate module from the module database. REFEREE runs the module's interpreter with the policy and list of arguments, and then returns an answer to the application.

Trust Scope

REFEREE provides resource access trust and service provision trust. Trust is specified using the trust language Profiles-0.92, which is a rule-based policy language designed to work with REFEREE. Rules are evaluated top down and the returned value of the last rule is the policy's returned value. It combines the theoretical framework of PICS and PolicyMaker in a unified system. A PICS label is the main trust input.

Collaborative environment capabilities

REFEREE inherits the rating scheme of PICS. REFEREE can process and resolve conflicting information from multiple sources by indicating whether all labels must agree or the existence of a single label of the desired type carries the decision.

Dynamic trust features

There is no support of trust re-evaluation. It is not clear if expiration of labels is handled by REFEREE or by the application.

Composable trust constructs

Composable trust is not supported.

7.1.5 Identity and Resource Access Trust: IBM TRUST ESTABLISHMENT

System Description

IBM has developed a role-based access control model that consists of three major parts: certificates, a Java-based Trust Establishment module and a Trust Policy Language (TPL). IBM view is that e-commerce is possible through trust establishment. The basic assumption is that any trust issues and requirements that are related to an e-business transaction could be solved using certificates. Certificates can be issued by various certificate authorities.They vouch for an entity's role. An example is a certificate stating that entity Y is a business partner.

The IBM trust establishment system is similar to PolicyMaker. The default certificate scheme

used is X.509v3, but other certificate formats could be supported. The basic functionalities of the system is to validate a certificate and map the certificate holder to a role.

Policies are specified in TPL using XML. These policies map permissions to roles. The primitive structure in TPL is a *group*. For each group, there are rules that define group membership, and this is done by indicating the certificates to check. An example of a policy is given in [30] :

$< POLICY >$

$< GROUPNAME = \text{“}self'' >$

$< /GROUP >$

$< GROUPNAME = \text{“}partners'' >$

$< RULE >$

$< INCLUSIONID = \text{“}partner''TYPE = \text{“}partner''FROM\text{“}self'' >$

$< /INCLUSION >$

$< /RULE >$

$< /GROUP >$

$< GROUPNAME = \text{“}departments'' >$

$< RULE >$

$< INCLUSIONID = \text{“}partner''TYPE = \text{“}partner''FROM = \text{“}partners''$

$< /INCLUSION >$

$< /RULE >$

$< /GROUP >$

$< GROUPNAME = \text{“}customers'' >$

$< RULE >$

$< INCLUSIONID = \text{“}customer''TYPE = \text{“}employee''FROM = \text{“}departments'' >$

$< /INCLUSION >$

$< FUNCTION >$

$< GT >$

$< FIELDID = \text{“}customer\text{”} NAME = \text{“}rank\text{”} >< /FIELD >$

$< CONST > 3 < /CONST >$

$< /GT >$

$< /FUNCTION >$

$< /RULE >$

$< /GROUP >$

$< /POLICY >$

Trust Scope

Trust Establishment is a tool for making access control decisions for strangers based on public key certificates. Trust Establishment extends traditional role-based access control systems by validating the certificate and then mapping the certificate owner to a role. The system does not make any access control decisions but it rather determines group membership. The policy provides the rules that determine how to map entities to roles. X.509 certificate and the rules are the trust inputs.

Collaborative environment capabilities

Policies only define groups that represent specific roles and rules on how to become a member of each group. There are no external sources of trust information.

Dynamic trust features

The results from the reasoning process are stored in a database. These results include the assignment of the certificate holders to particular groups. The resultant database is static and does not change considerable over time. Therefore, a role is permanently assigned to a certificate holder.

Composable trust constructs

There are no provisions for composable trust.

7.1.6 Identity and Resource Access Trust: TRUSTBUILDER

System Description

Trustbuilder is a trust negotiation framework. Trust negotiation is an emerging approach that exploits the properties of peer entities as a means for establishing trust. Trustbuilder addresses the problem of establishing trust relationships between entities, possible strangers, through credential exchange. A credential is defined as a digitally signed assertion by a credential issuer about a credential owner, and a sensitive credential is one that includes private information. The Trustbuilder negotiation model assumes that a negotiation is a sequence of credential disclosures, which alternates between a client and a server.

While a trust negotiation protocol defines the ordering of messages and the type of information messages will contain, a trust negotiation strategy controls the exact content of the messages, i.e., which credentials to disclose, when to disclose them and when to terminate a negotiation. Suppose during trust negotiation, entity A requests access to entity Bs resources. Entity B returns the access control policy that describes the credentials that A needs to present in order to access the resource. Entity A sends its policy to B that describes what credentials are needed from B in order for A to disclose its credentials. Entity B sends the credentials and then A sends its credentials too.

Trust Scope

The TrustBuilder framework focuses on a protocol for establishing trust in a particular context. TrustBuilder mostly deals with access control that is based on credentials and policies, with applicability in ecommerce scenarios. Credentials are the central pieces in this protocol.

Trust is specified via a credential expression language that is separated into two parts: property-based authentication language (PAL) and a role-based authorization language (RAL). A PAL policy defines one or more roles. A RAL authorization policy consists of a role-constraint expression and a PAL policy defining each role in the expression.

Collaborative environment capabilities

It appears that there are no feedback aggregation algorithms. It is not clear if past experience is considered during the negotiation phase. It seems that recommendation schemes are not part of the framework.

Dynamic trust features

It is not clear what happens to the binding of credentials to roles. Is it stored with the intention to be retrieved for future interactions? Or is it a temporary binding that expires after the specific interaction is terminated?

Composable trust constructs

There are no provisions for supporting composable trust.

7.1.7 CONTENT TRUST: PICS

System Description

PICS (Platform for Internet Content Selection) was developed to give users control over the Internet material they receive by facilitating the use of filters between the potential viewer and web contents. The PICS specification enables labels to be associated with Internet content, where a label (a.ka. rating, content rating) is a data structure containing information about a given document's contents. Rating service is an organization that assigns labels according to some rating system and then distributes them via label bureaus or on-line from HTTP servers. A rating system is a rating information method that specifies the dimensions used for labeling, the allowable values on each dimension and a description of criteria that are used to assign values. Each rating service chooses its own criteria for rating and the user must choose a rating service whose criteria are close to the ones the user would make. PICS provides for self-rating as well. Labels may be embedded in the header of HTML document, transmitted using any protocol that uses RFC 822-style headers or requested from a label bureau that runs the HTTP protocol.

A PICS-compliant application processes PICS labels and uses the user-defined filtering rules

to decide whether to accept or reject a web document. PICSRules is a language for writing filtering rules that allow or block access to URLs based on PICS labels that describe those URLs. PICSRules can specify one or more rating services to use, one or more label bureaus to query for labels and criteria about the contents of the labels that would be sufficient to make an accept or a reject decision. The effectiveness of the PICS framework lies in the expressiveness of the filtering language and the quality of the rating services.

Trust Scope

PICS addresses content trust because it decides on the appropriateness of received materials. A PICS-aware application or web browser requires the potential user to make decisions on which rating service(s) to use, which label bureaus to query, whether or not to use self-rating labels, and what policy clauses (a.k.a profile) implement the user's definition or interpretation of "appropriate" material. All these decisions are based on the user's perception of the trustworthiness of rating services and their respective rating systems as well as the trustworthiness of label bureaus. Based on the rating system that the rating service is using, the user assesses trust in the received material by specifying a set of policy clauses that determine whether or not a document is to be accepted or rejected. A trust assessment is a binary value and it depends on the truth of the policy propositions. The clauses are evaluated in the order given in the rule and evaluation stops at the first clause that is satisfied and the corresponding action is taken.

The trust inputs used are the label and the rules. Labels are generated by human reviewers, computer analysis of document contents, or a hybrid between third-party rating and self-rating. Labels are digitally singed to ensure that they are authorized by the specific rating service. A label can include a cryptographic checksum on the contents of the document. If the checksum matches a checksum of the current contents, then the label is valid. If not, then the document has changed since the label was created.

Collaborative environment capabilities

PICS is essentially a recommendation system but without the mechanisms to assess the trustworthiness of the rating services. There are no official guidelines or rules to assess the trustworthiness of the service or bureau. Furthermore, there is no support for reasoning about multiple labels from different rating services concerning a specific document. As a result, a user does not have the capability of choosing the "best" label for a document from a pool of labels regarding that document. There are currently no mechanisms to assess the quality of rating services or systems in a comparative manner. Even though filtering languages allow the usage of labels from multiple services, PICS-compliant software vendors may only allow the usage of a single service (for example, NetNanny).

PICS assumes that each service will distribute public keys in the way it chooses.

Dynamic trust features

PICS assumes that no keys will ever have to be invalided, an assumption that leads to security risks. A label may provide the date on which the specific rating expires. Depending on the PICS compliant software filter vendor, there are techniques to keel local copies of labels in order to reduce the performance delay due to fetching labels from label bureau. However, if the document changes prior to the expiration day and the label copy from the cache is used, there is the potential for viewing material that dont correspond to the saved label.

Composable trust constructs

Composable trust is not supported by existing filtering rules.

7.1.8 Content Trust: POBLANO

System Description

Poblano is a trust system by Chen and Yeager of Sun MicroSystems to build a decentralized trust model based on the JXTA platform. Poblano is based on the following assumptions:

- Trust has multiple components and Poblano is looking at a factor of trust which is based on the groups interests, or group content relevance.

- Users in peer groups can identify the source of the received information and can communicate their opinions on both the information received and its source.

- Opinions can be collected, exchanged and evaluated.

Poblano represents not only trust relationships between peers but also trust relationships between peers and *codats* (codat contains information, either data or code that is shared in a JXTA peer group). Poblano divides trust into three components:

- Trust between different peers (Peer Confidence): this is the peer's reputation based on both local knowledge of its codat and on other peers' evaluation of the peer in question. In addition to the confidence value, the average of popularity of each codat accessed from the specific peer for a given keyword is maintained.

