# DUALTRUST: A TRUST MANAGEMENT MODEL FOR SWARM-BASED AUTONOMIC COMPUTING SYSTEMS

By

WENDY MARIE MAIDEN

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

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2010

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of WENDY MARIE MAIDEN find it satisfactory and recommend that it be accepted.

David E. Bakken, Ph.D., Chair

Carl H. Hauser, Ph.D.

Deborah A. Frincke, Ph.D.

Ioanna Dionysiou, Ph.D.

# ACKNOWLEDGEMENTS

I am so grateful to Dr. David Bakken for allowing me to be one of his graduate students! He introduced me to the topic of trust management, worked to ensure that the classes I needed were offered at the Tri-Cities branch campus where I am located, and made many opportunities available to me over the years of my graduate studies. I also appreciated his contagious enthusiasm.

It was also Dr. Bakken's suggestion for me to ask Dr. Ioanna Dionysiou to be on my advisory committee. I have benefited greatly from her work with me. Her counsel has been wise and her encouragement invaluable. Her flexibility made communicating over the 10-hour time zone difference to Cyprus no problem at all.

I also want to thank Dr. Deb Frincke for the many strategic opportunities she has made available to me through her leadership of the Information and Infrastructure Integrity Initiative (I4). I4 funded much of the trust management research for the Cooperative Infrastructure Defense (CID) framework that became the basis for this thesis. The guidance provided by Dr. Frincke and principal investigator Dr. Glenn Fink was invaluable. The Information and Infrastructure Integrity Initiative is an internal (Laboratory Directed Research and Development) investment of the Pacific Northwest National Laboratory in Richland, WA. The Pacific Northwest National Laboratory is managed for the US Department of Energy by Battelle Memorial Institute under Contract DE-AC05-76RL01830.

I also appreciate Dr. Carl Hauser for his above-and-beyond efforts to ensure that his students really learn and are well-prepared for their future endeavors.

This thesis and degree would also not have been possible without the consistent, loving, patient, encouraging, and prayerful support of my husband Wayne.  My parents and good friends have also prayed and encouraged me through many challenges during my graduate school days.

There are so many more to name – my PNNL line and project managers who allowed me the flexibility I needed to manage work and school together, Sharon Johnson whose editing advice helped make this thesis look its best, Sidra Gleason who graciously answered questions and coordinated paperwork for me in Pullman since I am in the Tri-Cities, and PNNL's legal, copyright, and intellectual property experts who enabled me to use my PNNL work for my WSU degree without running afoul of anything.

I couldn't have done it without all of you.  Thank you!

# PUBLICATIONS

Portions of this thesis were previously published, or have been submitted for publication, in the following works:

- W.M. Maiden, J.N. Haack, G.A. Fink, A.D. McKinnon, and E.W. Fulp. "Trust Management in Swarm-Based Autonomic Computing Systems." In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE Computer Society, Brisbane, Australia, 2009. Copyright 2009 IEEE. Reprinted with permission. [1,2]
- [In review] W.M. Maiden, I. Dionysiou, D.A. Frincke, G.A. Fink, D.E. Bakken. "DualTrust: A Distributed Trust Model for Adaptive Trust Management in Swarm-Based Autonomic Computing Systems," *15th European Symposium on Research in Computer Security*, LNCS, 2010.

[1] This material is reprinted and posted with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of Washington State University's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this material, you agree to all provisions of the copyright laws protecting it.

[2] Wendy Maiden was the lead author on this paper and the trust management researcher for the Cooperative Infrastructure Defense (CID) project. The paper's co-authors are the CID principal investigator (Glenn Fink) and researchers who focused on other aspects of CID. They provided the CID system description details, including the list of design elements for host platform protection, and the "red team" concept which inspired the mobile monitor agent concept in this thesis. They also provided helpful and much-appreciated peer reviews and guidance throughout my first research experience, including the writing of my first lead-author research paper.

- W.M. Maiden. *Trust Management Considerations for the Cooperative Infrastructure Defense Framework: Trust Relationships, Evidence, and Decisions*, Pacific Northwest National Laboratory Technical Report PNNL-19117, Pacific Northwest National Laboratory, Richland, Washington, 2010. Available at http://www.pnl.gov/main/publications/external/technical_reports/PNNL-19117.pdf. [3,4]

[3] Copyright status (http://www.pnl.gov/notices.asp): Documents provided from this web server are sponsored by a contractor of the U.S. Government under contract DE-AC05-76RL01830. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce these documents, or to allow others to do so, for U.S. Government purposes. These documents may be freely distributed and used for non-commercial, scientific and educational purposes.

[4] General disclaimer (http://www.pnl.gov/notices.asp): This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

# DUALTRUST: A TRUST MANAGEMENT MODEL FOR SWARM-BASED AUTONOMIC COMPUTING SYSTEMS

## Abstract

by Wendy Marie Maiden, M.S.
Washington State University
May 2010

Chair: David E. Bakken

For autonomic computing systems that utilize mobile agents and ant colony algorithms for their sensor layer, trust management is important for the acceptance of the mobile agent sensors and to protect the system from malicious behavior by insiders and entities that have penetrated network defenses. However, certain characteristics of the mobile agent ant swarm – their lightweight, ephemeral nature and indirect communication – make the design of a trust management model for them especially challenging.

This thesis examines the trust relationships, issues, and opportunities in a representative system, assesses the applicability of trust management research as it has been applied to architectures with similar characteristics, and finds that by monitoring the trustworthiness of the autonomic managers rather than the swarming sensors, the trust management problem becomes much more scalable and still serves to protect the swarm. This thesis then proposes the DualTrust conceptual trust model. By addressing the

autonomic manager’s bi-directional primary relationships in the ACS architecture, DualTrust is able to monitor the trustworthiness of the autonomic managers, protect the sensor swarm in a scalable manner, and provide global trust awareness for the orchestrating autonomic manager.

# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........ iii

PUBLICATIONS ........ v

ABSTRACT ........ vii

LIST OF TABLES ........ xii

LIST OF FIGURES ........ xiii

CHAPTER

1. INTRODUCTION ........ 1

2. SWARM-BASED AUTONOMIC COMPUTING SYSTEMS ........ 5

   2.1. Autonomic Computing Systems ........ 5

   2.2 CID as a Swarm-Based ACS ........ 6

3. INTRODUCTION TO TRUST AND TRUST MANAGEMENT ........ 11

   3.1 Motivation for Trust Management ........ 12

   3.2 Definition of Trust Management ........ 13

   3.3 Trust Context ........ 15

   3.4 Foundations for Trust ........ 16

   3.5 Trust Concerns in Swarm-Based ACSs ........ 17

4. SURVEY OF LIGHTWEIGHT TRUST MANAGEMENT FRAMEWORKS 18

   4.1 Wireless Sensor Networks ........ 18

   4.2 Mobile Ad-Hoc Networks (MANETs) ........ 28

4.3 Applications of Existing Research ........ 37

5. SECURING THE MOBILE AGENT SWARM ........ 39

5.1 Characteristics of Mobile Agent Systems ........ 39

5.2 Mobile Agent Threats and Their Countermeasures ........ 40

5.3 Challenges in Applying Trust to Protect Mobile Agent Swarms ........ 41

5.4 Analysis of Existing Reputation Management Techniques for Securing Mobile Agent Systems ........ 45

5.5 Applications of Existing Research ........ 49

6. SECURING THE AUTONOMIC MANAGERS: RELATED TRUST RESEARCH ........ 50

6.1 Characteristics of P2P Systems ........ 50

6.2 Trust Model Constraints ........ 51

6.3 Threats in P2P Environments ........ 51

6.4 Cryptographic Keys in P2P ........ 53

6.5 Evidence for Trust Establishment ........ 54

6.6 Threat Countermeasures in Trust Evaluation ........ 54

6.7 Trust Management Frameworks Proposed for P2P Systems ........ 56

6.8 Differences in Protecting Autonomic Managers ........ 66

7. TRUST RELATIONSHIPS IN A SWARM-BASED AUTONOMIC COMPUTING SYSTEM ........ 67

7.1 Detailed Analysis of CID Trust Relationships ........ 68

7.2 CID Trust Relationships Most Likely to Benefit ........ 81

8. DUALTRUST: A DISTRIBUTED TRUST MODEL FOR MANAGING THE TRUST OF AUTONOMIC MANAGERS ........ 83

8.1 DualTrust Foundations ........ 83

8.2 Architectural Design Constraints ........ 87

8.3 DualTrust Architecture for Evidence Storage and Distribution ........ 88

8.4 Trust Evaluation ........ 94

8.5 Reputation-Enhanced Relationship Scenarios ........ 98

9. CONCLUSIONS AND FUTURE WORK ........ 108

9.1 Conclusions ........ 108

9.2 Future Work ........ 109

BIBLIOGRAPHY ........ 110

# LIST OF TABLES

7.1 Overview of direct trust relationships in CID ........................................................ 68

7.2 CID relationships pertaining to protection and control of resources........................ 70

7.3 CID relationships pertaining to trust in a service ..................................................... 75

8.1 Reputation evidence for Sentinel C.......................................................................... 97

8.2 Reputation of Sentinel C using all evidence ............................................................ 97

8.3 Reputation scores of evidence data reporters ........................................................... 98

8.4 Reputation of Sentinel C using evidence from Sentinels with reputation scores >= 0.85 ....................................................................................................................... 98

# LIST OF FIGURES

2.1 CID hierarchy.........................................................................................................7

2.2 CID entities compared to the structure of an ACS. ...................................................9

7.1 CID trust relationships ..........................................................................................69

8.1 Reputation evidence collection scenario..................................................................91

8.2 The Sergeant's hierarchical relationship to the Sentinels..........................................94

8.3 High-level architecture.........................................................................................101

# CHAPTER ONE

# INTRODUCTION

Trust continues to be a major issue in the acceptance of mobile agent systems, but trust management techniques can be used to establish the trust needed for acceptance.[5] Trust management augments the capabilities of traditional authentication and access control techniques. Whereas traditional security techniques emphasize *prevention* of security failures, trust management (particularly a reputation-based approach) serves to *detect* security gray areas that are not especially suited to the traditional approach to authentication, as well as potentially malicious quality of service (QoS) issues (e.g., resource starvation). This detection capability is critical when the mobile agents are part of a security system that will need to withstand the attacks of those wishing to thwart the system's traditional security measures.

Trust management techniques must be adapted to the unique needs of the system architectures and problem domains to which they are applied. Most mobile agent trust management research efforts focus either on tightly constrained e-commerce-style architectures or on heavyweight agent-collaboration architectures. In contrast, this thesis considers the less constrained and lighter weight architecture required by mobile agent swarm-based autonomic computing systems. In swarms that are inspired by nature (e.g.,

[5] The introductory text in this chapter was originally published in Maiden WM, JN Haack, GA Fink, AD McKinnon, and EW Fulp. 2009. "Trust Management in Swarm-Based Autonomic Computing Systems." In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE Computer Society, Brisbane, Australia. © Copyright 2009 IEEE. Reprinted with permission.

ant colonies), the individual agents are ephemeral, act without centralized coordination, and may have no direct communication with each other. These characteristics require substantially different trust management techniques than are typically used with mobile agents. Therefore, after looking at the trust issues and opportunities in swarm-based autonomic systems [34], [35] such as the research framework being developed at the Pacific Northwest National Laboratory to detect and respond to security problems in complex cyber infrastructures, this thesis analyzes the applicability of trust management research as it has been applied to architectures with similar characteristics, specifies required characteristics for trust management mechanisms that are to be used for monitoring the trustworthiness of the entities in a swarm-based autonomic computing system, and presents a representative architecture that has the required characteristics.

The research contributions of this thesis are:

- Analysis of trust management architectures for various types of distributed systems, showing how the requirements and constraints of each type of system have been addressed in representative trust management architectures
- Characterization of the unique trust requirements of swarm-based autonomic computing systems (ACSs) and analysis of the limitations of the existing research efforts to address the problem of trust management in swarm-based ACSs,
- Analysis of the trust relationships, trust evidence, and trust decisions in a representative swarm-based ACS, and
- A recommended trust management model for swarm-based ACSs showing how

the architecture addresses the requirements. Algorithms for calculating trust and trust evidence storage and distribution mechanisms are discussed.

The thesis is organized as follows:

Chapter 2 provides an overview of autonomic computing systems (ACSs) and describes a representative example of an ACS that uses mobile agent swarms as its sensors.

Chapter 3 provides an overview of trust management – why trust is needed in distributed systems, the definition of trust management, a categorization of the purposes for which trust management systems are used and several examples of each, and the contexts and foundations for trust. It also discusses why trust is a concern in swarm-based ACSs.

Chapter 4 looks at lightweight trust management frameworks used in resource-limited distributed systems such as wireless sensor networks and mobile ad-hoc networks to understand the unique characteristics of these systems and how trust frameworks presented in the research literature are uniquely adapted to these characteristics.