- Trust between a peer and a codat (Codat Confidence): an evaluation of the codat of a given peer in a specific problem domain such as content relevance based on keyword matching and popularity. Popularity is monotinally increasing and is incremented at the provider each time the codat is requested.

- The risk factor (Risk): how much risk is associated with this peer? How reliable is it from a performance viewpoint (accessibility, integrity of codat)?

Poblano addresses the issue of calculating the overall trust based on these factors. It also suggests methods for processing, propagating, and updating trust relationships. Each peer has two tables: Peer confidence and codat confidence tables. When a peer does a search on a keyword, the following steps are taken: First, lookup the keyword in its own Codat Confidence table. If there is a local codat associated with the keyword, then retrieve local copy. Next, if there is no

local codat, lookup the keyword in the peer Confidence table. If there are peers highly associated with the keyword, forward the request to those peers. Each of those peers will perform the first two steps using their own tables. If such a peer has a codat related with the keyword, the peer will inform the requesting peer and terminate the search. The requester peer accesses the codat from the provider. The provider increases the popularity value of the codat it sends. The requester calculates its confidence in the received codat and saves the information in its table. The Codat confidence is calculated and is a function of codat confidence returned by remote peer and peer confidence of all peers along the path to codat confidence information discovery. New confidence information is propagated to all contributing peers codat path– (including the provider). All peers can update their codat confidence tables based on the new information. Note that although the actual codat is sent directly to the requester, the confidence value for the codat is propagated on the entire codat path.

Over time, a web of trust is built in the network.

Trust Scope

Poblano was built to perform reputation guided searching based on content relevance. Using the information stored in the Peer Confidence and Codat Confidence tables, Poblano users calculate trust values for peers and codats using predefined equations. Peer trust values are in the range [-1,4], where -1 indicates distrust and 4 indicates complete trust.

A peer computes the codat confidence of a codat path using the following equation:

$$CodatConf_{path} = \frac{1}{4n}(\sum_{i=1}^{n} PeerConf(P_i)) \times CodatConf$$

A peer updates his old codat confidence using its own rating and the codat confidence propagated from remote peers as follows:

$$newCodatConf = F(oldCodatConf, propagatedCodatConf, userRating)$$

$$newCodatConf = a \times oldCodatConf + b \times propagatedCodatConf + c \times userRating$$

A peer updates the peer confidence of the provider. The peer generates his own opinion about

the provider, without any external recommendations:

$$newPeerConf = F(oldPeerConf, setOfCodatConfRelatedToProvider)$$

$$newPeerConf = (oldPeerConf + \frac{1}{K} \sum_{a \in K} CodatConf_{provider}) \div 2$$

where K *= number of keywords related to the provider*

In order to make the above trust values meaningful for users, Poblano introduces the cooperation threshold. Peers decide whether to collaborate with one another based on a balance between trust and risk. If the peer confidence value is greater than the threshold, the peer is considered to be cooperative. Otherwise, the cooperation is too risky to the user. A user may override the Poblano recommendation for cooperation.

Codat confidence values are propagated and used for updating trust values for both peers and codats.

Collaborative environment capabilities

There is limited propagation of recommendations. As a matter of fact, codat confidence values are distributed only to the peers that are involved in the information discovery process. Peer confidence values are not propagated.

The equation for calculating codat trust is based on aggregating codat values but it is restrictive. A weighted average formula is used to derive the trust value of a target peer from a chain of peers. This is problematic in two ways. First, the formula is based on transitivity for deriving trust, a property that is still an open trust issue. Second, there exists a richer set of algorithms to aggregate feedback, including the weighted average technique, which gives the user more flexibility to choose the aggregation method.

Dynamic trust features

Trust is re-evaluated whenever a peer resides on a codat path. It appears that the peer has no choice but to update its tables whenever new information arrives. There are no policies or mechanisms to dictate when and how to update trust.

Composable trust constructs

Transitivity is inherent in the system but not composability.

7.1.9 Behavior Trust: TRUST-AWARE MULTICAST (TAM)

System Description

Trust-aware multicast system is a reliable multicast system that, unlike any other multicast protocol, deals with uncooperative nodes. Uncooperative nodes can modify, fabricate, replay, block and delay messages. The goal of TAM is to detect this behavior and adapt the multicast tree in such a way that more trusted nodes are located closer to the root node.

TAM is a single-source multicast system operating at the application level. A multicast group consists of one root node, which is the source of information, and many user nodes, which are functionally equivalent. The root node controls the entire group. TAM messages are constructed in a way that allows nodes to detect misbehavior of other nodes. For example, modified or fabricated messages are detected by message signature. TAM has four commands (Join, Leave, Report, and Relocate). A user node initiates the first three commands whereas the root node executes the last one. The Report command allows a user node to notify the root of uncooperative behavior and the Relocate command allows the root node to move a user node to a different location in the tree. A user node can only report problems about its parent.

According to the behavior of a user node n in the system, the root updates the user nodes quality, or trustworthiness, attribute $q(n)$. The root also maintains behavior history for each node in addition to q. TAM computes a level of trust for each node and adapts the multicast tree according to the trustworthiness of nodes, which leads to provide lower message delay to trustworthy nodes.

Trust Scope

The issue that is addressed is how a system can provide efficient and reliable information dissemination to well-behaved nodes when the messages are relayed via possibly uncooperative nodes. TAM is designed to disseminate freely available data without any access restrictions.

The root node receives reports for suspicious behavior and that's the major kind of feedback used in computing the quality attribute. Past behavior is also used. More specifically, trust is specified by three attributes: quality attribute *q(n)*, behavior of node as sender *hs(n*) and behavior of node as a receiver *hr(n)*. When a node joins the system, it is labeled 'trusted' and each of the above attributes is set to 0.

Quality attribute *q(n)* is determined by the number of reports from or against a node; that suggests a node behaves well. Another indication of a positive behavior is the willingness to maintain other nodes as children. These two signs of positive behavior affect q(n), which is increased by *ac+b*, where *a,b* are system parameters and *c* is the number of children that node *n* maintains. Note here that TAM controls the assignment of children to parent nodes.

The root also maintains *hs(n)* and *hr(n)*. Lower values of the attributes indicate that a node behaved better. When *n* reports against its parent *p*, the root increases *hs(p)* by *gq(n) + d* and *hr(n)* by *g'q(p) + d'*, where *g,g',d'* are system parameters. There are two levels of the system thresholds for each of *hs* and *hr*. If a node's history exceeds the 'soft' threshold, the *q(n)* is multiplicatively decreased. If it exceeds that 'hard' threshold, the node is labeled distrusted and its children are relocated. The thresholds are adjusted based on the runtime system load.

Collaborative environment capabilities

The recommendation scheme is the dissemination of complaints to the root against a parent node. There is no support of exchanging of information between two different root nodes. It is not clear how different kinds of suspicious behavior are treated, but we assume that they are treated equally.

Dynamic trust features

Whenever a report arrives at the root node, the appropriate *hs, hr* are updated. TAM also increases *q(n)* periodically.

Composable trust constructs

In a sense, TAM provides composable trust by using *q(n)* for evaluating the trustworthiness of a constructed multicast tree.

7.1.10 Behavior Trust: TCPA

System Description

TCPA (Trusted Computing Platform Alliance) is a consortium of companies that collaborate to make the computer platform trustworthy, with the primary goal to help users protect their information assets from compromise that can originate from external software attacks. It was originally formed by Compaq, HP, IBM, Intel and Microsoft, but it was later extended to cover a group of hardware, software, communications and technology vendors. The primary goal was to develop, define, and promote open standards for hardware-enabled trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices. However, what was evolved was an architecture to ensure that customer computers are trustworthy by content providers. It seems that there is very little conceptual difference between the TCPA framework and a PKI. Thus, TCPA suffer sfrom the same problems faced by PKIs.

7.1.11 General Trust: SULTAN

System Description

SULTAN (Simple Universal Logic-Oriented Trust Analysis Notation) trust management framework (TMF) is designed to facilitate the management of trust relationships. It is a collection of specification, analysis, and management tools. SULTAN consists of four components:

- Specification Editor: this is a toolkit for creating, storing, retrieving, editing and translating SULTAN specifications. Trust, distrust, negative and positive recommendations are specified using the specification notation. A SULTAN-to-Prolog translator is included as well.

- Analysis Tool: it incorporates an analysis notation, a query statement builder that makes it easier to formulate queries and a template of queries common to most situations, e.g. conflict of interest, separation of duties, implicit (and possibly dangerous) relationships, etc.

- Risk Service: it encapsulates the TMFs risk management strategy. The risk service collects risk information and performs risk calculations.

- Monitoring Service: The trust monitor keeps the information in the SULTAN TMF up to date, by gathering information on the outcome of transactions that involve domain users. The trust consultant is the component that the domain user interacts with. The user may ask questions that may help in the determination of a trust decision.

The novel aspects of the SULTAN TMF are manyfold. First, the specification is high-level and the the analysis notation is highly expressive, i.e. a wide range of queries may be constructed. Second, the TMF includes a component that handles risk. In addition, trust relationships are monitored in order to reevaluate them and experience (and other information) is used to enable better decision making.

Trust Scope

SULTAN does not perform access control or authentication. It simple returns the trust relationships, and these can be used as a basis for developing access control schemes. SULTAN accommodates the specifications of trust relationships and recommendations, in both assertions and rule formats.

There are two primitive constructs: the trust construct and the recommend construct. Trust and distrust are specified using the trust construct, while positive and negative recommendations are specified using the recommend construct.

The trust construct has the following syntactic form:

$$PolicyName : trust(Tr, Te, As, L) < -Cs;$$

The semantic interpretation of a statement in the form above is given in [27] as follows: Tr trusts/distrusts Te to perform As at trust/distrust level L if constraint(s) Cs is true. PolicyName is the unique name for the assertion. Tr, the trustor, is the entity that is trusting. Te, the trustee, is the entity to be trusted. As, the action set, is a colon-delimited list of actions (function names which effectively specify the context) or action prohibitions. The first parameter in an action name specifies the entity the action is performed on (whether on the trustor, or the trustee, or some other entity that is a component of either the trustor or trustee). L is the level of trust/distrust. L can be an integer or a label. Labels are converted to integers for analysis and management. For integer values of L, $-100 \leqslant L < 0$ represents distrust assertions and $0 < L \leqslant 100$ represents trust assertions. Cs, the constraint set, is a set of delimited constraints that must be satisfied for the trust relationship to be established. The delimiters are the logical *and* and logical *or*. Cs must evaluate to true or false.

The recommend construct has the following form:

$$PolicyName : recommend(Rr, Re, As, L) < -Cs;$$

Semantically (as explained in [27]), the above statement means that Rr recommends/does not recommend Re at recommendation level L to perform As if constraint(s) Cs is true. PolicyName is the unique name of the rule being defined. Rr, the recommendor, is the name of the entity making the recommendation. Re, the recommendee, is the name of the entity that the recommendation is about. L, the recommendation level, is the level of confidence in the recommendation being issued by Rr. L can either be a label or an integer. All labels are translated to integers for analysis and management. L is$\geqslant -100 and < 0$ for negative recommendations and L is > 0 and $\leqslant 100$ for positive recommendations. It is important to point out that the recommendation level and trust level are assumed to be independent of each other, unless otherwise specified. As, the recommended action set, is a colon delimited set of actions or action prohibitions that Rr recommends Re be trusted/distrusted to perform. Each action name stipulates the entity on which the action is performed.

A recommendation may result in a trust specification and vice versa. In addition to these two constructs, the auxiliary specification library contains two functions: the risk and the experience function. Experience, recommendations, risk, credentials are the inputs to the trust engine.

Collaborative environment capabilities

Recommend construct is one of the primitive constructs. There is no direct support for feedback aggregation algorithms other than embedding the desired aggregation scheme in the trust and recommend constructs as part of the constraints set.

Dynamic trust features

The SULTAN Monitoring system is responsible for updating the state, risk and experience information, as well as updating the risk profiles.

Composable trust constructs

There is no provision for reasoning about action and constraint sets of different trust relationships with the intention of composing them.

7.2 Analysis of How TMS Requirements are Met by Current TMSs

Now, we give more details on how each requirement is met by the systems described above. The results are shown in Table 7.1. We put "yes" if the system clearly meets the requirement, whereas "—" means that it does not meet it. If the system implements the feature not completely, the "some" is used. We have to note here that the evaluation of these systems is based on our interpretation of TMS requirements as described in Chapter 3. Additional capabilities of these systems are not included in the discussion below.

Table 7.1: TMSs and Generalized Trust Requirements

	Evidence Collection			Trust Analysis		Trust Evaluation			Trust Monitoring		Other
	R1.1	R1.2	R1.3	R2.1	R2.2	R3.1	R3.2	R3.3	R4.1	R4.2	R5.1
PK	—	—	—	—	—	—	—	—	—	—	yes
Policy Maker	—	—	—	—	—	—	some	some	—	some	yes
KeyNote	—	—	—	—	—	—	some	some	—	some	yes
REFEREE	—	—	—	—	—	some	—	some	—	—	yes
TE	—	—	—	—	—	some	some	some	—	—	yes
Trust Builder	—	—	—	—	—	—	—	some	some	—	yes
PICS	—	some	—	—	—	—	—	some	—	—	—
Poblano	—	—	some	—	—	some	—	—	some	—	—
TAM	yes	—	—	yes	some	some	—	—	yes	—	—
TCPA	—	—	—	—	—	—	—	—	yes	—	yes
SULTAN	yes	—	—	yes	—	some	—	some	yes	—	yes
Hestia	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

R1.1 Heterogeneous Forms of Evidence
R1.2 Selective Collection and Distribution of Evidence
R1.3 Dynamic Management of Evidence Streams
R2.1 Time-aware Trust Relationships
R2.2 Composable Trust Constructs
R3.1 Evidence Aggregation
R3.2 Evidence-to-Expectation Mapping Functions
R3.3 Expectation Satisfaction
R4.1 Trust Re-evauation
R4.2 Imputed Trust Mode Support
R5.1 Security Services and Certificate Management Tasks

7.2.1 R1.1 – Heterogeneous Forms of Evidence

It appears that almost all TMS use a single evidence type, which in most cases is the certificate. The only two exceptions are TAM and SULTAN. In TAM, the root node uses reports for suspicious behavior and past behavior to compute the quality attribute. SULTAN uses experience, recommendations, risk and credentials as inputs to its trust engine.

7.2.2 R1.2 – Selective Collection and Distribution of Evidence

The public keys of PGP users are available to all from many PGP key servers. The same holds for certificate authorities. Key servers and certificate authorities cannot refuse any user to obtain a public key from their databases. TrustBuilder appears that it accepts credentials but does not further distribute them. PICS users are able to select the label bureaus that they consider trustworthy. However, some PICS applications do not allow multiple label bureaus to be selected. TAM supports distribution of evidence but it is limited to the parent node. SULTAN, even though it uses various types of evidence, does not explicitly specify how evidence is collected and distributed.

7.2.3 R1.3 – Dynamic Management of Evidence Streams

Most of the TMS do not update their collaborator list. For example, in PGP, once a key is added to the PGP system, it remains valid at the user's discretion. Poblano peers decide whether to collaborate with one another based on a balance between trust and risk. If the peer confidence value is greater than the cooperation threshold, the peer is considered cooperative. TAM also supports some form of dynamic management of evidence streams that is enforced and it's not optional. Whenever the multicast tree rearranges its nodes, nodes may have new parent nodes to report to. SULTAN does not explicitly mention how this is done.

7.2.4 R2.1 – Time-Aware Trust Relationships

There is no explicit mention of time as an abstract concept in the TMS models, with the exception of TAM and SULTAN. TAM uses time explicitly when updating the quality attribute and the

behavior of nodes, as well as adjusting thresholds based on the runtime system load. SULTAN is also using time when re-evaluating trust relationships.

7.2.5 R2.2 – Composable Trust Constructs

A very limited form of trust composition construct is provided in TAM by using a specific equation that evaluates the trustworthiness of a constructed multicast tree.

7.2.6 R3.1 – Evidence Aggregation

REFEREE supports a limited form of evidence aggregation. It can process and resolve conflicting information from multiple sources by indicating whether or not all labels must agree or a single label carries the decision. In Trust Establishment, more than one credential may need to be examined for a specific policy. In Poblano, confidence values are aggregated to derive updated confidence values for paths and other peers. However, these equations are fixed for all Poblano peers. In SULTAN, there is no direct support for aggregation algorithms but it might be possible to specify them as constraints. Finally, in TAM it is not clear how different kinds of suspicious behavior are treated, but we assume that they are treated equally.

7.2.7 R3.2 – Evidence-to-Expectation Mapping Functions

PolicyMaker and its successor KeyNote map keys to actions. TrustBuilder maps credentials to roles.

7.2.8 R3.3 – Expectation Satisfaction

Expectation satisfaction is implemented in most systems, but the scope is rather narrow. Starting with the PolicyMaker and KeyNote systems, expectation satisfaction occurs whenever a key can be mapped to an action. In REFEREE, expectations take the form of rules, which are evaluated top down. The return value of the last rule is the policy's return value. Similarly, PICS expectations are expressed as rules that apply to labels. In Trust Establishment, expectation refers to the required fields that must be filled in with the information that is embedded in certificates. In TrustBuilder,

expectation satisfaction occurs when the required credentials are presented during negotiations. SULTAN uses the standard equality and inequality operators to indicate the required value for a requirement.

7.2.9 R4.1 – Trust Re-evaluation

The majority of the systems do not support dynamic changes to their established trust relationships. For instance, PolicyMaker considers trust to be monotonic (only increases) because each policy or credential can only increase the capabilities granted. Similarly, in Trust Establishment the database that stores the bindings of certificate holders to groups is static and does not change considerably over time. In REFEREE, it is not clear if expiration of labels is handled by system or the application. It is also not clear what happens to the bindings of credentials to roles in TrustBuilder environment. In Poblano, confidence values are updated only when a peer resides on a codat path. In this case, the peer has no choice but to update its tables. There are no mechanisms to dictate when and how to update these values. SULTAN, TAM and TCPA support trust re-evaluation.

7.2.10 R4.2 – Imputed Trust Mode Support

None of the systems support this feature. However, it might be possible in PolicyMaker and KeyNote to embed modes in the filters.

7.2.11 R5.1 – Secure Collection and Distribution of Evidence

Almost all systems rely on an external certificate authority, with the exception of PICS, Poblano and TAM.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

This dissertation concludes by presenting the contributions of the research work, its applicability and the future directions.

8.1 Contributions

This dissertation begun with a discussion of the generalized trust problem for indirect interactions. It was demonstrated that security research has not kept pace with this challenge because cryptographic mechanisms and authentication protocols do not provide any characterization of the data itself. Related to security is the notion of trust, and in particular trust management systems (TMSs). However, the current state-of-the-art in this research area also lacks the mechanisms to address end-to-end trust for indirect interactions.

Chapter 2 discussed the importance of trust management in critical infrastructures. Cross-communication and interaction between and within critical infrastructures, such as electric power grid, transportation, and intelligence agencies is required and mandated by the Homeland Defense Department. Information dissemination among the interested parties should accommodate their diverse trust requirements. In order to emphasize the need for trust in such infrastructures, a set of application suites for the electric power grid was presented.

Chapter 3 identified and documented the objectives and requirements that a trust management system must meet in order to provide dynamic and composable trust for indirect interactions. The requirements were categorized in accordance with the activities of a general trust management system. Additionally, a number of paradigms, new to trust management, were presented.

The detailed design of Hestia was the focus of Chapter 4. Hestia is a configurable trust management system that meets the TMS requirements. Its components were discussed, followed by three scenarios that illustrated the operation of the system.

Hestia's provision of configurable trust policies allows entities to set up policies for the functionality of Hestia components. Chapter 5 discussed the space of Hestia policies and a number of policies were specified in XML format.