Chapter 5 examines how the mobile agent swarm in the Cooperative Infrastructure Defense (CID) framework differs from the typical research scenarios used to motivate trust frameworks for mobile agent systems.

Chapter 6 surveys characteristics of peer-to-peer (P2P) systems, threats that can occur in these systems, the role of trust as a countermeasure to these threats, and why existing trust research is insufficient for autonomic managers of swarm-based ACSs.

Chapter 7 details the trust relationships in the CID system.

Chapter 8 introduces DualTrust, a trust model that reflects the dual nature of the autonomic manager’s horizontal peer relationships and vertical reporting relationship and benefits both the autonomic manager community and the swarming sensors.

Chapter 9 discusses conclusions and future research directions.

# CHAPTER TWO

# SWARM-BASED AUTONOMIC COMPUTING SYSTEMS

This chapter provides an overview of autonomic computing systems (ACSs) in general and of an example swarm-based ACS.[6]

## 2.1. Autonomic Computing Systems

ACSs provide automated, flexible, context-aware application of human-derived policy to the overall maintenance of computer systems [27]. The general architecture of an ACS is a set of autonomic elements that cooperate according to business logic. Each autonomic element has two general parts, the managed element that usually corresponds to underlying hardware or legacy software, and the autonomic manager that provides the autonomic behavior of the element and the interface between the managed element and the ACS. Within the autonomic element, the autonomic manager uses sensors to probe the state of the managed element and effectors to configure and maintain it. Each autonomic manager also has external-facing sensors and effectors that enable it to act as part of a larger system, adjusting itself and other elements according to the business policy that dictates the behavior of the overall system. Some ACSs also have an orchestrating autonomic manager that translates and communicates policy to the

[6] The text in this chapter was originally published in Maiden WM, JN Haack, GA Fink, AD McKinnon, and EW Fulp. 2009. "Trust Management in Swarm-Based Autonomic Computing Systems." In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE Computer Society, Brisbane, Australia. © Copyright 2009 IEEE. Reprinted with permission.

autonomic elements and collaborates with orchestrating autonomic managers from other ACSs.

## 2.2 CID as a Swarm-Based ACS

Swarms are generally composed of large numbers of relatively simple agents that act without centralized coordination toward a common global goal [22]. The Cooperative Infrastructure Defense (CID) framework being developed by the Pacific Northwest National Laboratory to meet the challenge of securing complex cyber infrastructures is an ACS that utilizes swarming mobile agents as sensors to provide input to the autonomic managers.

The CID framework uses a hierarchy of rational agents, as shown in Figure 2.1, to monitor hosts within and across enclaves (i.e., security domains) to provide infrastructure protection in complex, interconnected environments, such as an electric power grid. CID’s hierarchy includes human Supervisors who are ultimately responsible for the actions of their cyber-defense systems. Rather than taking humans out of the loop or putting them down at the level where every decision requires their action, CID attempts to place humans in the right loop to maximize their efficiency without damping system responsiveness [20]. In Figure 1, the enclaves that comprise the infrastructure are depicted in blue, yellow, and red.

At the lowest level of the hierarchy, CID employs a swarm of lightweight, mobile agents called Sensors with diverse, narrowly targeted classifiers. Sensors roam throughout an enclave, comparing each host they visit with previously-visited hosts to detect differences that may indicate security problems.

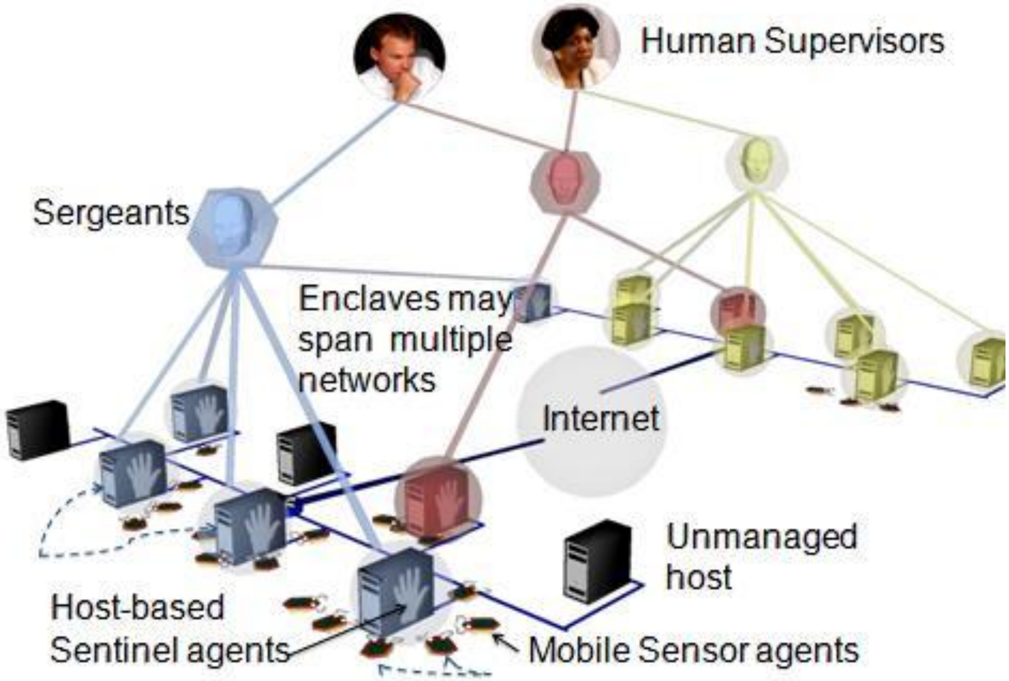

Figure 2.1: CID hierarchy.
Elements from top to bottom are: human Supervisors, Sergeants, host-based Sentinels, and mobile Sensors.

Sensors report their findings to the host-based Sentinel agent that uses semi-supervised learning to diagnose the reported potential problems. Sentinels reward Sensors that find potential problems. This activates the Sensors, and they leave behind a digital pheromone path that attracts the attention of other kinds of Sensors to further characterize the problem. The Sentinel fixes diagnosed problems according to policy and reports the problem and resolution to the enclave-level Sergeant agent. In its report, the Sentinel credits the Sensors whose findings resulted in successful diagnosis.

The Sergeant is responsible for overall enclave security, and it dialogues with the human Supervisor to create executable policy statements to pass to the Sentinels. The Sergeant controls the Geography, which is the set of hosts that the Sensor agents are allowed to visit. The algorithm by which the Geography is created is designed to turn the discrete network landscape into a continuous grid so that each Sentinel has a set of neighbors that adjoin it and the number of network hops between neighbors is minimized. The Sergeant also tracks new solutions and may share them with its peers, the Sergeants of other security enclaves. The Sergeant may also be authorized by its Supervisor to make service-level agreements with other Sergeants.

CID enables many characteristics of ACSs, such as self-protection and self-healing. However, CID concentrates on adaptive security across an infrastructure as opposed to general maintenance of an individual machine or enclave. An infrastructure is defined to be a multi-organizational entity whose members share a unified purpose. The boundaries separating ownership of computing equipment within an infrastructure may be fuzzy and inconsistent. A prime example is an electrical power grid where companies in the grid may share management, maintenance, and housing of computer, network, and Supervisory Control and Data Acquisition (SCADA) resources for the unified purpose of providing reliable power to a large geographic area.

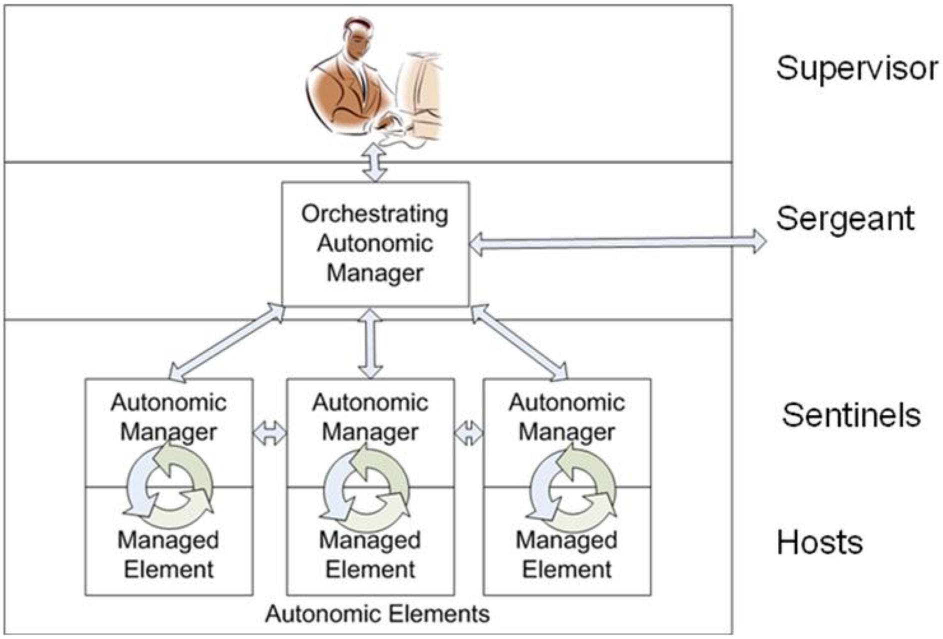

Figure 2.2: CID entities compared to the structure of an ACS.
Not shown: Mobile agent Sensors travel between hosts providing input to the Sentinels. Graphic adapted from An Architectural Blueprint for Autonomic Computing, 4th edition, IBM Corporation [23].

As shown in Figure 2.2, each autonomic element in CID consists of a host computer as the managed element and a CID Sentinel as its autonomic manager [23]. The Sentinel provides the effectors while the sensors are provided by mobile CID Sensor agents that adaptively sense conditions across the entire enclave. The Sergeant resembles an orchestrating autonomic manager, coordinating Sentinels, dictating enclave policy, and collaborating with Sergeants from other enclaves in the infrastructure.

CID uses a modified ant colony algorithm to sense and affect system stability. Control is fully distributed among the independent Sensor and Sentinel agents. Although individual Sensor communications are limited to local, stigmergic[7] messaging, the overall effect creates useful emergent behaviors that characterize the swarm as a whole. The CID approach differs from ant colony optimization [13] in that, rather than trying to find a near-optimal solution to any particular problem, CID seeks to cover an entire search space regularly using stigmergy to attract mobile agents to troubled hosts in the enclave. In some ways, the implementation resembles a lightweight collective intelligence [46] swarm, borrowing heavily from social insect behaviors. Thus, CID is a swarm-based, autonomic, cyber-defense system that allows multiple organizations to cooperate in their cyber defense while respecting proprietary boundaries and requiring minimal human intervention.

[7] From http://en.wikipedia.org/wiki/Stigmergy: "Stigmergy is a mechanism of indirect coordination between agents or actions. The principle is that the trace left in the environment by an action stimulates the performance of a next action, by the same or a different agent. In that way, subsequent actions tend to reinforce and build on each other, leading to the spontaneous emergence of coherent, apparently systematic activity. Stigmergy is a form of self-organization. It produces complex, seemingly intelligent structures, without need for any planning, control, or even direct communication between the agents. As such it supports efficient collaboration between extremely simple agents, who lack any memory, intelligence or even awareness of each other."

# CHAPTER THREE

# INTRODUCTION TO TRUST AND TRUST MANAGEMENT

Trust has many different interpretations depending on the context[8]. It can pertain to authentication, authorization, competence, reliability, integrity, dependability, timeliness, accuracy, or any combination of these properties. Authentication and authorization are the "hard" side of trust; they are determined by the policies and credentials of a structured environment. The remaining attributes are the "soft", and often social, side of trust. They speak to quality of service (QoS) and are not black-and-white, but rather measured in degrees and changeable over time as one's perception of an entity's reputation is formed through direct personal experience and/or through the recommendations of others based on their experience.  This thesis uses the following definition of trust: "Trust is the firm belief in the competence of an entity to act as expected such that this firm belief is not a fixed value associated with the entity but rather subject to the entities' behavior and applies only within the context at a given time." [4]

Trust is only useful when it is managed in a systematic way. Trust management is the process that does that, and it will be discussed further in this chapter.

[8] The text in this chapter was originally published in WM Maiden. *Trust Management Considerations for the Cooperative Infrastructure Defense Framework: Trust Relationships, Evidence, and Decisions*, Pacific Northwest National Laboratory Technical Report PNNL-19117, Pacific Northwest National Laboratory, Richland, Washington, 2010. Available at http://www.pnl.gov/main/publications/external/ technical_reports/PNNL-19117.pdf.