The theory of sets and relations was then used to represent the trust relationships that exist in Hestia. In particular, trust was formally defined and its ontology, properties, and operations were studied in Chapter 6.

The current state-of-the-art in trust management was presented in Chapter 7. A brief overview of well-known trust management systems was given, and their capabilities and limitations were discussed in terms of the TMS requirements outlined in Chapter 3.

In summary, the key contributions of this research work are the following:

- An analysis of why dynamic and composable trust are both needed in many distributed application programs today which span multiple principals, particularly ones involving data aggregation or indirect interactions such as publish-subscribe applications
- A set of requirements which must be met by any trust management system in order to solve the generalized trust problem
- New trust paradigms which support the implementation of these requirements
- Informal model of Hestia trust management system that supports
  - dynamic trust management, including time-aware trust relationships
  - trust composition, in order to support indirect interactions and data aggregation
  - evaluation of both *access trust*, generalized policies for access control, and *data trust*, systematic reasoning about the quality of (possibly aggregated) data provided by a set of principals
- Trust formalisms, including trust ontology, properties, and operations.

Furthermore, the thesis of the dissertation:

> ***Thesis:*** Trust management techniques can be extended to meet the trust needs of architectures such as publish-subscribe that support indirect interactions between principals.

has been demonstrated by the design and formalisms of Hestia.

8.2 Generalized Trust and I3P

As it was mentioned at the very beginning of this dissertation, generalized trust is important to several aspects of the I3P research agenda. The existing baseline Hestia research already addresses I3P research goals for "Trust Among Distributed Autonomous Parties" (TADAP) and "Enterprise Security Management" (ESM). In particular, Hestia meets the following goals:

- One TADAP goal is striving for "dynamic security relationships in P2P settings". In addition to P2P, Hestia's architectural framework supports general formulation of trust relationships between principals, including publish-subscribe interactions , where intermediate entities act as forwarding agents that deliver data from its source to its destination. Integrating Hestia with a publish-subscribe framework allows for a rich experimental environment for exploring trust relationships. We have begun to investigate how the Hestia architecture will be implemented and integrated into an existing status dissemination middleware, GridStat, to validate and assess the conceptual framework of trust management for indirect interactions. GridStat [38, 25, 48, 43, 29] is a data dissemination middleware which is designed for the needs of the electric power grid and has been involved in a technology trial with a local electric utility since 2003 [19] and will soon be employed in a national energy lab. It is developed to provide better communications services for the electric power grid. GridStat takes advantage of the semantics of the data and supports a hierarchical plane that manages admission control issues and QoS requirements.

- The TADAP area also asks, "What is required to add 1-to-n peers to an existing trust relationship". Hestia to date supports not only this, but more general graphs involving composing trust across indirect interactions such as publish-subscribe, as well as supporting both access trust and data trust. Hestia provides a systematic way to assess an end-to-end trust relationship that involves a chain of *n* participants. The TADAP area also emphasizes, "ongoing research remains on authorization rather than on defining and communicating expectations". Hestia has explicit notions of expected behavior (expectations) and observed/measured behavior as well as a wide range of techniques that reason about allowed and actual behavioral values. Additionally, Hestia provides for different operational modes (similar to homeland security alert codes), and this allows for adjusting expectation specification depending on the current operational mode.

- The TADAP area stresses the need of "technology that does not rely on a previously determined trusted third party". One of Hestia's objectives is to operate in a collaborative environment. An entity's ability to monitor and manage all interactions in a large-scale distributed system is limited, and therefore it needs to rely on external opinions and experiences. A number of evidence acquisition mechanisms are integrated in Hestia, including recommendation networks, failure detectors, and QoS instrumentations. The wide range of evidence types allows for broader trust applicability, including *"determining the trustworthiness of readings from a remote sensor" and "monitoring user activity"*.

- The EMS area stresses the need to "define policies for secure and appropriate use of its information resources and translate those policies into practice". Hestia provides a solid basis for this. The EMS area also stresses the need for monitoring of user activity and Hestia supports these goals in many ways, for example, by supporting evidence types involving not only traditional security/trust evidence but a wide range of more general evidence such as performance anomalies, power grid anomalies, business relationships, etc.

8.3 Future Directions

The future directions for Hestia include the integration of the Hestia architecture into an existing status dissemination middleware, GridStat, to validate and assess the conceptual framework of trust management for indirect interactions, including a wide range of traditional trust evidence mechanisms plus more general ones such as the crash failure detectors, value failure detectors, intrusion detection systems, and QoS instrumentations. Hestia should not be tailored to the specific GridStat architecture, but instead it should be a reusable ”plug-in” service. One of the challenges of developing a reusable Hestia is making the correct and necessary architectural choices that will allow a stand-alone implementation to be easily integrated and reused by any other system. In addition to this challenge, it is nontrivial, or feasible, to generate all low-level policies that are necessary to support a high-level policy for a reusable and configurable Hestia.

Currently, there are no standardized evaluation metrics to assess the performance, effectiveness, and usefulness of TMSs. The evaluation of Hestia and comparison to other TMS should focus on devising a set of metrics that evaluate and validate TMSs in various dimensions, including performance, usability, and complexity. The diversity in metrics will enable the TMS evaluator to assess the entire spectrum of the system capabilities and features for given topologies and configurations, while at the same time assessing the complexity of the user-application interface. The main challenge for this task is to ensure that the proposed set of metrics covers all necessary aspects in assessing trust management systems. Currently, TMSs are using their own validation and assessment methods that are tied to the particular system and within a particular trust scope.

Another future direction is to explore the implications of Hestia in automated trust negotiations (such as the ones provided by TrustBuilder) and how it leverages the capabilities of such negotiations; this is an area that is gaining increased popularity. Hestia could be used to collect evidence about future collaborators, before the trust negotiation system starts its negotiation of trust. If evidence is available prior to negotiations, then credential and access control policies could be

adapted according to what is already known about the negotiating party.

Another future task could involve developing a suite of Hestia policies for electric power grid applications and studying their performance, tradeoffs, and applicability. There are two main challenges of developing such a suite of policies. The first one is the difficulty of defining high-level policies that conform to the organizations goals. The second challenge is the complexity of the space of the interoperability between organizations with heterogeneous policies.

As it was mentioned earlier, one of the design assumptions was that the trust service is considered to be trusted completely. Techniques are needed to assess the degree of trust placed in the service.

A final future task is the policy suite development. This task includes a policy language to specify the organization policies with a front-end tool. This tool will compile the policies into a representation that the policy engine can use. Furthermore, there is a need to devise techniques responsible for compile and runtime checking for potential policy conflicts.

APPENDIX

APPENDIX A

TRUST TERMINOLOGY

Trustor - Entity that makes a trust assessment.

Trustee - Entity about which a trust assessment is made.

Recommender - External source of trust assessments.

(Pairwise) Trust Relationship - This is the relationship between a trustor and a trustee.

End-to-end Trust Relationship - This is the relationship between a trustor and multiple trustees that are involved in the same interaction.

Pairwise Trust - It is the dynamic belief regarding the extent to which a particular trustee will act as expected for a specific context during a specific information lifecycle stage. This extent is subject to the satisfaction of expectations set by the trustor for the specific imputed trust mode.

End-to-end Trust - It is the dynamic belief regarding the extent to which a group of trustees will act as expected for specific contexts during a specific information lifecycle. This extent is subject to the satisfaction of expectations set by the trustor for the specific imputed trust mode.

Direct Trust - It refers to the trustors own trust assessments.

Experience - See Direct Trust.

Indirect Trust - It refers to trust assessments from external sources.

Recommendations - See Indirect Trust.

Trustworthiness - This is the extent to which the trustee is observed to honor the trust expectations. Trustworthiness is an extrinsic property of the trustee.

Imputed Trust Mode (trustfulness)- This is the extent to which the trustor is willing to take the risk of trust expectations being violated by the trustee.

Trust Level - Trust level quantifies the trustworthiness and the trustfulness of an entity.

Evidence - It refers to information that stands as proof of ones behavior, attitude, or external

attributes. Evidence must not be confused with forensic evidence, which must adhere to the standards of evidence that is admissible in a court of law.

Evidence Acquisition Mechanisms - Mechanisms that collect evidence for different kinds of trust properties. Recommendation mechanism is a specialized acquisition mechanism for recommendations.

Expectation - It is defined as a requirement and its allowed values that a trustor sets for a particular trust relationship.

Expectation Satisfaction- It occurs when a trustee's observed value for a requirement falls into the range of allowed values as set by the trustor.

Information Lifecycle - It is the interval during which information is created and consumed within an information dissemination system.

Information Lifecycle Stage- Information lifecyle is decomposed into three stages: generation, dissemination, and consumption.

APPENDIX B

TRUST RELATIONSHIPS BASED ON THE INFORMATION LIFECYCLE CONCEPT

This appendix focuses on describing the information lifecycle concept and how it could be used to evaluate end-to-end trust is different topologies. The various trust relationships are explained and illustrated for the peer-to-peer (P2P) and publish-subscribe systems.

B.1 Information Trust Classification

There are different classifications of trust and these depend on the context that trust is applied for. Grandison et al. [26] have identified different forms of trust in the literature relating to whether:

- access is being provided to the trustors resources (resource access trust)
- the trustee is providing a service (service access trust)
- trust concerns authentication (trustee certification)
- trust concerns infrastructure (infrastructure trust)
- trust is delegated (delegation).

The authors state that this taxonomy is not exhaustive but it rather provides a useful way of classifying trust for Internet services. Our trust model supports a different classification that is related to the role of the entity within the information lifecycle. This classification provides a different way of viewing different situations that involve trust in indirect interactions. Figure B.1(a) shows the trust relationship between an information producer entity and an information consumer entity. Each entity is simultaneously a trustor and a trustee, but with different trust requirements. In distributed systems such as publish-subscribe systems information is delivered by intermediaries and their trustworthiness must be assessed too.

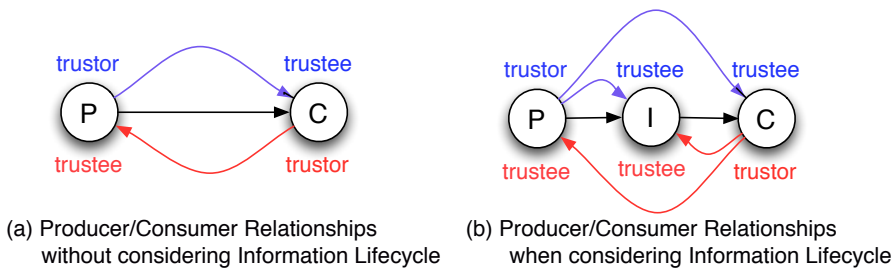

Figure B.1: Trust Relationships without and with Information Lifecycle

Information lifecycle was defined earlier as the interval during which information is created and consumed. By decomposition, we can break this three stages: generation, dissemination, and consumption. The entity responsible for the information at each stage is the information producer (generation stage), information dissemination medium (dissemination stage) and the information consumer (consumption stage). After decomposition, trust can be examined and evaluated at this finer granularity for each stage in the information lifecycle.

Figure B.1(b) illustrates the extended trust framework between an information producer and information consumer that takes into account the intermediary entity. Both the producer and consumer have a dual role (trustor and trustee) whereas the intermediary only acts as the trustee. There are two specialized forms of information trust in this setting: *Information Provider Trust* (IPT) and *Information Consumer Trust* (ICT). IPT refers to the subjective and dynamic belief placed by an information consumer entity (trustor) on an information provider entity (trustee) to provide information as expected. Similarly, ICT refers to the subjective and dynamic belief placed by an information provider entity (trustor) on an information consumer entity (trustee) to consume information as expected.

We argue that the new classification covers all trust forms mentioned above that are applicable in an information dissemination infrastructure. It should be possible to represent all five types of trust using the two specialized forms of trust. For instance, consider trustee certification, that refers to the certification of the trustworthiness of a trustee by a third party (certificate authority).

In our trust model, the information consumer entity will be the entity that receives the certificate and the information provider entity will be the certificate authority. Thus, the certificate receiver establishes an IPT between itself and the certificate authority regarding the certificate contents (Figure B.2).

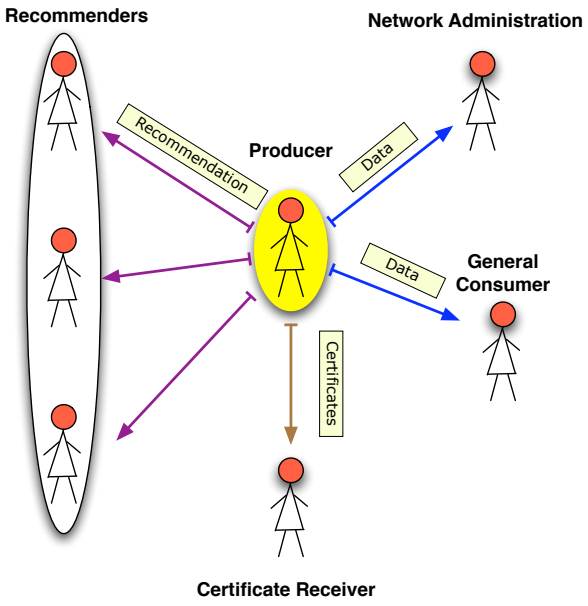

Figure B.2: Trustee Certification and IPT

The four trust relationships, illustrates in Figure B.1(b), are mapped to lifecycle stages. Consider each stage of the lifecycle in turn:

- In the generation stage, the information provider is the entity that assumes the role of the trust evaluator. It evaluates two ICT relationships, one for the information consumer and one for the information dissemination service.

- In the consumption stage, the information consumer assesses the trustworthiness of the information provider and the information dissemination medium by formulating the appropriate IPT relationships.

- In the dissemination stage, the dissemination medium has no expectations from either the provider or the consumer of the information.

We now show that this generic trust model can be applied to peer-to-peer and publish-subscribe paradigms.

B.2 Information Trust In Peer-to-Peer (P2P) Information Systems

Peer-to-peer file sharing systems allow users to collaborate in a dynamic manner and share information in large-scale distributed systems [36]. Information retrieval in P2P consists of two tasks: information discovery and information downloading. Information discovery includes forwarding query messages to locate peers that have the desired information and returning a list of those peers to the user who initiate the query. Information downloading is achieved with a direct connection with some peer that has the requested information.

There are four major entities in a P2P system: the downloading peer, the information dissemination medium, the uploading peer and the information discovery peer. We narrow the scope of the information discovery service to a single peer that locates target uploading peers. There are two information flows: the query request from the downloading peer to the information discovery service and the information stream from the uploading peer to the downloading peer. Based on these two information flows, the requirements for P2P require the following:

1. A downloading peer must be able to determine

    1.1. which uploading peers provide authentic files the contents of which match their descriptions

    1.2. which peers are trustworthy during the information discovery search

    1.3. the trustworthiness of the connection that facilitates file downloading

    1.4. the trustworthiness of the connection that provides the information discovery list of target peers

2. An uploading peer must be able to determine

   2.1. which downloading peers will not further upload without permission

   2.2. the trustworthiness of the connection that facilitates file downloading.

Each information flow lifecycle is considered separately in the lifecycle trust model. Therefore, two sets of the four trust relationships of the generic information sharing system are required, one set per lifecycle.

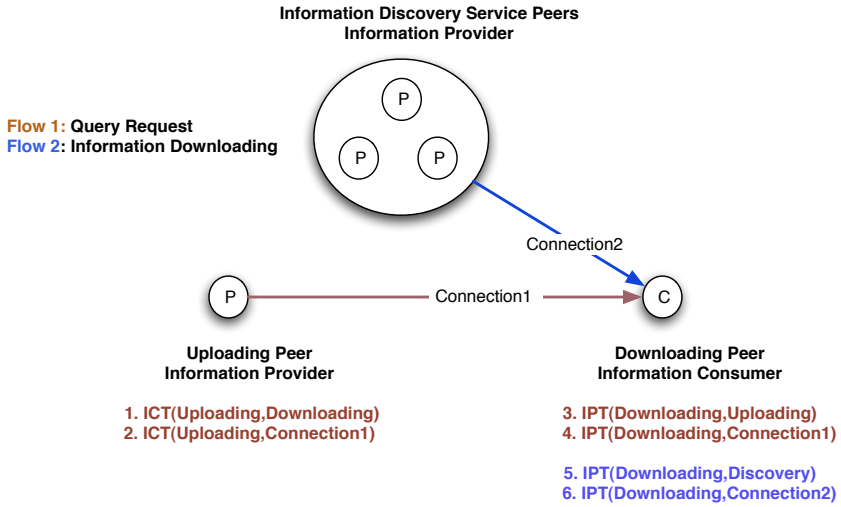

Figure B.3: Information Trust in Peer-to-Peer

The query request flow (Flow1 in Figure B.3) is initiated by the downloading peer and targets a discovery information service peer that provides a list of uploading peers. In the lifecycle trust paradigm, the requester peer is the consumer of the information and the discovery peer is its provider. Following the generic model of the trust relationships, the consumer establishes two IPT[1] relationships: one for the provider and another one for the connection (Relationships 5,6 in Figure B.3). Trust requirements 1.2 and 1.4 are satisfied by these two relationships. In a P2P environment, the information discovery service does not have any specific expectations from the downloading

[1]For brevity reasons, the attributes of both IPT and ICT are the trustor and trustee, in this order. We don't consider other trust relationships parameters.

peer and does not set the ICT relationships as dictated by the model ; the only trust issue would be to trust the requester peer that it does not abuse the service by launching denial of service attack.

The second information flow is the downloading of the requested file (Flow 2 in Figure B.3) from the uploading peer to the downloading peer. The uploading peer is the provider entity whereas the consumer entity is the downloading peer. The generic model is applied for this flow without any customizations. The provider formulates two ICT relationships to build trust for the downloading peer and the facilitating connection (Relationships 1,2 in Figure B.3). These two ICT relationships satisfy trust requirements 2.1 and 2.2. The equivalent IPT relationships (Relationships 3,4 in Figure B.3) are constructed at the consumer site, covering trust requirements 1.1 and 1.3.

The question that emerges is whether or not these trust relationships are feasible to establish and evaluate. At this point, we must consider the fundamental principle of anonymity that governs pure P2P systems. P2P systems did not make any provisions for trust evaluations. This is by no means a design fault, because P2P was built as an open shared file paradigm. However, malicious peers' behavior has destructive effects on the operation of the system. In order to defend and protect themselves, peers rely on reputation systems to obtain opinions about other peers. These schemes range from simple recommendation systems to more dynamically updated reputation systems that operate on some form of centralized reputation database. Assuming that a recommendation system exists for all entities, we believe that IPT and ICT relationships involving provider and consumer entitys are conceivable.

The focus on peer trustworthiness neglects the connection properties and how they affect the trustworthiness of the transferred data. A compromised connection is vulnerable to numerous attacks. The lifecycle approach explicitly considers the security (as part of the more general concept of trust) of the network. Due to the fact that a point-to-point connection (including transport layer TCP/UDP connections) is a degenerate form of an information dissemination medium, its trust properties are easy to verify by examining the security features of the direct connection. SSL, TLS, IPSec provide security services that guarantee authentication, integrity and confidentiality.

VPNs are also possible candidates for connecting extranet peers in a secure manner.

Finally, we consider the matter of synthesizing trust relationships. For example, trust relationships 3 and 4 of Figure B.3 assess at consecutive lifecycle stages the trustworthiness of the information during its generation (at the uploading peer) and its dissemination (by the connection). In order to minimize the risk of downloading malicious information, the consumer must decide the appropriate course of action when the uploading peer is untrustworthy and the connection is trustworthy, and vice versa. The contribution of the lifecycle approach is that it provides a model to enable the synthesis of the various relationships at each stage to reflect the overall trustworthiness of the information.