## 3.1 Motivation for Trust Management

The increasing prevalence of cross-domain distributed systems has driven the need for integrity mechanisms beyond the traditional mechanisms of authentication and access control lists. For example, in e-commerce, customers need to be able to minimize the risk inherent in dealing with service providers with which they have no experience. Service providers benefit when prospective new customers have a means to trust them to use their credit card and other personal information in a trustworthy manner, to deliver the order in exchange for the payment, and to deliver it in a timely fashion and in good condition [30].

Businesses need to be able to codify their security policies, test them for internally conflicting statements, and then enforce them by building these policies into their business process applications [6]. This need is magnified when businesses engage in limited cooperative relationships with other businesses that are at the same time competitors. However, traditional authentication and access control lists that are effective in client/server systems and in-house distributed systems cannot sufficiently express security policies related to finer-grained authorizations than the typical read/write/execute permissions, and their weaknesses are especially apparent in cross-domain distributed systems. In such systems, the principals (whether people or systems) often do not know each other, and there may be no mutually trusted third party. Even a mutually trusted certificate merely proves identity rather than the subject’s trustworthiness.

Researchers have responded to these needs with a variety of solutions, generally referred to as trust management systems, for managing the trust issues inherent in distributed systems.

## 3.2 Definition of Trust Management

The term “trust management” was coined by Matt Blaze of AT&T Research Labs, in 1996 [6]. Blaze’s concept of trust management centered on specifying security policies and applying them to authorization statements embedded in credentials to enable a trust management engine to directly assess whether a requested action should be allowed. Blaze’s use of the term reflects trust based on policy enforcement. More recently Tyrone Grandison [18] defined trust more generally to include both trust based on policy enforcement and trust based on a good reputation. He defines trust management as “the activity of collecting, encoding, analyzing and presenting evidence relating to competence, honesty, security or dependability with the purpose of making assessments and decisions regarding trust relationships” [18].

Trust management systems are generally based on reputation evidence [29] [48], credential-based evidence supporting policy enforcement [6] [39] or both [18] [11]. Trust management is most commonly used in distributed systems where there is no single authority, such as an employer or civil authority, that can control and levy punishment for inappropriate actions, and for which blind trust is not a viable option (e.g., because of the probability of a trust violation coupled with the associated risks).

Reputation-based trust management mechanisms are common in e-commerce [19] where a consumer needs to minimize the risk involved in using a service (e.g., purchasing a product) from an unfamiliar provider. Reputation is also useful in electronic communities, such as peer-to-peer [48] or wireless sensor networks [21] to detect nodes that are not being good citizens of the community, such as entities that freely consume resources without offering any resources in return. The risk to be avoided may be maliciousness or simply poor QoS. Trust management was originally conceived as a method for establishing trust across security enclaves where authentication is either impossible or meaningless, but reputation-based trust is also useful for maintaining trust within a security enclave when insider threat, intruders, or deteriorating QoS is a concern.

Reputation-based trust management works best when the principals have:

- Defined interactions with other principals over a period of time,
- Measureable elements of satisfactory and unsatisfactory behavior,
- A large community of peers to contribute trust observations, and
- A large number of transactions with peers.

Credential-based trust management systems typically have to do with authorization or delegation. In credential-based trust management systems, formally-specified policy statements are used by service providers or resource owners in conjunction with authentication certificates and/or authorization credentials to determine if consumers have the right to use a service or resource. This process may occur through a trust negotiation process wherein each entity iteratively reveals either policy statements or credentials to the other until trust has been established. Although authentication certificates, such as X.509 certificates, are useful within a security enclave to prove identity, attribute credentials and peer-granted identity certificates (e.g. PGP [36]) can be used to control access to resources in distributed systems that cross security enclaves where there are no shared authentication certificates controlled by a central authority. Even when a central authority spans enclaves, authentication merely serves to prove identity, not trustworthiness.

Credentials also provide much more expressiveness than the typical read/write/execute access control permissions. For example, an authorization credential might represent that "the holder is authorized to sign contracts worth up to $200,000" or a policy may require that only university students are eligible to sign up for a benefit and therefore a "student" attribute credential signed by a known university must be supplied.

Credential-based trust management works best when the principals have:

- Well-defined activities and interactions
- A shared trust anchor

Both forms of trust management have the potential to provide value to swarm-based autonomic computing systems.

## 3.3 Trust Context

Trust is context-specific; the trust that entity A (the trustor) has in entity B (the trustee) will vary depending on the specific context. The types of context that can occur in a distributed system can be classified as follows:

- **The trustor must decide whether to grant a trustee access to a resource.** In this context there may be a policy decision point (PDP) that considers the trustee’s trust evidence. The trust evidence may include its own authentication or authorization credentials, credentials delegated by others, and/or its reputation as known by the trustor and other entities.
- **The trustor wants to select a trustee that will provide a quality service.** In this context, if the trustor has several service providers to choose from, the selection may be based on each service’s quality of service (QoS) to other trustors as well as any previous direct experience the trustor has in using the trustee’s service.
- **Infrastructure trust** pertains to the foundational trust that an entity must be able to have in the hardware, operating system components, and networks that form the infrastructure upon which the trust relationship takes place.

### 3.4 Foundations for Trust

Wilhelm *et al* identified four foundations for trust [45]. Although their paper was specific to trust in mobile agents, the four foundations for trust are broadly applicable:

- **Blind trust**, such as the trust a child has for a parent.
- **Trust based on control and punishment**, such as the threat of loss of employment, fines, or jail time. This is typically applicable in situations where

there is a controlling entity such as a government or employer. It is more applicable to trust in humans than trust in software entities.

- **Trust based on a good reputation**. This type of trust assumes that because a good reputation is hard to build and easily destroyed, the entity will not want to do anything to harm its reputation. One or more trust *evidences* are used to form the basis of an entity's reputation.
- **Trust based on policy enforcement**. This is trust in the policy and the enforcement mechanism rather than in the entity itself. It includes the enforcement provided by standard security mechanisms such as X.509 certificates.

### 3.5 Trust Concerns in Swarm-Based ACSs

Although CID's hierarchical interactions occur within a single security enclave, its agents should not be blindly trusted, nor should authentication and role-based access be considered sufficient authorization. Because it is a security system, its agents could be targets of attack by malicious entities from inside and outside of the security enclave. The Sensors' mobility increases their exposure to potential malicious forces, as does the fact that they must not avoid going to hosts that are exhibiting problems. Sentinels are at risk if the host on which they reside is attacked by a malicious entity. Trust becomes especially complex when a Sentinel is compromised and begins to act maliciously. Trust management can be used to alleviate these potential weaknesses. Addressing these challenges will be the focus of the remainder of this thesis.

# CHAPTER FOUR

# SURVEY OF LIGHTWEIGHT TRUST MANAGEMENT FRAMEWORKS

This chapter analyzes the characteristics of wireless sensor networks and mobile ad hoc networks (MANETs) and the types of attacks to which they are vulnerable, and describes examples of trust management architectures that have been designed with these characteristics and vulnerabilities in mind. Most of these systems have been designed to support packet routing which can be seen as a parallel concept to the ad hoc routing of Sensor agents in CID. Because these types of systems are resource constrained, the trust management systems designed for them are necessarily lightweight.

## 4.1 Wireless Sensor Networks

### Characteristics

Wireless sensor networks (WSNs) are ad hoc networks of wirelessly-connected nodes equipped with one or more sensors, a radio transmitter or other wireless communications mechanism, a microcontroller, limited memory, and a power supply (e.g., a battery or solar power). WSNs are used for a broad range of monitoring activities including environmental, industrial, healthcare, logistics and inventory management, security, and military surveillance.

WSNs are ad-hoc, in that they lack the fixed routing infrastructure of managed networks. Instead, each node acts as a router and cooperates in a trusted manner with its neighbors to implement a multi-hop routing protocol for sending data to a base station. The operations of a WSN node also include data aggregation and time synchronization [17]. Base station(s) are responsible for final aggregation of data and for overall monitoring and control of the network. Although WSNs typically employ a flat topology, a hierarchical or clustered organization ([2], [47]) can be used to reduce communication overhead, aggregate data on the way to the base station, and improve the scalability of oversight operations; however, this imposes structure on the otherwise ad hoc network.

WSNs can vary in size by orders of magnitude, vary in the homogeneity of the nodes, and vary in the degree to which they are truly ad hoc. “Traditional embedded wireless networks” [14] may only have ten or twenty nodes. The nodes may have different, but complementary roles and be connected by a network that is perhaps more managed or structured than ad hoc. These types of networks can be found in healthcare, manufacturing, and security settings.

On the other end of the scale, Eschenauer and Gligor [14] refer to WSNs that employ massive numbers (perhaps tens of thousands) of uniform sensor nodes communicating over an ad hoc network as distributed sensor networks (DSNs). DSNs are dynamic because nodes can be added (to grow the network or replace failing nodes) or removed as needed with little or no pre-configuration.

Due to their limited resources and unattended operation, often in harsh environments, WSNs require algorithms and protocols that minimize computational and communication requirements to maximize battery life and are robust, fault tolerant, and self-configuring.

### Attacks against WSNs

Pirzada *et al* [37] identify five types of attacks against wireless ad hoc networks, including WSNs and MANETs:

- Passive attacks are eavesdropping attacks in which the attacker reads packets during transmission and seeks to learn which nodes have more value within the topology (e.g., cluster head nodes).
- Active attacks are intended to disrupt. Ad hoc routing protocols assume that nodes won’t tamper with the protocol fields, which leaves them vulnerable to traffic subversion and denial of service attacks.
- Modification attacks are frequently targeted against the integrity of routing computations. Black holes, grey holes, wormholes, sinkholes, and loops that can cause partitioning are examples of modification attacks where the attacker modifies routing information
- Fabrication attacks generate false routing messages or routing error messages incorrectly leading the sender to believe that a neighbor can no longer be contacted.
- Impersonation attacks are those in which the MAC or IP address of a node is altered in order to masquerade as another node, tricking the node into sending it data packets.

One of the most significant avenues of attack against WSNs is through active attacks, modification attacks, and fabrication attacks against their routing protocol. Depending on the choice of routing protocol and topology used by the WSN [28], several types of routing protocol attacks are possible. HELLO flooding is an attack that floods neighboring nodes with the HELLO messages that some protocols require to announce a node to its neighbors. Flooding prevents the node from responding to legitimate routing requests and is therefore a form of denial of service. Sinkhole attacks lure traffic through a compromised node by making it look attractive resulting in data leakage and data manipulation via selective forwarding. Wormhole attacks involve the tunneling of packets over a low-latency connection to another part of the network where they can be replayed without detection. Routing protocol attacks can also be used to partition parts of the network from the rest of the network.

If an attacker has access to the WSN vicinity, additional attacks become possible. Node(s) could be added to falsify data or to consume limited network bandwidth resources. If the attacker has a wireless laptop with greater transmission power than the WSN nodes and if the protocol uses geographically-based routing, the attacker could use the laptop’s greater transmission power to convince other nodes that are in range that the laptop is their neighbor [47]. If a node is captured, keys stored on the node will be compromised potentially leading to active manipulation of sensor data and sleep deprivation attacks [14]. In the latter, the adversary broadcasts communication just frequently enough to keep the node from being able to go into a low-power “sleep” mode. The wireless communications are also subject to eavesdropping.

Although it may not be possible to prevent a node from being captured, physical tamper-detection technologies exist that can minimize the resulting threat. For example, if tampering is detected, the sensor can be disabled or the key ring erased [14].

### Uses and Issues of Cryptographic Keys in WSNs

A single mission key can identify a node as a member of the WSN and keep communication private. However, if the key is compromised, perhaps because a node was captured, then all of the nodes are compromised. Mission keys, by themselves, are also not sufficient to support trust evaluation since trust-evaluating nodes must be able to uniquely and accurately identify the nodes whose trust they are evaluating. Node-specific keys are needed. Unfortunately, the overhead associated with asymmetric keys in comparison to symmetric keys (approximately two to four orders of magnitude according to Carman, Kruus, and Matt [9]), makes them impractical for most uses in resource-limited WSNs. However, even pair-wise symmetrical keys pose challenges for WSNs, particularly in regard to scalability in the presence of limited memory. Once deployed, nodes can only communicate with the subset of nodes that are within communication range. Unfortunately, because the network topology is not known prior to deployment, it is impossible to know which nodes will be within a given node's neighborhood when deployed, so every node would be required to store the n-1 keys. This limits the size of the WSN to the amount of storage space that can be allocated to storing keys. Adding a node to the WSN after deployment, revoking the key of a captured node, or re-keying would require extensive (and expensive) communication. [14]

Key distribution is yet another challenge since network topology is unknown prior to deployment and nodes can only communicate to other nodes within its wireless communication range. Sensor operation may also be intermittent. For these reasons key pre-distribution is a common mechanism for key distribution in WSNs (e.g., [14]).