B.3 Information Trust in Publish-Subscribe Information Systems

Publish-subscribe is an asynchronous form of messaging that allows decoupling in time, space and flow[22]. A subscriber registers its interest with an event service and a publisher advertises its publication to the same event service. A network of event servers facilitates the distribution of event messages between publishers and subscribers. The dissemination of events is accomplished through matching algorithms that control the way event messages are delivered to the subscribers.

There are a few papers, such as the work by Wang et. al [46], dealing with security issues for a type of publish-subscribe system (content-based system) but without any special attention to trust issues. The major concern is that security mechanisms eliminate, in most cases, the spatial decoupling feature of the publish-subscribe system. Trust is also challenging in this system because of the intermediary network of store-and-forward servers between publishers and subscribers.

The main entities in a publish-subscribe system are the publishers, subscribers and the event servers. The event servers facilitate information dissemination as well as matching between publications and subscriptions. There are three information flows: message forwarding from publisher to subscriber, publication advertisement from publisher to event servers, and subscription request from subscriber to the event servers. The trust requirements for publish-subscribe are:

1. A publisher must be able to:

   1.1. infer which subscribes are likely to leak information

   1.2. rely on the event servers regarding message forwarding

   1.3 rely on the event servers for proper handling of its publication notification

2. A subscriber must be able to:

   2.1. infer which publishers publish trustworthy data

   2.2. rely on the event servers regarding message delivery

   2.3. rely on the event servers for proper handling of its subscription request

3. An event server must be able to rely on the other trustworthy servers that receives messages from or forwards messages to them

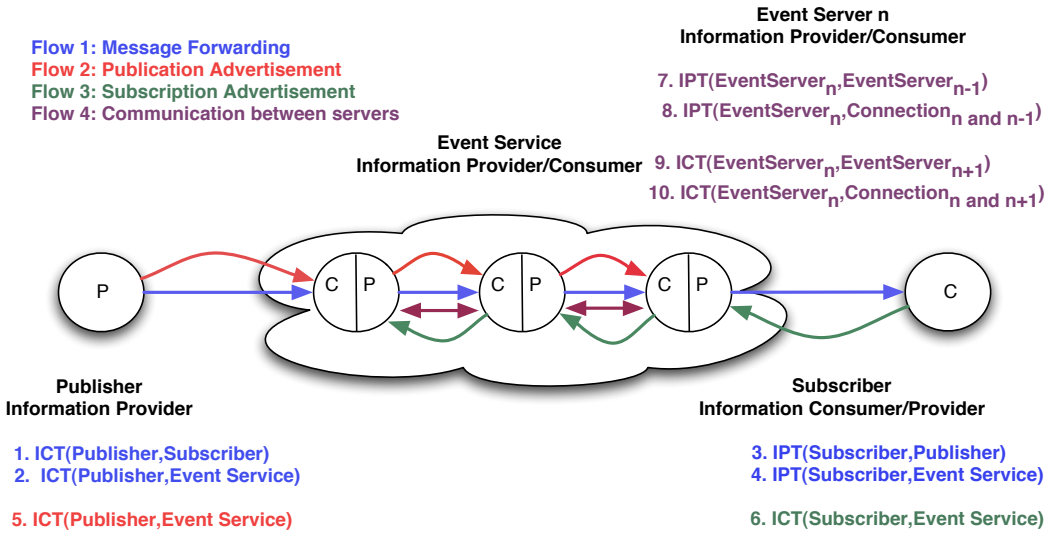

Figure B.4: Information Trust in Publish-Subscribe

Figure B.4 illustrates the trust relationships that satisfy the requirements mentioned above. Similar to P2P, each flow is examined independently in the lifecycle model. Starting from the first

flow of message forwarding, the trust relationships of the generic model are applied as they are. The provider is the publisher, the consumer is the subscriber and the event servers are the information dissemination medium. Hence, relationships 1, 2, 3, and 4 of Figure B.4 map to the trust requirements 1.1, 1.2, 2.1, and 2.2. The next flow is between the publisher and the event servers. The publisher expects that the event servers will operate correctly with respect to the proper forwarding and placement of its publication advertisement. We assume that the event servers have no expectations or demands from the publisher. Due to this simplification, there is only one trust relationship for this flow, which is the ICT between the publisher and the event servers (relationship 5 of Figure B.4). Trust requirement 1.3 is satisfied by this relationship. The third flow, the subscription notification request, is similar to the previous one and the same reasoning applies for this case too. The ICT between the subscriber and the event service (relationship 6 of figure B.4) maps to trust requirement 2.3.

The difference between the dissemination services of P2P and publish-subscribe is that the former is a point-to-point connection whereas the latter is a chain comprised of interconnected servers. Each event server acts as both a provider and consumer of information by forwarding to and receiving messages from its adjacent servers. Hence, the four trust relationships of the generic model apply for each pair of interconnected servers. Relationships 7 through 10 of Figure B.4 illustrate the trust establishment for server n, which is positioned as the middle server in the path chain $< n-1, n, n+1 >$. Both a provider and a consumer entity must synthesize the individual trust relationships for the servers involved in forwarding/matching operations so as to derive the trustworthiness of the information dissemination medium as a whole.

APPENDIX C

SECURITY REQUIREMENTS FOR STATUS DISSEMINATION MIDDLEWARE

Like any other distributed system, GridStat needs to provide the appropriate security services that will satisfy the needs of any application. Even though GridStat is to be deployed for the power grid, its security framework must be flexible to accomodate a wide range of security policies. A flexible, broad security framework allowing diverse security requirements within the same publish-subscribe architecture is needed. As of today, there is no such security framework tailored for the general publish-subscribe model or its more specialized form of status publish-subscribe model.

C.1 Related Work

In order to define the security framework for GridStat, the security requirements for a status dissemination publish-subscribe middleware are investigated. Status dissemination middleware is considered a specialization of the content-based publish-subscribe model,where the content collapses into a single status event. Furthermore, moving one step further, the content-based publish/subscribe paradigm itself falls in the category of distributed systems that span over different administrative domains. The relationships described are illustrated in Figure C.1.

Unfortunately, there is no work done on designing a framework covering the security aspects of the communication model mentioned above. In [46], the authors present both the security requirements and the challenges for a content-based publish-subscribe system. Wang et al. divide the security requirements for a content-based publish-subscribe system into two categories:

- Security requirements for the application, which refer to the security issues between publishers and subscribers
- Security requirements between the infrastructure and the publishers, subscribers.

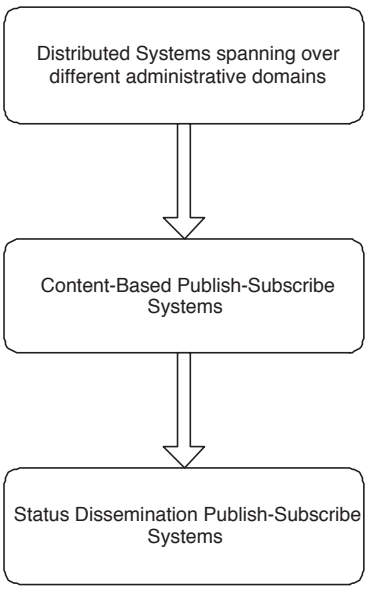

Figure C.1: Status Dissemination and Publish Subscribe Relation

For each category, the main security services of integrity, authenticity, confidentiality and availability are defined with respect to the category's domain space. For example, authentication between publishers and subscribers (category 1) refers to the end-to-end authentication where point-to-point captures the concept of proof of identity between immediate points of communication (category 2).

One of the paper's contributions is the identification of challenges faced in undertaking the task of implementing a complete security framework for the publish-subscribe model. The authors warn that enforcing current security mechanisms might lead into converting the publish-subscribe paradigm into a point-to-point or multicast communication model. The existing challenges are defined as the ones below:

- It is not trivial to design a security architecture for the publish-subscribe system that will provide a flexible security framework allowing diverse policies to be implemented within the same infrastructure.

- Service integrity, which refers to the correct operation of the internal publish-subscribe infrastructure, is only guaranteed by contacting a comprehensive fault tolerance analysis enumerating malicious faults and consequences.

- The requirement of confidentiality of information travelling within the network contradicts the basic content-based principle.

- Subscription confidentiality proposed methods are impractical and their performance and efficiency have not been evaluated yet.

- Publication confidentiality (or access control), which refers to the delivery of information to legitimate subscribers, can be handled by the infrastructure. In order to do this, the publish-subscribe framework needs to provide an interface for the application to specify their control policy and a mechanism to support such policies.

Miklos in [35] also emphasizes the serious security problems that are present in publish-subscribe systems and points out that a uniform solution accommodating all security scenarios listed in [46] might not be feasible. Instead, the author investigates those attack scenarios that can be prevented with an access control mechanism tailored for the publish-subscribe communication model. The main idea is to provide an access control mechanism (covering only positive access rights) that will control what notifications the subscribers are authorized to receive and what advertisements publishers can publish. His control mechanism for the subscriber is captured as follows:

Step 1 The subscriber sends the subscription together with its credentials to the dispatching network host

Step 2 The access control component on the host checks whether the policy rules and the attached credentials allow the subscriber to insert the subscription

Step 3 If permission is granted, the subscription is inserted.

The mechanism for the publisher is similar to the above. His access control scheme defines actions for credentials. Policy rules authorize the presenters of the credentials to perform those actions that are associated with the access control filters for the presented credentials. Access control filters (upper bound and lower bound) defining control rules for both publishers and subscribers are defined using the covering relation notation introduced by Carzaniga et al. [14]. For example, there are two ways to grant access rights to subscribers holding credential C:

- **Granting access rights based on upper bound subscribe filters**: Suppose that the policy rules define the upper bound subscribe filter (with credential C) to be *(string message newproduct)*. Then the subscriber is allowed to subscribe to *(string message newproduct, integer 10)* but not allowed to subscribe to *(string weather sunny, integer* $temperature \leq 25$*)*. In other words, the upper bound access control filter u covers the subscription as a subscription filter $u \sqsubset s$.