Eschenauer *et al*. [14] proposes a three-phase solution. First, during the key pre-distribution phase, a large pool of keys is generated, from which a ring of k keys is randomly selected (without removal from the pool). The IDs of each sensor and the keys on its key ring are stored on a trusted controller. The researchers showed that only a small number of keys are needed in each sensor's key rings to ensure that any two nodes can communication either directly or indirectly. Second, the shared-key discovery phase occurs during DSN initialization in the operational environment. In this phase, nodes discover which of their neighbors have a shared key. Finally, during the path-key establishment phase, a path-key is assigned to pairs of nodes that do not share a key but are connected by two or more links. Later, if a sensor is compromised, a controller node can broadcast a revocation message containing a signed list of the key identifiers for the ring to be revoked. Each node removes any of the revoked keys that it has on its ring. If the revocation causes any links to break, the shared-key discovery and path-key establishment phases are repeated to re-establish secure communication. Re-keying, if needed, is equivalent to a node self-revoking an expired key on its ring.

### Evidence for Trust Establishment

Most security and trust research targets DSNs due to their potential for physically unsecured and unattended deployment in environmentally or militarily hostile environments. Sensors operating in this type of environment can be easily compromised. As [2] states, “It is critical to detect and isolate compromised nodes in order to avoid being misled by the falsified information injected by the adversary through compromised nodes.” Falsified information can include both fake data as well as real data that have been manipulated such as by selectively forwarding packets. The problem further extends to nodes that are simply malfunctioning. This section provides an overview of the evidence that researchers have used to determine trust in WSNs.

Behavior evaluation, as implemented in e-commerce for instance, collects feedback from the recipient of a service, but in a WSN, feedback would add undesirable additional traffic [21]. There are, however, many trustworthiness indicators that can be measured without feedback.

Anomaly detection within a neighborhood is a key method for determining the trustworthiness of sensor nodes. In DSNs, the values measured by neighboring sensors tend to be similar to each other, due to the placement density of the sensors and the comparatively slow rate at which the monitored attribute (e.g., temperature or airborne particulates) changes across the region. This characteristic enables the use of outlier detection to identify nodes that are apparently reporting false data, either maliciously or because they are performing poorly.

Anomalies may also be detected through neighborhood monitoring of localization information [12] or of signal strength compared to the node's geographical position [24]. To minimize consumption of resources, neighborhood values may be only randomly sampled.

In [47], the authors propose monitoring a node's energy consumption (large increases and/or extending beyond a min/max range), working hours, and location, in addition to neighborhood monitoring for data anomalies.

Fernandez-Gago, *et al* [17] note that there are events that occur at multiple ISO layers that can be used as trust indicators. The more of these that can be used, the better the reputation score will be, but, again, the cost of trust must be weighed against the risks and benefits in this resource-constrained environment. At the hardware level, a node that appears and disappears from the network under normal conditions or that does not respond to pings to determine if it is alive should not be trusted. At the communication (network) layer, a node may be considered untrustworthy if it is alarming when its physical surroundings are calm, replying to non-existent queries, or creating packets outside of the expected timeframe when periodic sensor readings are normally forwarded to the base station. Because node communications are broadcast, selective forwarding and packet delaying can also be detected and used as trustworthiness indicators. Exchanging false or delayed data is another reason for mistrusting a certain node. The authors consider a node that is intermittently uncooperative to be completely untrusted.

### Trust Management Frameworks Proposed for WSNs

The resource constraints of a WSN make reputation-based trust management difficult to implement in a manner that provides sufficient benefit compared to the security and performance costs. Perhaps as a result, research in this area is not as well-developed ([17], [21]) as WSN authentication techniques. The reputation frameworks that have been developed align with the structure of the WSN itself and generally address one or both of the most common forms of attack – routing protocol attacks and data falsification or tampering. For flat ad hoc WSNs, reputation frameworks may use a fully distributed trust model, fitting naturally with the peer-to-peer style interactions of the WSN. Since most interaction in a WSN is with neighboring nodes, a node can track the behavior of the nodes with which it interacts and can also periodically exchange opinions with its neighbors.

For hierarchical or clustered WSNs, a hierarchical approach to reputation management is used, reducing the communications impact and providing a natural location at which to place the trust evaluation role, while removing this responsibility from lower-powered sensor nodes. Atakli, *et al* [2] specifies a trust framework for a three-tiered, hierarchical WSN network with sensor nodes at the bottom, the base station at the top, and a layer of higher-powered forwarding nodes in between. The forwarding nodes and base station are trusted. Values are weighted based on the node’s perceived trustworthiness over time, which is determined by the similarity of the node’s data values to the data values received from neighboring nodes. If a node's weight factor falls below a specified threshold, it is determined to be a malicious node.

Xu, *et al* [47] recommends a clustered approach that is essentially a three-tier hierarchy. The hierarchy consists of (top to bottom): the command node, multiple cluster head nodes, and, for each cluster head node, a set of common sensor nodes. The command node is trusted. The cluster head node's trust is checked by the command node above it and the sensor nodes below it. The sensor nodes' trust is checked by their cluster head node. Trust calculations are based on context changes (energy, geographic information, and operating hours) and data changes. The calculations are run initially to determine a set of trusted nodes, and are then re-run at a set interval. Untrusted common sensor nodes are logically removed from the WSN. Similarly, the command node compares current and historical context data (i.e., energy, number of common sensor nodes, and location) from cluster head nodes to context data from neighboring cluster head nodes. As before, untrusted cluster head nodes are logically removed from the WSN.

Others use fault tolerant methods to filter out bad data without determining the source. For example, several nodes may provide aggregated data to the base station and the base station may use an *m* out of *n* scheme to eliminate bad data [21].

## 4.2 Mobile Ad-Hoc Networks (MANETs)

### Characteristics

MANETs share many characteristics with WSNs. Both are ad hoc networks of wirelessly-connected nodes that act as routers. WSNs and MANETs are self-configuring and their nodes typically operate on batteries. The nodes may be physically vulnerable and may be located in hostile territory. Both use multi-hop routing to transfer packets and therefore require discovery of secure routing paths.

MANETs differ from WSNs in that the purpose of the network is communication rather than collection of sensor data, so end-to-end communication may be between any two nodes rather than from outlying sensor nodes back to a base station. MANET nodes may have more computational capability and memory than WSNs, but battery life is still limited. MANETs typically have a much lower node count than WSNs. WSNs may have a very long lifetime on the network, while MANET nodes may freely join and leave the network due to their mobility, thereby offering their functionality to neighboring nodes for shorter periods of time. Whereas WSNs can discover their topology once deployed, MANETs must deal with a constantly changing topology due to their mobility. Minimal configuration and quick deployment make MANETs well-suited for communication in emergency situations, such as natural disasters, and on the battlefield.

### Attacks against MANETs

In general, the attacks that apply to WSNs also apply to MANETs since both are wireless ad hoc networks [37]. These include routing attacks (e.g., HELLO flooding, worm holes, and sink holes, selective forwarding, route modification, and packet dropping), fabrication of data, denial of service, spoofing, eavesdropping, and captured nodes.

### Cryptographic Keys in MANETs

MANETS have much in common with WSNs in regard to cryptography also. Authentication of nodes is necessary for the implementation of secure routing protocols. Since MANET nodes are mobile and can move around and out of the network at will, identifying them with a unique key becomes even more important. Because MANET nodes typically have more computational and memory resources than WSNs, the use of asymmetric keys is possible, but battery life is still a limiting factor. The relatively smaller size of MANETs makes key management for MANET nodes more feasible than for WSNs.

### Evidence for Trust Establishment

In the area of trust evidence, both similarities and differences exist between WSNs and MANETs. WSNs (and particularly, DSNs) are able to use anomaly detection as a significant source of trust evidence because the uniformity of the task and the placement density of the sensors permit the assumption that the collected values should be nearly the same. This does not work for MANETs for multiple reasons: they are not data collectors, they carry differing communication streams, and their nodes are not as densely placed. For WSN's the appearance and disappearance of a node from the network or a change in location is considered reason for suspicion; for MANET nodes, this is expected behavior.

There are, however, trust evidences that can be gathered to determine the trustworthiness of MANET nodes, and many of these apply to WSNs also. Most of these evidences arise from analyzing received, forwarded, and overheard packets to determine how nodes handle both the data and control packets for which they are responsible [37]. Evidences that can be monitored include frames received, data packets forwarded vs. dropped, control packets forwarded vs. dropped, data packets received, control packets received, streams established, tampering with data packets, control packets, or route replies, and unidirectional behavior by a bidirectional node [37]. Comparing the transmission rate of a neighboring node to a threshold rate is useful for identifying potential denial of service attacks [7].

For a credential-based trust system such as the one proposed by Eschenauer *et al.* [15], trust evidence may include an identity or attribute (such as location) or other information required by the policy. Trust evidence must be digitally signed by the originating node.

## Constraints and Criteria for MANET Trust Establishment

The multi-hop routing of ad hoc networks requires that they act cooperatively and trust their neighbors; however, trust is difficult to establish between MANET nodes because relationships originate, develop and quickly expire due to mobility [37]. This mobility also precludes dependence upon centralized trust authorities and hierarchies of trust relationships between nodes since node reachability is not guaranteed. Due to these constraints, trust management frameworks for MANETs tend to use peer-to-peer trust relationships [7], [15], [37].

The trust establishment process must be fast since the communication path between the two nodes may not be available for very long. A slow trust establishment process could prevent secure communications [15].

Another constraint, particularly for credential-based, peer-oriented trust systems is that it cannot assume that all established evidence (credentials) will be available when needed since a node may be out of wireless range. Trust establishment must be able to be function with incomplete trust evidence [15].

## Trust Management Frameworks Proposed for MANETs

### *TEAM: Trust-Enhanced Security Architecture*

In [7], Balakrishnan *et al.* present TEAM, a Trust Enhanced Security Architecture for MANET, which is a unified architecture that integrates trust with key management, secure routing, and inter-node cooperation. The components of TEAM are a cooperation model, a secure routing protocol, a key management mechanism, a basic routing protocol, and a peer-to-peer trust model known as SMRTI, Secure MANET Routing with Trust Intrigue. SMRTI asynchronously captures trust evidence gathered by the various TEAM components during their operations, and in return, the components can synchronously request information by which to make better security decisions. In this way, components that contribute to trust evaluation also benefit from it.

TEAM defends against free-riding, honest-elicitation, flooding, and packet drops, and can identify and isolate malicious and selfish nodes that fail to share the communication channel or forward packets for other nodes. Nodes running TEAM check the trust of the sending node before accepting a packet, check the trust of the next node before sending a packet, check the trustworthiness of the packet itself (based on the nodes in its route) before transmitting the packet, and check the trust of the nodes in a route before using the route. The key management system also uses trust input to determine whether to revoke a node's keys.

When a source node creates a packet, it must determine a safe route to the destination node. The node maintains a cache of trusted routes (i.e., trusted as of the time they were saved). The trustworthiness of the cached routes to the desired destination is re-checked first, and the route with the highest trust is the route that is encoded into the packet. If none exist or none are satisfactory, the route discovery algorithm is initiated. Intermediate nodes in the route gather trust evidence and evaluate trustworthiness three times -- for the previous node (which indicates whether the packet was exposed to malicious behavior), for the packet (as indicated by the authenticity and trustworthiness of the route), and for the next node (to ensure that the packet will safely reach its destination). The authors suggest that these trustworthiness checks be relaxed when the nodes are not in a malicious environment.

Some MANET trust systems disseminate trust recommendations, but this increases overhead, degrades performance, and makes the system susceptible to malicious behavior by the recommender such as honest elicitation and free-riding behavior. Instead, TEAM's trust module (SMRTI) derives the recommendation using the route contained in a received packet. Essentially, the recommended trust is derived by assuming that each node in the route thus far indicates that the node prior to it trusted it. The reputation score is negated if the current node does not trust the previous node. Scaling is also applied to indicate that the recommendation for a node further back in the route is proportional to the trustworthiness of subsequent nodes in the route.

### *Pirzada and McDonald*

In 2003, Pirzada and McDonald [37] surveyed secure routing protocols and found that all secure routing protocols at that time depended on a central trust authority. The authors classify this type of environment which must be pre-configured as a "managed ad-hoc network" as compared to a "pure ad-hoc network" which is true to the original intended improvisational nature of ad hoc networks.

Pirzada and McDonald [37] note that an ad hoc network trust model must be designed for the routing protocol. In some routing protocols, the source calculates the route for a node and stores it in the node before transmitting the node to the next node in the list, whereas other routing protocols only determine the next hop. Their trust model, described in [37], is designed for the Dynamic Source Routing protocol developed by Johnson, Maltz, and Hu [26] and does not superimpose managed constructs such as central trust authorities or cryptographic keys on the ad hoc network. Their model equates node reliability to trustworthiness, which the authors acknowledge does not make the routing protocols “secure” in the strict sense of the word.

To collect evidence data, the authors recommend “passive acknowledgement” of successful packet transmission wherein the sending node, after transmitting a packet, places itself in promiscuous mode in order to observe how the receiving node handles the packet that was forwarded to it. This mode allows the sending node to determine if the next node is dumping packets, delaying forwarding packets, or modifying packet contents. Alternative methods for verifying successful transmission include link-layer acknowledgements by the underlying MAC protocol and network layer acknowledgements specifically requested by the sender, but these don’t provide the additional insights that the passive mode provides.

The authors propose a distributed trust model wherein each node has a trust agent that maintains its own trust database. Each agent has three functions: trust derivation (gathering trust evidence), computation, and quantification. Trust derivation is performed at the lower levels of the OSI reference model using passive acknowledgement of successful packet transmission as described previously. Quantification and computation occur in the upper levels of the OSI reference model. Trust information is categorized according to specific node functions. Trust quantification translates the trust data for a given node function into a reputation score in the range of -1 (not trusted) to +1 (completely trusted). Trust computation applies a weight to the reputation score for each type of function based on its importance to the action for which trust is being determined. The computed trust levels are assigned as weights to the potential links, so the sending node can use a shortest path algorithm to determine the most trustworthy path.

### *Eschenauer et al.*

Eschenauer, Gligor, and Baras [15] utilize credential- and policy-based trust for authentication and provide scenarios for its use in multi-national military MANETs. As noted previously, trust evidence in this paradigm may include a signed identity or attribute credential or other signed information as required to satisfy policy. Trust evidence can be collected by any node about any other node and must be signed by the originating node. The effective lifetime for the trust evidence is also recorded.

Since a particular node may not always be reachable by another MANET node, the trust system cannot be dependent upon a centralized trust database or certificate authority. Therefore, storage and processing of trust evidence occurs in the nodes of the ad-hoc network. Interestingly, the scenarios provided by the authors include a centralized certificate authority (CA); however, the trust system is still able to function when the CA is out of range because certificates are cached in other nodes and policies can be written to provide alternatives when the CA is not available.

The authors compare the distribution of trust evidence to distributed data storage systems, but rather than routing a query for a single piece of information to the closest source, the distribution of trust evidence requires that all related information of the requested type about a principal needs to be returned from all available sources. They suggest considering a peer-to-peer networking service such as Freenet which sends out requests using optimized routing rather than flooding. However, Freenet would need to be extended to support the return of all related pieces of information. Swarm intelligence is also briefly mentioned as a possible mechanism for evidence distribution.

When evidence is received in response to a request, a confidence metric is applied to each item of evidence. The outcome of this metric is a confidence-rated trust relation that is stored locally by the node. If a set of credentials (i.e., signed evidence and trust relations) meets policy requirements, a policy decision can be made. In this type of system, it is not uncommon to encounter transitive trust. Trust is transitive if the following statement is true: “If A trusts B and B trusts C, then A trusts C.” Since trust is typically considered to not be transitive, Eschenauer *et al.* discuss the conditions under which trust can be safely considered transitive. Specifically, if the following two conditions hold, then transitivity holds. The first condition is that B’s policy and mechanisms for determining trust in C must be at least as strong as A’s policy and mechanisms for determining trust in B. This may be determined off-line and cached as trust relations signed by the certificate authority. The second condition is that B’s relationship with C must be at least as stable and long-term as A’s relationship with B.

In the following two scenarios provided by the authors, UK refers to the United Kingdom and UKCA is their military’s certificate authority. Similarly USCA is the United States military certificate authority.

Military scenario 1: A UK unit has lost contact with UK command and needs to call for help. A US unit is nearby. The UK unit presents to the US unit an identity certificate signed by UKCA. The US unit gets the UKCA certificate signed by USCA from USCA. It can then accept the UKCA signature on the UK unit's certificate and exercise the transitive trust relations established between the UK and US to allow the UK unit access to the US ad-hoc network.

Military scenario 2 is the same as military scenario 1 with the exception that the satellite link to USCA is down: The UK unit (UK1) presents to the US unit (US1) an identity certificate signed by UKCA, but US1 is unable to check the certificate since the satellite link to the USCA is down. Helicopter unit US3 recently spotted UK1 and could generate a certificate for UK1 and make it available in the US network that US1 could then use.

## 4.3 Applications of Existing Research

The trust management systems reviewed in this chapter were designed for resource-constrained systems and are primarily used as part of the routing protocol to determine safe nodes to use in the routing. The goal in CID is much the same: to use trust to protect the Sensors (rather than packets) as they adaptively choose their own route from Sentinel-to-Sentinel (node-to-node) throughout the system. Concepts such as using anomaly detection rather than peer feedback are of potential use in swarm-based ACS's. Random sampling to reduce the performance impact of trust management, the blending of hierarchical and distributed trust models, and the use of forward and reverse reputation checking are all of interest but will need to be tailored to the ACS environment.

# CHAPTER FIVE

# SECURING THE MOBILE AGENT SWARM

This chapter[9] surveys the threats that can occur in mobile agent systems, briefly references the traditional security-based countermeasures that can be implemented, and discusses why these countermeasures are insufficient for swarm-based autonomic computing systems.

## 5.1 Characteristics of Mobile Agent Systems

Mobile agent systems in the research literature tend to have several characteristics in common. The agents have a known agent-owner/creator that sends them on a mission, often across security domains, to a preset itinerary of hosts, from which they are expected to return and report their findings or accomplishments. Agents interact with other agents, sometimes extensively, to accomplish their mission. Reputation management is usually done for the purpose of removing an untrusted host from the agent's itinerary to protect the agent.

[9] This chapter appeared in Maiden WM, JN Haack, GA Fink, AD McKinnon, and EW Fulp. 2009. "Trust Management in Swarm-Based Autonomic Computing Systems." In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE Computer Society, Brisbane, Australia. © Copyright 2009 IEEE. Reprinted with permission.

## 5.2 Mobile Agent Threats and Their Countermeasures

Mobile-agent-based systems face the same threats as other types of distributed systems, and they may likewise use many of the traditional countermeasures. However, mobile-agent-based systems face additional threats that may be characterized as agent-to-platform, other-to-platform, agent-to-agent, and platform-to-agent threats [25]. This chapter will discuss each of these threats in turn.

The following design elements mitigate the agent-to-platform and other-to-platform threats from CID Sensor agents and others:

- Sensors carry a digitally-signed hash of their classifier code, enabling the host to verify the code’s integrity before running it.
- CID uses a port-hopping transport algorithm that does not require keeping any static ports open. Ports remain open only long enough to transport the incoming Sensor.
- Sensors are always sandboxed so they may not store data on the host and they may only read where they are permitted by policy. For example, to prevent writing to the host, when pheromone is to be left on a host, the Sensor makes the request through the Sentinel.
- Sensors may not communicate with external machines except to move to them, and the Sentinel will only allow them to move to neighboring hosts.

Agent-to-agent threats are unlikely in CID because the ant-like Sensors interact only by means of leaving a digital pheromone trail. It would be necessary to corrupt a large number of Sensors to affect the Sensor community's overall results.

The remaining mobile agent threat class – platform-to-agent – is that of malicious threats by the platform (i.e., host) toward the swarming agents. In CID each Sentinel is responsible for monitoring its host, thereby establishing trust for the hosts in the enclave. This shifts the concern to the trustworthiness of the Sentinel, the autonomic manager of the host, to mitigate platform threats against the CID swarm. Sentinels are vulnerable to corruption because they reside on the hosts they monitor. An untrustworthy Sentinel may be capable of threatening a significantly large number of agents, thereby impacting the actions of the collective swarm.

## 5.3 Challenges in Applying Trust to Protect Mobile Agent Swarms

One of the benefits of mobile agent systems is that processing is distributed and only the final results are returned, whereas stationary systems generally require a much greater network bandwidth to accomplish the same task. However, one of the consequences of mobile agent systems is the increased vulnerability of their code. Trust management can help reduce the risk associated with the increased vulnerability, but if it is too network-intensive, adds too much processing overhead, or adds rigidity preventing the system from adjusting to a dynamic environment, the cost of trust will outweigh the benefits of mobile agent-based autonomic systems. A balance must be found.

In addition to these architectural constraints, CID Sensors have a number of unique characteristics that require a very different trust management approach from what has been described in the research literature to date. The typical research scenario is an agent-owner host that sends the agent to a list of hosts on the open network, as defined in an itinerary, to accomplish a computation and return with the answer. An alternative (and sometimes overlapping) scenario involves trust amongst collaborating heavyweight agents.

CID has characteristics that differ considerably from these typical research scenarios. Other swarm-based autonomic systems are likely to have some combination of the characteristics described here. Some of these characteristics simplify trust management, but most make it more challenging.

### No pre-set itinerary

Mobile agents usually have a pre-determined itinerary. Therefore, trust management systems have been designed to use reputation to detect and avoid malicious hosts by removing those hosts from a mobile agent's itinerary. In comparison, in CID, the Sergeant defines a two-dimensional Geography for all Sensors in the enclave at once and distributes it to the Sentinels, with occasional updates as needed. When the Sensor is ready to move to a new host, it selects its direction by randomly perturbing its current heading on the Geography. The Sentinel determines the nearest neighbor in this direction and transfers the Sensor to this host.

### Open-ended mission

Mobile agents in research literature usually return to the agent-owner and terminate once they have finished the computation they were sent to accomplish across the specified itinerary. In comparison, CID Sensors continue their wanderings until the Sensor dies from lack of energy or is terminated by a Sentinel for misbehavior.

### Malicious hosts must not be avoided

Trust management has been used to identify potentially untrustworthy nodes and to modify the agent's itinerary to avoid such nodes. In contrast, CID Sensors are required to go to potentially untrustworthy nodes to perform their mission of detecting and characterizing problem nodes. A topic of future research is whether the risk involved in allowing a Sensor to continue to roam after being on a host that is deemed untrustworthy can be adequately contained to allow the Sensor to continue to other hosts leaving a pheromone trail. We allow the Sensor to continue its travels so long as the suspect host's Sentinel is still considered trustworthy. Alternatively, we could allow the Sensor to leave pheromone but not allow it to "forage" (have its classifier run and receive reward in return for its findings), leading eventually to Sensor termination.

### Dynamic sensor community

Although a Sensor's lifespan is indefinite, it is expected to be relatively short. Sensors that do not perform (i.e., find problems on hosts) will not be rewarded and will terminate quickly. Characteristics of the most successful Sensors are used to generate new Sensors in a manner analogous to genetic programming. This adaptive selection is essentially a form of reputation-based trust management that is inherent to CID. The result is a population of mobile agents with the logic that best serves the enclave.

### Minimal communication between agents

Although in most agent paradigms, mobile agents communicate directly with each other to complete a task; CID Sensors and other ant-inspired swarms communicate only through marking their environment with digital "pheromone"—stigmergy. Therefore, trust management research for highly interactive mobile agent communities applies minimally to ant-inspired swarms.

### Minimal sensitive data

Mobile agents often carry sensitive data, but CID Sensors only carry a small amount of aggregate information that should not be sensitive and cannot easily be traced back to the source systems. Therefore, mobile agent data confidentiality is not a major concern in CID. Theoretically, a malicious entity could acquire information from enough Sensors on what each considers normal to design its activities to stay beneath CID's detection threshold. Investigation of this possibility is a matter for future research.

### Sensors don't have a clear agent-owner

In most mobile-agent scenarios, the owner of the agent bears permanent responsibility for the agent's activities. But in swarms of ephemeral agents where the agents are autonomous, such responsibility is not a particularly useful deterrent to malicious behavior. Sentinels may create Sensors, but the creator bears no particular responsibility for the newly created Sensor. Although the Sentinel is the next higher level in the CID hierarchy, there is no owner relationship because the Sentinel they report to varies according to which host the Sensor visits. Ultimately, the human Supervisor bears responsibility, but there is no direct path of control between Supervisors and Sensors, and creating one would result in a communication bottleneck and loss of the advantages of stigmergic collaboration.

## 5.4 Analysis of Existing Reputation Management Techniques for Securing Mobile Agent Systems

Trust management literature with direct application to swarm-based autonomic computing systems could not be found. Therefore this chapter analyzes the suitability of trust management frameworks in research literature whose architectural assumptions and design constraints were closest to those of CID’s swarms. While not directly applicable, the papers reviewed in this section offer useful approaches.

### Integrated security and trust management for mobile agents

In the MobileTrust architecture, Lin *et al*. [31] [32] and Lin and Varadharajan [33] integrate security and reputation to provide additional protection for both the agent and the executing host in a mobile agent system where agents traverse the open network. The authors note that the additional protection is required because identity and intentions are not as well known as they are within a security domain. While this is true, this thesis asserts that because domain perimeters are somewhat porous and because of the threat of malicious insiders, there is a need for trust-enhanced mobile agent security even within a security domain. This is especially true of an autonomic computing system designed to protect the security of the domain [10].