- **Granting access rights based on lower bound subscribe filters:** Suppose that the lower bound subscribe filter is *(string message any)*. Then the subscriber is allowed to subscribe to *(string message newproduct)* but not *(string message newproduct,* $price \leq 100$*)*. In other words, the lower bound access control filter l covers as an advertisement the subscription $l \sqsubset s$.

The described mechanism only works for a single credential C. Multiple credentials will complicate the procedure. Moreover, the paper does not mention how to compose two access control filters originating from two different sources. The assumption is that the access control filters are defined at the edge routers by the local host.

Bertino et al. in [10] also present an access control mechanism for large scale data dissemination systems. The authors, stressing the fact that existing dissemination systems lack any kind of access control mechanisms, attempt to design such an access control facility that will accommodate the following:

- Deny access to certain users
- Deliver confidential data only to authorized users
- Selectively disseminate information based on special agreements.

Their access control model is based on two concepts; user profiles and authorization domains. User profiles capture user interests and credentials where support of authorization domains aims in grouping together information objects to which the same access control policies apply. An information object consists, among other things, of a unique identifier, set of concepts and a set of weights associated with each concept.

As mentioned earlier, the user profile is built based on an interest profile and a set of credentials. An interest profile can be either explicitly specified by listing the information object identifiers that are of interest to the user or by dynamically evaluating concept queries in order to get the object identifiers. On the other hand, credentials are defined to be instances of credential types, where credential types are described by a unique identifier and a set of attributes of the form attribute name, attribute domain, value. The user profile contains the following information:

- user identifier
- lifespan of the profile
- update frequency that indicates how often the user wants to be notified of new relevant information objects
- relevance threshold that indicates the minimum similarity score that an information object must have against the interest profile in order to be returned to the user
- credentials
- interest profiles

Since the proposed access control mechanism is intended for a dissemination system, there are only two access modes that are supported; namely, notification and propagation (selective and total) as shown in figure C.2.

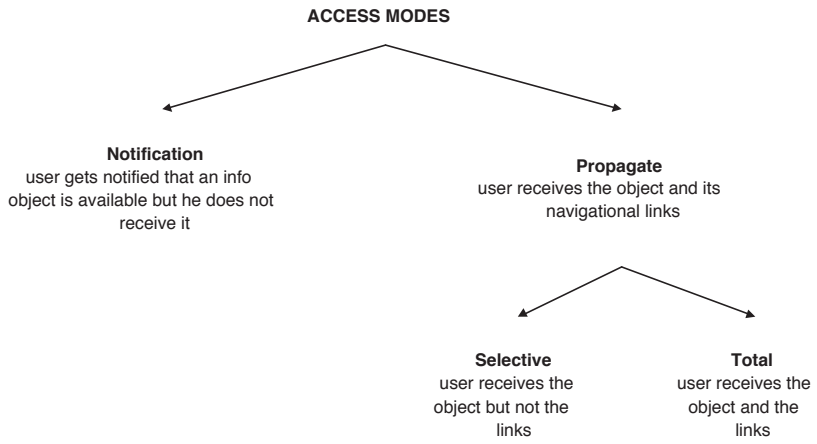

Figure C.2: Access Modes

The final element of the proposed access control mechanism is defining the actual access control policies. This is accomplished by imposing a set of conditions on user credentials, which are defined by using credential expressions, authorization domains and access control modes. An example of such access control policy is *(manager AND managertype = senior, Project Budget, total propagate)*, meaning that a user holding a senior manager credential has a total propagation access right on the information object named Project Budget.

C.2 Initial Security Approach for GridStat

The content-based publish-subscribe model lacks a security framework that will protect it from malicious attacks. The status dissemination middleware that we are trying to built is a specialization of the content-based publish-subscribe, implying that the lack of a security framework is inherited to this model too. GridStat needs to provide the policies and mechanisms for defining security properties as desired by a variety of applications with different security needs.

In order to define the security framework for the GridStat middleware, the following were considered:

- What security services are available for large scale status dissemination systems
- What entities need protection and what security services will be provided to them
- Where security will be placed (relative to the TCP/IP mode)

C.2.1 Classification of security services

The following list outlines the security services that network systems may implement:

- Confidentiality (privacy) — usually provided through encryption
- Authentication — usually provided through digital signatures and certificates
- Integrity — usually provided through hash functions (keyless) or digital signatures
- Nonrepudiation — requires a trusted third party
- Access Control — ACL, capabilities, role based, domain access control, access based on membership
- Availability — protect access to resources in order to avoid denial of service attacks

C.2.2 Entities that need protection and security services that can be provided

GridStat must provide the following services to the various GridStat entities and structures:

- Status items
  - Authentication and Integrity of message: unaltered content during transmission
  - Authentication of origin: verification of publisher's identity
  - Confidentiality: secrecy might be required during transmission

  - Availability: status items need to be available to the legitimate subscribers
  - NonRepudiation: for audit purposes

- Hierarchy of QoS Brokers
  - Authentication: identity verification
  - Authorization and access control: QoS brokers will be responsible for delivering commands to lower-level QoS brokers and therefore an authorization scheme must be in place.

- Status Routers
  - Authentication: the intermediate points of communication need to authenticate themselves to each other.

- Publishers
  - Authentication: this might not be easy (or feasible) in the case of sensor devices because it depends on the communication protocol used between a substation and the devices (serial lines and one-to-one connection or any other protocol). The assumption is that the edge status router is able to authenticate its publishers using a technique that is beyond the GridStat scope. With the same token, publishers' identity can be verified through their edge status router.
  - Authorization and access control: this refers to the ability of a publisher to advertise a status item. In addition to this, publishers should also be able to impose restrictions on the receivers of their posted status.

- Subscribers

  - Authentication: this is a critical element because subscriptions will get validated based on who the subscriber is. Probably a credential service (distributed or hierarchical one) will be an essential part of the overall framework for distributing credentials to subscribers.
  - Authorization and access control: a subscriber is either allowed to receive an event or not. A scalable access control scheme must de defined for controlling subscriptions.

- Resources (bandwidth)
  - Access Control: this is taken care of by employing an access control scheme for controlling publishing and subscribing.
  - Availability: since QoS guarantees are provided, all the information flow is prearranged and any new flow is subject to bandwidth availability. Furthermore, by restricting subscriptions and filtering publications, an attempt is made to prevent denial of service.

The above entities and their security requirements can be partially represented by the security requirements guideline presented in [46]. The QoS hierarchy is not depicted in the guideline but it will be useful to check how the status dissemination middleware as we envision it fits in that security framework:

- Authentication of end-to-end entities (publishers and subscribers): as mentioned earlier,it is not known yet how publishers authenticate themselves to their edge status router. Either a secure connection needs to be established between each sensor device and the edge status router (and in this way preventing identity theft) or the subscriber will authenticate the edge status router instead of the publisher. This is feasible due to the hierarchical nature of the GridStat naming scheme.

- Authentication of point-to-point entities: internal server communication can be provided by employing IPSec or using SSL connections.

- Information Integrity: the integrity of the status item can be guaranteed by IPSec. Edge status routers can also digitally sign the message using PKI.

- Subscription Integrity: this refers to the protection of the subscription from unauthorized modifications possibly during runtime. In GridStat environment, this can be restated as a question of how a subscription can be modified without authorization and be who. Edge status routers and internal cloud servers might attempt a subscription modification by altering their own forward/routing tables. Note that each server can only modify the next hop because this is the only thing they know regarding a subscription. So, if a server redirects a status item message to another internal server than the intended one, the message will not be expected and the receiving server will drop it and possibly notify its leaf QoS broker. Furthermore, the redirection of the status message will result in the loss of the message and therefore the subscriber will not receive the status item. Missing status items will also result in further actions (notifying edge status router, leaf QoS broker).

- Service Integrity: this is one of the challenges mentioned in the authors' paper [46]. GridStat supports a redundant path mechanism but this is a partial solution. Due to the GridStat framework, cooperation of a number of faulty or malicious servers need to be present in order for an unauthorized subscriber to receive a message. In addition to this, integrity mechanisms will prevent the modification of a message by an internal status router. An inexpensive security mechanism for verifying a status item that travels through redundant paths is to let the subscriber wait until N different copies of the status event arrive and then employ a vote/compare mechanism. Note that we will not have full Byzantine security any time soon.

- Information Confidentiality: Can the infrastructure perform status routing without the publishers trusting the internal servers with the status value? Currently, information subscription cannot be supported because the internal status routers forward messages based on the status identifier, which is composed by the publisher name and status type. It is possible that a malicious router can collect status item sets and supply them to interested parties that are not authorized to view the values of those status items. The value of each status item is very short-lived so maybe there is no real gain in trying to protect their value.

- Subscription Confidentiality: Can the subscribers obtain dynamic status data without revealing their subscription function to the publisher or the internal server infrastructure? The status router infrastructure has no knowledge of what subscriptions the publish-subscribe system supports due to the fact that all subscriptions are validated by the QoS Broker hierarchy. A subscriber submits its subscription request to its edge status router and from there it gets forwarded to the leaf QoS Broker. If the subscription's QoS requirements are satisfied and all access control policies are satisfied, then the internal servers are notified for adding a new entry to their routing table. Therefore, Gridstat's architectural design supports this security requirement.

- Can publishers control which subscribers may receive particular publications? An access control scheme needs to be implemented for controlling the status dissemination. This is considered to be a nontrivial task.

- Accountability: edge status routers and leaf QoS Brokers may want to charge the recipients of status information originated from their publishers. Furthermore, the QoS Broker hierarchy might want to charge for bandwidth consumption. Since all subscriptions are stored in the hierarchy, it is feasible to gather information on who is using what and how much.

- Availability (of bandwidth): customized publication control by filtering at the edge status

routers as well as controlling the number of subscriptions are techniques for partially preventing denial of service attacks.