To protect the agent, the MobileTrust agent-owner checks the reputation of each host in the agent's prospective itinerary and removes hosts with a poor reputation. Then, to protect the host, the MobileTrust agent-host authenticates the agent, checks the agent's authorization, and checks the reputation of the agent-owner and each agent-host in the itinerary that has been visited so far before executing the agent.

MobileTrust cannot be applied directly to the CID system because the Sensor agents have no pre-set itinerary and the agent-creator is not the owner. Nor is there any on-going relationship between the Sensor and the agent-creator. However, the recommended forward and reverse reputation checking relative to the Sensor's path is an applicable concept for swarm-based autonomic computing systems. Trust between CID Sentinels can be achieved by adapting the recommended forward and reverse reputation checking relative to a Sensor's path. Stating it precisely, the $i^{th}$ Sentinel in Sensor *j*'s path checks the reputation of the $i+1$ Sentinel before passing a Sensor to it unless it has just checked it within delta time *e*. Symmetrically, Sentinel $i+1$ checks the reputation of Sentinel *i* prior to running Sensor *j* unless it has checked this reputation within delta time *e*. The reputation of the host itself has no bearing on whether it should be selected as part of a Sensor's path because Sensors are required to visit troubled hosts to help characterize the problem. Instead, the reputation of the Sentinel, the autonomic manager, must be monitored. A maliciously altered Sentinel could cause persistent, serious problems for the CID framework.

The authors also specified some details about the standard security techniques it uses: Each type of entity has its own key pair. A hash function and digital signature are used to make the agent tamper-proof. The agent and any interim data results are protected for confidentiality and integrity during transmission. Logging with digital signatures is used for non-repudiation. The receiving host computes and compares the hash function to ensure that the agent has not been tampered.

Agents are sent out with a "passport" containing the agent's certificate, the identity and privileges of the owner, and other information needed for authorization by the hosts on its itinerary. The agent platform on the receiving host maintains policies, commonly-used certificates, public/private keys, and a name server. Agent authorization results from a hierarchical certificate chain from a Security Management Authority (similar to the Sergeant) to the Security Manager (similar to the Sentinel) to the agent.

### Comparison of mobile code execution traces

Tan and Moreau [44] propose a method for code security based on comparison of mobile code execution traces to detect whether the previous host has tampered with an agent. Although not the primary goal, the method also protects hosts from malicious agents (host security). Traces are sent to a trusted, certificate-issuing verification server that issues a capability certificate to the agent-host that created the trace certifying that the agent-host correctly executed the agent template. If the trace does not compare, the agent-host's capability certificate is revoked, and in the future agents will avoid this host, thereby enhancing code security. The verification server also produces an execution certificate that it sends to the next agent host. The receiving host will not execute the agent unless this certificate is received, thus enhancing host security.

Execution traces could be applied to CID, but the public key infrastructure required by Tan and Moreau would be too encumbering for CID's lightweight Sensor agents. Instead, if a Sentinel used an execution trace to determine that a Sensor was corrupt, the Sentinel would simply terminate the Sensor. Because each Sentinel vouches for the trustworthiness of Sensors it sends to other Sentinels, the receiving Sentinel would then lower the reputation of the sending Sentinel.

To cover the former case, a special kind of Sensor agent is needed that would check its own execution trace as it went from host to host looking for differences in its own execution traces. When a difference was found, the Sensor would alert the Sentinel and leave a pheromone trail as usual. This special probe Sensor would alert the neighboring Sentinels as well, raising suspicion and causing them to lower the reputation of the Sentinel where the difference was noted. Sentinels could use probe Sensors periodically or whenever they begin to suspect a neighboring Sentinel's trustworthiness. Although this should not be the only reputation evidence gathering mechanism, this represents a decentralized and scalable mechanism for gathering reputation evidence in autonomic systems that use swarming sensors.

## 5.5 Applications of Existing Research

Existing trust management frameworks for mobile-agent applications were designed for systems with substantially different characteristics than swarm-based systems. Nevertheless, some contain applicable concepts. The reputation evidence-gathering mechanism identified by Tan and Moreau [44] is unlikely to be sufficiently lightweight and scalable for use with autonomic swarms but the concept of indirect reputation evidence-gathering offers a means to minimize or remove this responsibility from the swarming agents. The concept of forward and reverse reputation checking as described by Tan and Moreau [44] and Lin *et al*. [31] can be used to verify the trustworthiness of the autonomic managers and has the benefit of protecting the mobile sensor agents as well.

# CHAPTER SIX

# SECURING THE AUTONOMIC MANAGERS: RELATED TRUST RESEARCH

The last chapter concluded that verifying the trustworthiness of the autonomic managers in a swarm-based ACS can be designed to protect the mobile sensor agents as well, and it is a more scalable approach. Since the autonomic managers interact as peers with each other, this chapter surveys characteristics of peer-to-peer (P2P) systems, threats that can occur in these systems, the role of trust as a countermeasure to these threats, and why existing trust research is insufficient for autonomic managers of swarm-based ACSs.

## 6.1 Characteristics of P2P Systems

Generally speaking, a peer-to-peer community (P2P) is one wherein each entity (peer) in the network provides and uses the same services. P2P communities can include consumers that provide and use others' input on their satisfaction with e-commerce transactions. They can also include a group of collaborating autonomous agents.

In a more traditional sense P2P communities refer to peers in a P2P overlay network such as Gnutella, Napster, or Chord where each peer provides a service to its peers and is a client of the same service provided by its peers. Most commonly, P2P overlay networks are used for file sharing. In both cases, these networks are characterized by the anonymity of the peers (i.e., the service providers), and the maliciousness and carelessness of a certain percentage of peers who take advantage of the lack of identity and accountability. For example, files downloaded from P2P services may contain viruses, worms, or Trojan horses. Anonymity also leads to free-loading wherein some peers take full advantage of the services of their peers without equally contributing services to the community.

Reputation-based trust management research is providing mechanisms by which peers can determine which of their peers are trustworthy, and thus manage the risk they face in using the services of peers they do not know. In some cases, the question is whether to trust the one peer that offers to provide the needed service; in most cases, the question is which of a set of peers that claim they can provide the needed service is mostly like to provide it in a trustworthy manner.

### 6.2 Trust Model Constraints

Although Napster used a centralized database, pure P2P communities have no central authority or trusted third party that can monitor trustworthiness, store trust data, or manage proof-of-identity. As a result P2P reputation frameworks use a distributed trust model. In terms of resources, the primary constraint on the design of reputation frameworks for P2P communities is network bandwidth and the resulting performance impact rather than CPU or storage.

### 6.3 Threats in P2P Environments

In a P2P community, peers are at risk of receiving information that has been tampered with, such as files with misinformation or with viruses or worms inserted. Peers are also subject to man-in-the-middle attacks wherein a malicious peer intercepts message traffic between a requestor and a provider in order to gain information from the provider in the guise of the requestor or to provide a malicious response to the requestor in the guise of the provider [48].

Reputation-based trust mechanisms can be designed to address these threats but also have threats of their own. Xiong and Liu [48] provide an extensive list of the threats that occur in P2P trust systems:

- Peers can provide dishonest feedback to manipulate the reputations of others.
- Groups of peers can collude to provide good feedback for each other or bad feedback about others for selfish or malicious purposes.
- If a reputation system does not consider the context of the interaction that is being rated, a malicious peer can provide good service in less significant contexts to build a good reputation and then act maliciously in a more significant context.
- After gaining a bad reputation, a peer may discard the pseudonym identity with which the bad reputation is associated and create a new identity to get a fresh start in an attempt to mislead others again.
- A peer may also create multiple identities for the purpose of providing feedback multiple times to build up another malicious entity or tear down a good entity's reputation.
- Peers may need incentives to provide trust feedback.
- Occasional dishonest feedback can be worse than consistent dishonest feedback. A peer may build up a long-term good reputation and then use it in a one-time attack to take malicious advantage. Similarly a peer may oscillate between

building a good reputation through numerous small transactions and taking advantage of the good reputation to act maliciously in a larger transaction.

### 6.4 Cryptographic Keys in P2P

It is widely recommended that each entity have a public/private key pair [8][41][48]. Although key pairs are usually tied to identity, anonymity is prized in P2P communities. However, the need for security services, particularly integrity and authentication still exist. Xiong and Liu [48] recommend using the public key or a digest of the public key as the peer ID. Selcuk *et al* [41] recommend that the public key serve as the peer's pseudonym ID. Cornelli *et al* [8] recommend using a hash of the public key as the servant (peer) ID.

For P2P systems within a security enclave, X.509 certificates can be used; however, for P2P systems that span enclaves and have no central authority, Pretty Good Privacy (PGP) certificates are widely used. "PGP combines the convenience of the Rivest-Shamir-Adleman (RSA) public key cryptosystem with the speed of conventional cryptography, message digests for digital signatures, data compression before encryption, good ergonomic design, and sophisticated key management. And PGP performs the public-key functions faster than most other software implementations" [36].

PGP [36] is a certificate system that does not use a certificate authority. Rather, peers sign certificates for peers that they know, creating a Web of Trust. When a peer signs another peer's certificate, it specifies its degree of trust in the peer – Complete Trust, Marginal Trust, or No Trust – and the level of validity – Valid, Marginally Valid, or Invalid. If a peer validates Alice's key on its key ring and assigns her Complete Trust, then any key that Alice signs is treated as Valid on the peer's key ring.

### 6.5 Evidence for Trust Establishment

Reputation in a P2P system is derived through (1) direct experience with the service provided by a peer, such as a successful download of a good file, or (2) recommendations from other peers based on their own direct experience with the peer's service. A large quantity of transactions increases confidence in the peer's trust rating.

### 6.6 Threat Countermeasures in Trust Evaluation

Xiong and Liu [48] present a series of trust calculations that are increasingly sophisticated in their accuracy and ability to counter threats:

- The simplest measure of satisfaction is the simple sum of the feedback received. With this type of rating system, it is easy to "game" the system. For example, a peer who is malicious in one out of four transactions, but has performed many transactions, will have a better reputation than a newer peer who has been completely honest, but has had fewer transactions.
- A better metric is the sum of the feedback received divided by the number of transactions.
- The trust metric can be further improved by considering the credibility of the feedback provider. Xiong and Liu suggest two methods of calculating credibility. The first method sets a peer's credibility equal to their reputation score. This

method is effective when the number of malicious peers is less than 50% and when collusion is not present. Their second method bases the credibility rating on feedback similarity between the feedback provider and the feedback requester about entities they have previously dealt with in common. The authors note that this method is quite effective against collusion and works in the presence of high percentages of malicious peers, but of course requires that the peers have both had transactions with some of the same peers in the past. In Selcuk *et al* [41], the credibility rating is upgraded when the recipient of the recommendation experiences service that is consistent with the recommendation (i.e., peer A recommended peer B and the file from peer B was good, or peer A did not recommend peer B and the file from peer B was bad), and downgraded otherwise.

- The importance of the transaction context should be considered so peers do not game the system by racking up a good reputation via less important transactions in order to take advantage of its peers in a large transaction [48]. Dionysiou [11] also suggests tracking trust separately for each type of transaction context. Xiong and Liu [48] consider the context as a weighting factor for the interaction in the overall trust calculation, but the resulting reputation score is a collective evaluation that is not specific to the particular type of interaction. Xiong and Liu multiply these factors (satisfaction, credibility, context, and a weighting factor) together.

- An optional weighted “adaptive community context factor” can be added. The latter can be used to include the trust that derives from a credential if the peer has one, and it can also be used to provide incentive or can act as a default reputation score if there are not yet enough interactions from which to calculate trust.

Xiong and Liu [48] further recommend that the calculation of the trust rating should consider that a peer’s trustworthiness can change over time. Therefore, only recent transactions should be counted, so the peer has incentive to continue to treat others well. This helps to counter peers who would otherwise game the system by building up a good reputation over time in order to eventually take advantage of peers.

To address the threat of peers that rack up a bad reputation under one ID and then drop that ID and create a new one, [16] suggests making it difficult or unprofitable to change online identities.

## 6.7 Trust Management Frameworks Proposed for P2P Systems

### eBay – Centralized Trust

Although eBay’s buyers and sellers are distributed, eBay uses a centralized trust model to track their experience with each other. Buyers and sellers rate each transaction as a +1, 0, or -1. The entity’s trust rating is the sum of their ratings over the last six months. So long as a peer has more positive than negative ratings, the peer’s positive reputation will continue to increase. When trust evaluation is based on a simple summation of ratings and nothing more, peers can game the system with good interactions on small transactions and malicious interactions on large transactions [48].

### Distributed Trust Model

Abdul-Rahman and Hailes [3] proposed one of the earliest distributed trust systems. In their model, trust is calculated for each type of interaction (i.e., activity) between a pair of peers, and direct trust is calculated separately from recommended trust. Direct trust and recommender trust are each measured on a scale of -1 (Distrust) to 4 (Complete Trust).

If a peer lacks direct trust data about a peer, it sends out a *Request for Recommendation* message to peers that it trusts. These peers either respond with their direct trust data or pass the request along to other peers that they trust, and so on. The authors propose that trust is transitive (A trusts C) when the following conditions are true:

- B recommends its trust in C to A explicitly
- A trusts B as a recommender; and
- A can judge B’s recommendation and decide how much it will trust C, irrespective of B’s trust in C.

As the *Request for Recommendation* is passed along, a *recommendation path* develops: A X B X C X D means that A trusts B’s recommendation and B trusts C’s recommendation and C trusts D’s recommendation and D has a recommendation to offer. If multiple recommendation paths exist, the target (i.e., D) is given a reputation score that is the average of the recommendation paths. Recognizing that an entity’s trustworthiness changes over time, recommendations expire after a period of time or can be updated using a *Refresh* message. The *Refresh* message can also be used to revoke a recommendation.

### P2PRep – Trust Polling for Gnutella P2P Overlay Networks

When a Gnutella peer wants to search for a file, it broadcasts a Query message asking for the file. Neighboring peers who do not have the file, pass the request on to their neighbors. As a result, the requesting peer receives a set of QueryHit messages from peers who have the file and are willing to offer it to the requester. The requestor then downloads the file from one of the offerors.

P2PRep [8] modifies this process by letting the requesting peer broadcast a request for trust information for the peers who are offering the file. The responses are treated as votes which can be tabulated as desired. The requestor then directly contacts a selected set of voters to verify their vote. Having used the trust votes to select a peer from which to download, the requestor initiates a challenge-response exchange with the selected peer to confirm that it is communicating with the correct peer and then downloads the file.

P2PRep messages are carried as payload in ordinary Query and QueryHit messages. Upon receipt, a Gnutella peer processes messages through a packet processor. A P2PRep-aware Gnutella peer has a packet processor that recognizes and processes P2PRep messages in the payload. Such a peer will also have an additional Reputation Manager that corresponds directly with peers (e.g., to confirm votes), a Crypto agent to handle key generation, digital signatures, encryption, and decryption, and Experience and Credibility Repositories.

Several security measures are employed by P2PRep. First, each servant is required to have a servant_id that is a digest of a public key, obtained using a secure hash function. The servant knows the corresponding private key. For each poll, a public/private key pair is generated. The public key is included in the message requesting trust votes for a list of servants; voters then encrypt their responses with the public key to protect the confidentiality of their vote and their identity in transit and to allow the recipient to confirm the vote's integrity. The recipient uses decryption to check for tampered votes, then directly contacts remaining voters to confirm their votes. In the challenge-response the requestor requires the selected offeror to respond with a message containing its public key and the challenge signed with its private key to verify that it is communicating with the correct peer. The recipient then downloads the file, and based on the success of the download (i.e., file is not corrupted or infected), updates its trust information for the servant peer.

*Impact of P2PRep on Gnutella security*

Depending on the Gnutella variant being used, identifiers are often randomly generated upon activation. The presence of P2PRep encourages servants to keep their identifiers and build a good reputation by providing good files for download. It also discourages the practice of creating a new reputation in order to shake a bad reputation since it will take awhile to build up a sufficient reputation again. The challenge-response sequence avoids man-in-the-middle attacks.

*Impact of P2PRep on Gnutella performance*

The increase in storage capacity is proportional to the number of servants with which a peer interacts; however, network bandwidth is the more limited resource in P2P networks. Unfortunately P2PRep approximately doubles the traffic on a Gnutella network. A variant of P2PRep for low-bandwidth networks involves having the servants host signed positive votes in their favor as *credentials.* The authors don't mention the role of negative votes, if any, in this scenario. P2PRep can also be modified to evaluate other context-specific quality of service metrics beyond the normal reliable/malicious metric.

### Reputation-Based Trust and Credibility

Selcuk *et al* [41] recommend a distributed reputation-based system for pure P2P systems such as Gnutella and Kazaa. The authors do not explain how they integrate their trust mechanism into the P2P system other than to say that their "protocol relies on the P2P infrastructure to obtain the necessary reputation information". Reputation information is stored in binary trust vectors, one per peer. The length of the trust vector determines the number of reputation scores that can be stored for the peer. A similar credibility vector is also kept for each peer. Each time a new trust rating of 1 or 0 is added the vector is shifted and the oldest value is discarded. Having a separate credibility vector prevents attacks where one peer builds up a good reputation in order to recommend a malicious peer with which it is colluding.

When a peer receives responses to a query for a file, the responses are grouped according to their file hashes. In other words, they are grouped by versions of the file.

The trust coefficient of the peers that contributed the files in the group is calculated as the average of the trust ratings of the top n most trusted peers in the group, where n is configurable. (The authors recommend setting n to 1 or 2 since increased benefit is negligible for higher numbers.) If an insufficient number of the peers have trust information locally available, the querying peer sends out a request for trust information on a randomly selected set of the remaining peers so that trust will be available for n peers. Responses are weighted by the credibility rating of the responder. Interestingly, the result is only used to determine which version of the file to download. The requesting peer randomly selects which of the top n peers offering the file will be the peer from which the download will be performed. The purpose is to allow new peers to build a reputation and to not overload the trusted peers. However, a bad peer may offer a hash for a good file and then at download time, substitute a bad file, so at a minimum, the file hash must be checked after download. The authors also suggest that the file can be chunked and hashes done on each chunk so that integrity problems can be discovered prior to download the whole file.

After a successful download of a good file, the requesting peer upgrades (only) the trust rating of the peer that provided the download. If the file was corrupt, the trust rating of both the providing peer and the recommending peers is downgraded. The credibility rating is upgraded if the recommending peer was correct (i.e., peer A recommended peer B and the file from peer B was good, or peer A did not recommend per B and the tile from peer B was bad), and is downgraded (by adding a zero to the credibility vector) otherwise.

By using only the most trusted responses to evaluate which version to select, two goals are achieved. First, low-trust responses are prevented from discrediting the responses of the high-trust peers and second, the number of responses that must be authenticated are minimized. When peers return responses, they first sign the hash of the response, where the response includes the IDs of the querying and responding peers, a query ID number and the file hash being offered by the responding peer. This signature helps to prevent replay and cut-and-paste attacks. Upon receipt this signature must be authenticated.

Trust and distrust are calculated separately even though they are derived from the same trust vector. The purpose is to prevent a dishonest transaction from being too quickly masked by a series of good transactions.

To counter freeloading peers, the responding peer can prioritize its service when faced with multiple requests by providing the best service to requestors for which it has the most 1's in its trust vector since these indicate when the requesting peer has provided files to the responding peer in the past.

For authentication in the P2P environment where privacy is prized, the authors recommend binding a public/private key pair to the entity pseudonym, and using the public key itself as the pseudonym.

### PeerTrust – An adaptive, distributed trust model for P2P e-commerce communities

PeerTrust [48] uses a distributed trust model. The authors note that the model is independent of the architecture. They provide a sample architecture which uses P2P overlay concepts for storing and locating data. Each peer contains a Data Locator that figures out which node has the data of interest and a Trust Manager that submits the peer's feedback (to be stored on the peer selected by the Data Locator) and evaluates the trustworthiness of peers.

To determine the trustworthiness of a particular peer, trust evidence is gathered from all peers in the P2P network that store data for the particular peer. Caching is used to minimize the network traffic and performance hit, so that only the newest values need to be obtained from the network and calculated. Caches do not have to be large since the data they store is aggregated.

The Data Locator can use any of the usual data location schemes typically used by P2P overlay networks. These include broadcast-based methods that "do not guarantee reliable content location" (such as Chord, CAN, and Pastry use) or a distributed hash table (such as what P-Grid uses). The distributed hash table method, which is the basis for Xiong and Liu's implementation, can "deterministically map keys into points in a logical coordinate space and guarantee a definite answer to a query in a bounded number of network hops, typically in the order of logN." In P-Grid, "feedback *u* receives for each transaction are stored at designated peers that are located by hashing a unique ID of peer *u* to a data key." Feedback data includes peer *u*'s ID (as the data key), the timestamp or counter of the transaction, feedback about that transaction, the ID of the peer that provided the feedback, and any applicable transaction context information. Each peer maintains data for some peers and contains routing information for other peers so it can route any requests that it doesn't have the data to answer. To prevent data tampering and to provide fault tolerance, data can be stored on more than one peer. When data is requested, all data is acquired and a majority voting scheme is used to determine the "correct" data.

PeerTrust adaptively changes the time window of transactions to be used. If the peer's most recent trust ratings fall below a threshold, the time window is shortened. Trust ratings are binary – either 0 or 1.

Each PeerTrust peer has a PKI-based public/private key pair used for source integrity, data integrity, and encryption. The public key, or a digest of the public key, is used as the peer ID. When submitting feedback, to guarantee the integrity of the data and the authenticity of the source, the feedback submission is signed with the provider's private key and is submitted with its public key. When a peer requests trust data about a peer, it includes its public key in the search request.

The provider encrypts its response with the requester's public key for confidentiality, signs it with its own private key, and returns it with its public key. This allows the recipient to confirm the integrity and source of the data. Additionally, data replication is used to guard against tampering by the peers that store the data.

The calculated reputation score is compared to the client peer's trust threshold. This threshold can vary from peer to peer or by the context (e.g., importance or risk) of the transaction. For the second purpose, the client peer can simply choose to interact with the servant peer with the best reputation.

The authors performed a number of tests which showed the effectiveness of the trust mechanism. One finding of special significance however, was that in a collusive setting, the trust mechanism that used the trust-providing peer's trust to determine credibility actually performed worse than no trust at all. However, when credibility was calculated using feedback similarity, good peers were able to find other good peers to work with 100% of the time. The authors didn't address whether this might also allow bad peers to find each other to form collusive teams more easily.

The two factors that contribute to the overhead of the trust system are the number of lookups and the cost of each lookup. Their findings indicated that the two cached versions of PeerTrust have the same cost and scale well (estimated as O(logN)). The similarity method of computing credibility scales well even when not using cache, but the non-cached version of computing credibility by looking at the recommender's trust does not scale well (O(N)).

### Stakhanova *et al*: Using Anomaly Detection as Trust Evidence in P2P Networks

For peer-to-peer networks, Stakhanova *et al* [38] notes that peer feedback captures only the evidence that is known to peers. Activities such as the sudden download of a system file rather than the usual mp3 file may escape the attention of peers. Therefore the authors propose using anomaly detection with an unsupervised learning algorithm to provide reputation evidence in addition to peer feedback. Because predictable behavior helps to establish trust, evidence of unpredictable behavior should negatively reflect on trust. The anomaly detection component analyzes a peer's session data, looking at parameters such as connection time, connection duration, number of uploaded bytes, and number of query requests. If an anomaly is found, it calculates the degree of anomaly based on the mean and standard deviation, using Chebyshev's rule, and updates the peer's reputation accordingly. Data that are more than three standard deviations from the mean are considered an indicator of anomalous activity. The degree of anomaly is calculated throughout the session and periodically applied to the peer's reputation score. The author's experimental system showed that adding anomaly detection to peer feedback resulted in reputation scores that more accurately reflected peer behavior. In some cases, a good peer was negatively impacted but the mechanism was very successful in detecting malicious peers. The positive result of Stakhanova *et al*. [38] supports the idea of using anomaly detection as a possible indirect method for gathering reputation evidence from sources such as log files and session data. However, anomaly detection would need to be carefully designed in order to be scalable and not interfere with the adaptivity of the ACS.

## 6.8 Differences in Protecting Autonomic Managers

Unlike peer-to-peer systems, Autonomic Managers of an ACS communicate across multiple levels (with the swarming sensors and with the Orchestrating Autonomic Manager), not just with their Autonomic Manager peers. Therefore the trust model must take into consideration the need for these other entities to access and update trust data. They also lack two key issues that impact P2P systems, in that they are not anonymous and cannot shed their identity and get a new one.

# CHAPTER SEVEN

# TRUST RELATIONSHIPS IN A SWARM-BASED AUTONOMIC COMPUTING SYSTEM

Although trust relationships vary based on context, the general pattern of CID trust relationships is shown in Table 7.1[10]. In addition to the trust foundation listed in the cells of the table, entities authenticate each other prior to interacting. For completeness, the next section provides a detailed itemization of context-specific trust relationships throughout CID with a brief assessment of the foundations on which trust can be built for each relationship. The focus of this thesis, however, is the Sentinel trust relationships for which reputation is a potential foundation for trust. These relationships are shaded light gray in Tables 7.1, 7.2, and 7.3.