C.2.3 Security Placement and Requirements

Suppose that we are using the TCP/IP model. In this case, there are five places where security mechanisms can be implemented. We do not want to have the same security services implemented at more that one place. Duplication of services does not add any real value into the overall security framework and it rather slows down the system performance. Therefore, based on figure C.3, the services mentioned above could be allocated as follows:

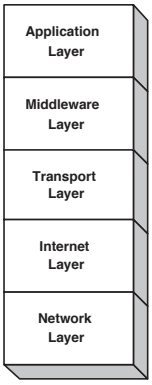

Figure C.3: TCP/IP model

- **Application Layer:** Above GridStat middleware, a power application runs. It is the responsibility of this layer to authenticate the users. Note that there is a difference between authenticating the user and authenticating the message. In the first case, user authentication can be performed using passwords, biometrics, smart cards, etc. Message authentication verifies that the received message came from the alleged source and has not been altered.

- **Transport Layer:** Integrated into the TCP stack is the Transport Layer Security (TLS), which is the same as SSL (which is placed between the Application and the Transport layers). We will probably not going to use either of those.

- **Internet Layer:** IPv6 supports extension headers that provide packet (and therefore message) integrity, authentication and privacy. By implementing IP level security, the distributed system does not need to take any further security actions against authenticating source address, packet integrity and privacy and key management. CERT reported that the most serious attacks include IP spoofing and eavesdropping. IPSec, which is mandatory for IPv6 but optional for IPv4, can encrypt and/or authenticate all traffic at the IP level.

- **Network Layer:** That's the layer where hardware encryption is placed. Dedicated line is needed with no intermediate routing and with the same crypto-box at the other end. There is also hardware that implements IPSec, and that will speed up cryptographic operations.

- **Middleware:** So far, authentication and confidentiality can be addressed and covered by existing mechanisms implemented at the TCP/IP layers. Access control and nonrepudiation are the services that need to be addressed in the layer.

Even though the categorization of the security requirements outlined in [46] is intended for a content-based publish-subscribe system, it can also be applied for the specialized case of status dissemination middleware with some modifications as shown in the previous section. GridStat also supports a hierarchy of QoS brokers in order to manage QoS. A pure status dissemination middleware does not need to support a hierarchy because maybe there is no need to support any QoS properties.

Therefore, the security requirements categories for GridStat include the ones defined in [46] (for a general status dissemination middleware) as well as the ones below that take into consideration the hierarchy:

- Security requirements between the QoS hierarchy and the publish-subscribe infrastructure

- Security requirements between the QoS hierarchy and the publishers, subscribers.

A complete analysis of the security requirements of a status dissemination middleware managed by a QoS hierarchy can be defined by filling in the table C.1, where applicable. Note that due to lack of space, we use the following abbreviations:

I:Integrity, C: Confidentiality, A: Authenticity, NR: NonRepudiation, Av: Availability, AC: Access Control

Table C.1: Security Requirements for GridStat

	I	C	A	NR	Av	AC
publisher AND subscriber						
infrastructure AND (publisher, subscriber)						
hierarchy AND infrastructure						
hierarchy AND (publisher, subscriber)						

BIBLIOGRAPHY

[1] Eastern interconnection phasor project. www.phasors.pnl.gov.

[2] Institute for information infrastructure protection. www.thei3p.org.

[3] Trust issues in pervasive environments. Technical report, University of Southampton and QinetiQ, September 2003.

[4] Protected critical infrastructure information (pcii) program, 2006. www.dhs.gov.

[5] Reinventing the internet. Technology Quarterly, The Economist, May 2006.

[6] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33th Hawaii International Conference on System Sciences (HICSS)*, pages 1769–1777, Maui, Hawaii, January 2000.

[7] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the ACM New Security Paradigms Workshop*, pages 48–60, September 1997.

[8] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of ACM*, 26(11):832–843, November 1983.

[9] D. Bakken, Z. Zhan, C. Jones, and D. Karr. Middleware support for voting and data fusion. In *DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks*, pages 453–462, Gőteborg, Sweden, 2001. IEEE Computer Society.

[10] E. Bertino, E. Ferrari, and E. Pitoura. An access control mechanism for large scale data dissemination systems. In *Proceedings of the 11th International Workshop on Research Issues in Data Engineering*, Heidelberg, Germany, 1–2 April 2001. IEEE Computer Society.

[11] M. Blaze, J. Feigenbaum, and A. D. Keromytis. Keynote: Trust management for public key infrastructures. In *Proceedings of the 6th International Workshop on Security Protocols*, Cambridge, UK, April 1998.

[12] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, page 164, Washington, DC, USA, 1996. IEEE Computer Society.

[13] Vincent W. Buskens. *Social Networks and Trust*, volume 30 of *Theory and Decision Library. Series C, Game Theory, Mathematical Programming, and Operations Research*. Boston, London Kluwer Academic Publishers, 2002.

[14] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[15] Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. Referee: trust management for web applications. *Comput. Netw. ISDN Syst.*, 29(8-13):953–964, 1997.

[16] David D. Clark, Karen Sollins, John Wroclawski, and Ted Faber. Addressing reality: an architectural response to real-world demands on the evolving internet. *SIGCOMM Comput. Commun. Rev.*, 33(4):247–257, 2003.

[17] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, 1989.

[18] CSI/FBI. *Computer Crime and Security Survey*, 2005.

[19] GridStat Avista Demo. *www.gridstat.net/avista.htm*. Washington State University, 2006.

[20] Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems*. Addison-Wesley Longman, Inc., 2000.

[21] EPRI, Palo Alto and Electricity Innovation Institute, Palo Alto. *The Integrated energy and communication systems architecture*, 2004. www.epri.com/IntelliGrid.

[22] Patrick Eugster, Pascal Felber, Rachid Guerraoui, and Anne-Marie Mermarec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, July 2003.

[23] U.S. Canada Power System Outage Task Force. *Final report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*, March 2004. https://reports.energy.gov/BlackoutFinal-Web.pdf.

[24] Andy Franz, Radek Mista, David E. Bakken, Curtis E. Dyreson, and Muralidhar Medidi. Mr. fusion: A programmable data fusion middleware subsystem with a tunable statistical profiling service. In *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, pages 273–278, Washington, DC, USA, 2002. IEEE Computer Society.

[25] Kjell Harald Gjermundrød. Flexible qos-managed status dissemination middleware framework for the electric power grid. Technical Report EECS-GS-011, Washington State University, 2006.

[26] Tyrene Grandison and Morris Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–16, 2000.

[27] Tyrone Grandison. Trust specification and analysis for internet applications. Technical report, Ph.D. Thesis, Imperial College of Science Technology and Medicine, Department of Computing, London, 2001.

[28] Trusted Computing Group. *TCG Specification Architecture Overview*. TCG, 2004.

[29] Carl H. Hauser, David E. Bakken, and Anjan Bose. A failure to communicate: Next-generation communication requirements, technologies, and architecture for the electric power grid. *IEEE Power and Energy*, 3(2):47–55, March/April 2005.

[30] Amir Herzberg, Yosi Mass, Joris Michaeli, Yiftach Ravid, and Dalit Naor. Access control meets public key infrastructure, or: Assigning roles to strangers. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 2, Washington, DC, USA, 2000. IEEE Computer Society.

[31] Frederick S. Hillier and Gerald J. Lieberman. *Operations Research*. Holden-day Inc., 1974.

[32] Institute for Information Infrastructure Protection. *Cyber Security Research And Development Agenda*, January 2003.

[33] Ryan A. Johnston, Carl Hauser, K. Harald Gjermundrød, and David Bakken. Distributing time-synchronous phasor measurement data using the gridstat communication infrastructure. In *Proceedings of 39th Annual Hawaii International Conference on System Sciences (CD/ROM)*, Kauai, Hawaii, January 2006.

[34] Seung Jun, Mustaque Ahamad, and Jun (Jim) Xu. Robust information dissemination in uncooperative environments. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 293–302, Washington, DC, USA, 2005. IEEE Computer Society.

[35] Z. Miklos. Towards an access control mechanism for wide-area publish-subscribe systems. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002.

[36] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno

Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[37] A. G. Phadke. Synchronized phasor measurements in power systems. *IEEE Computer Applications in Power*, 6(2):10–15, April 1993.

[38] GridStat Project. *www.gridstat.net*. Washington State University, 2006.

[39] Platform For Internet Content Selection. *http://www.w3.org/PICS/*. W3C, 2005.

[40] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. The MIT Press, 2006.

[41] William Stallings. *Cryptography and Network Security: Principles and Practice (Second Edition)*. Pearson Education, 1998.

[42] Sun Microsystems. *Poblano: A Distributed Trust Model for Peer-to-Peer Networks*, 2000.

[43] Kevin Tomsovic, David E. Bakken, Mani Venkatasubramanian, and Anjan Bose. Designing the next generation of real-time control, communication and computations for large power systems. *Proceedings of the IEEE (Special Issue on Energy Infrastructure Systems)*, 93(5):965–979, May 2005.

[44] Edna Ullmann-Margalit. Trust, distrust, and in between. In Russell Hardin, editor, *Distrust*, volume VII of *The Russell Sage Foundation Series on Trust*, pages 60–82. Russell Sage Foundation, 2004.

[45] John Vacca. *Public Key Infrastructure: Building Trusted Applications and Web Services*. AUERBACH, 2004.

[46] Chenxi Wang, Antonio Carzaniga, David Evans, and Alexander Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35)*, Big Island, Hawaii, January 2002.

[47] M. Winslett, T. Yu, K.E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. The trustbuilder architecture for trust negotiation. *IEEE Internet Computing*, 6(6):30 – 37, November/December 2002.

[48] Felix F. Wu, Khosrow Moslehi, and Anjan Bose. Power system control centers: Past, present, and future. *Proceedings of IEEE*, 93(11):1890–1908, 2005.

[49] Philip R. Zimmermann. *The official PGP User's Guide*. MIT Press, 1995.

[50] John A. Zinky, David E. Bakken, and Richard E. Schantz. Architectural support for quality of service for CORBA objects. *Theory and Practice of Object Systems*, 3(1), 1997.