[10] The text in this chapter was originally published in WM Maiden. *Trust Management Considerations for the Cooperative Infrastructure Defense Framework: Trust Relationships, Evidence, and Decisions*, Pacific Northwest National Laboratory Technical Report PNNL-19117, Pacific Northwest National Laboratory, Richland, Washington, 2010. Available at http://www.pnl.gov/main/publications/external/technical_reports /PNNL-19117.pdf.

Table 7.1: Overview of direct trust relationships in CID

| | **Trusted Entity** | | | |
|---|---|---|---|---|
| **Trusting Entity** | **Sensor** | **Sentinel** | **Sergeant** | **Supervisor** |
| **Sensor** | Indirect (pheromone) | Reputation and credentials of receiving Sentinel | NA | NA |
| **Sentinel** | Reputation and credentials of sending and creating Sentinels and Policy Enforcement | Reputation and Credentials | Credentials | NA |
| **Sergeant** | NA | Reputation and Credentials | Reputation and Credentials | Credentials |
| **Supervisor** | NA | NA | Credentials | NA |

## 7.1 Detailed Analysis of CID Trust Relationships

In the remainder of this section, the CID trust relationships, many of which are shown graphically in Figure 7.1, are detailed as belonging to one of the three types of trust contexts – trust to access resources, trust in a service, and infrastructure trust. In each case the potential foundations for trust are noted.

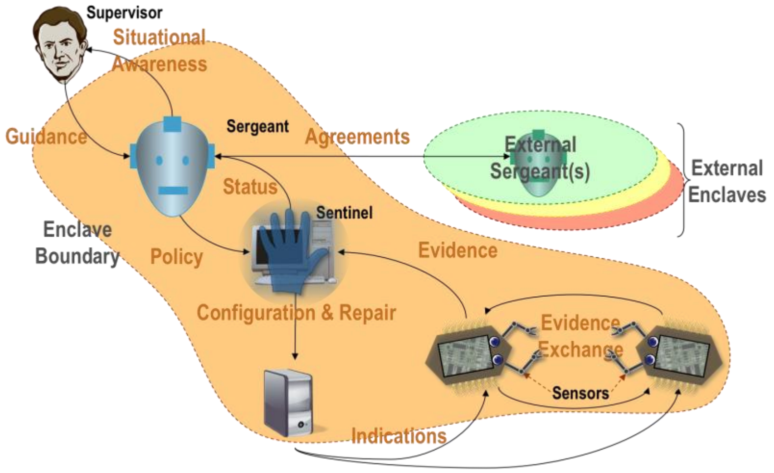

Figure7.1: CID trust relationships

### Trust to Grant Access to a Resource

This type of trust, *trust to grant access to resources*, is from the perspective of the resource provider who has a policy decision point (PDP) to protect a resource or service from unauthorized users.

In most CID cases (in contrast to the typical resource-granting scenario), the resource is pushed to the recipient or pre-allocated to the recipient rather than requested by the recipient. This provides a stronger degree of control from the start since the entity that is pushing the information must already know the receiving entity (or list of entities) to whom the information must be sent; the recipients are not unknown identities.

Table 7.2 lists CID resource providers, the resources they need to protect or control, the consumer/user, and the potential trust foundation – blind trust, control and punishment, reputation, or policy enforcement. The latter column also indicates any standard security measures that could be used in place of or in addition to trust management, so accurate decisions can be made with regard to where trust management would be most effective.

All CID entities within the context of a single security enclave are subject to authentication using identity certificates issued by a trusted third party. The exception is the cross-enclave Sergeant/Sergeant relationship.

Table 7.2: CID relationships pertaining to protection and control of resources

| **Resource to be Protected/ Controlled** | **Resource Controller** (trustor) | **Consumer / User** (trustee) | **Pushed / Requested** | **Potential Foundation for Trust** |
|---|---|---|---|---|
| Policy dialog | Supervisor | Sergeant | Pushed (initiated by Supervisor) / Requested (clarification requested by Sergeant) | Blind Trust<br>Policy enforcement:<br>• Verify that role is Sergeant and parent is the Supervisor<br>• Dialog (or decisions) must be digitally signed and logged by the Sergeant |

| **Resource to be Protected/ Controlled** | **Resource Controller** (trustor) | **Consumer / User** (trustee) | **Pushed / Requested** | **Potential Foundation for Trust** |
|---|---|---|---|---|
| Geography (the set of hosts in the enclave to which Sentinels can allow Sensors to move) | Sergeant | Sentinels | Pushed | Policy enforcement:<br>• Sergeant maintains a current list of authorized Sentinels<br>• Authenticate the Sentinels to which the Geography is pushed: Verify that role is Sentinel and parent is the Sergeant<br>• Sergeant digitally signs the Geography prior to pushing it out to the Sentinels<br>• The publication of the Geography update must be logged by the Sentinel and each log entry must be digitally signed.<br>• The receipt of the Geography update must be logged by the Sentinel and each log entry must be digitally signed<br><br>Reputation:<br>• If a Sentinel's reputation falls below a threshold, publish a new Geography that excludes that Sentinel. |

| **Resource to be Protected/ Controlled** | **Resource Controller** (trustor) | **Consumer / User** (trustee) | **Pushed / Requested** | **Potential Foundation for Trust** |
|---|---|---|---|---|
| Policy statements | Sergeant | Sentinels | Pushed | Policy enforcement:<br>• Sergeant maintains a current list of authorized Sentinels<br>• Authenticate the Sentinels to which the Geography is pushed: Verify that role is Sentinel and parent is the Sergeant<br>• Sergeant digitally signs the policy statements prior to pushing it out to the Sentinels<br>• The publication of the policy statements must be logged by the Sergeant and each log entry must be digitally signed.<br>• The receipt of the policy statements must be logged by the Sentinel and each log entry must be digitally signed |
| Execute permission, and ability to modify system configuration | Host (system administrator) | Sentinel | Pre-allocated | Policy enforcement:<br>• Privileges are granted to the Sentinel upon installation.<br><br>By granting these privileges to the Sentinel, the Sentinel is made responsible for establishing a PDP for controlling access to the host by the Sensors. Therefore, CID treats the Sentinel as a proxy for the host. |

| **Resource to be Protected/ Controlled** | **Resource Controller** (trustor) | **Consumer / User** (trustee) | **Pushed / Requested** | **Potential Foundation for Trust** |
|---|---|---|---|---|
| Share of limited CPU, memory, and disk resources | Host (system administrator) | Sentinel | Pre-allocated where possible, else requested | Blind Trust (unless allocations can be restricted upon installation)<br>Control and punishment:<br>• Resource monitoring (optional) |

| **Resource to be Protected/ Controlled** | **Resource Controller** (trustor) | **Consumer / User** (trustee) | **Pushed / Requested** | **Potential Foundation for Trust** |
|---|---|---|---|---|
| Execute permission and read access to system logs; share of limited CPU, memory, and disk resources | Receiving Sentinel | Sensors | Requested | Policy enforcement:<br>• Limit permissions to read and execute; no writing<br>• Sandboxing<br>• Limit amount of resources dedicated to a Sensor<br>• Limit number of Sensors allowed on the platform (log this number; digitally signed)<br>• Authenticate the sending and creating Sentinels prior to accepting the Sensor or allocating resources to it.<br>• Static verification of Sensor code (via digitally signed hash) upon arrival.<br>• Log each Sensor received and the sending Sentinel.<br><br>Reputation:<br>• Check sending Sentinel's reputation<br>• Check creating Sentinel's reputation<br>• Decrement the creating Sentinels' reputation if the Sensor causes problems on the host. |
| Sensor data | Sensor | Receiving Sentinel | | Reputation:<br>• Sending Sentinel checks receiving Sentinel's trust level prior to moving. |

### Trust in a Service

*Trust in a service* represents the perspective of the resource consumer who needs to be able to trust services provided by another entity. Reputation scores can be used for (1) determining whether to perform a transaction with a particular peer or (2) determining which of a list of peers is most trustworthy. CID uses reputation scores for the former purpose. This includes entities higher in the CID hierarchy which must be able to trust the entities under them to perform their duties. Table 7.3 shows the trust relationships in CID between service providers and service consumers from the perspective of the service consumers.

Table 7.3: CID relationships pertaining to trust in a service.

| **Con-sumer / User** (trustor) | **Service to be Used** | **Service Provider** (trustee) | **Potential Foundation for Trust** |
|---|---|---|---|
| Super-visor | Situational awareness | Sergeant | Blind Trust:<br>• Implicitly trust Sergeant, but observe the process (e.g., look for a hung process). Accuracy and timeliness issues may be the result of inefficiency in the code or with host or network throughput rather than maliciousness.<br>Policy Enforcement:<br>• Verify that role is Sergeant and parent is the Supervisor<br>• Sergeant must log (digitally signed) the situational awareness reports that exceed a given importance threshold. |

| **Consumer / User** (trustor) | **Service to be Used** | **Service Provider** (trustee) | **Potential Foundation for Trust** |
|---|---|---|---|
| Supervisor | Interpret and enforce policy | Sergeant | Blind Trust:<br>• Implicitly trust the Sergeant, but monitor actions and results.<br>Policy Enforcement:<br>• Verify that role is Sergeant and parent is the Supervisor<br>Control and punishment:<br>• The Sergeant is programmed to modify its behavior to maximize the value of rewards received from the Supervisor. |
| Supervis or | Authorization to negotiate with other Sergeants | Sergeant | Trust via policy enforcement:<br>• An authorization credential signed by the Supervisor can be given to the Sergeant to prove its authorization to peers. The Supervisor demonstrates degrees of trust in the Sergeant by granting credentials containing levels of authorization. |

| **Consumer / User** (trustor) | **Service to be Used** | **Service Provider** (trustee) | **Potential Foundation for Trust** |
|---|---|---|---|
| Sergeant | Policy guidance | Supervisor | Blind Trust<br>Policy Enforcement:<br>• Verify that role is Supervisor and that Supervisor's key matches the Sergeant's parent's key<br>Policy Enforcement:<br>• Log all policy changes and include the timestamp and identification of the Supervisor that made the policy change. The Supervisor's private key should be used to sign the log for non-repudiation. Although non-repudiation is not usually discussed in trust management literature as a foundation for trust, it does serve this purpose through control and punishment. |
| Sergeant | Sensor logic or service agreements offered by a Sergeant from another enclave | Other Sergeants | Policy Enforcement:<br>• Credential-based trust negotiation<br>Reputation:<br>• Reputation is used in peer-to-peer systems to detect when members are providing something bad or are just not "pulling their own weight" in the community. |
| Sergeant | Implement policy | Sentinel | Policy Enforcement:<br>• Verify that role is Sentinel and parent is the Sergeant<br>Reputation:<br>• Where possible, independently verify policy implementation and use this as input to a Sentinel's reputation. Consider using a Sensor to compare logs and settings of Sentinels vs. the Sergeant's version. |

| **Con-sumer / User** (trustor) | **Service to be Used** | **Service Provider** (trustee) | **Potential Foundation for Trust** |
|---|---|---|---|
| Sergeant | Accurate, actionable, and responsible policy dialog | Supervisor | Blind Trust<br>Policy enforcement:<br>• Verify that role is Supervisor and that Supervisor’s key matches the Sergeant’s parent’s key |
| Sergeant | Accurate and timely status | Sentinel | Policy Enforcement:<br>• Log time of request and time of receipt of information from the Sentinel.<br>Reputation:<br>• If accuracy or timeliness suffers, downgrade the Sentinel’s reputation. |
| Sentinel | Geography (the set of hosts in the enclave to which Sentinels can allow Sensors to move) | Sergeant | Blind Trust<br>Policy enforcement:<br>• Verify that role is Sergeant and that Sergeant’s key matches the Sentinel’s parent’s key. Geography received by Sentinel must be digitally signed by the Sergeant and logged by both the Sentinel and the Sergeant. Design a Sensor to compare these and report it to the next Sentinel. |
| Sentinel | Accurate and actionable policy | Sergeant | Blind Trust<br>Policy enforcement:<br>• Verify that role is Sergeant and that Sergeant’s key matches the Sentinel’s parent’s key<br>• Sergeant and Sentinel should both log (and digitally sign) all policy changes and include the timestamp. Design a Sensor to compare these and report it to the next Sentinel. |

| **Con-sumer / User** (trustor) | **Service to be Used** | **Service Provider** (trustee) | **Potential Foundation for Trust** |
|---|---|---|---|
| Sentinel | Accurate and timely information on what the Sensor found on the Sentinel's Host | Sensors | Perform checks on and before arrival (as described in Table 2), then Blind Trust. |
| Sentinel of Sensor's next host[11] | Provide pheromone | Sentinel of Sensor's current host | Policy Enforcement:<br>• Before accepting pheromone, the Sentinel should verify that the Sensor has a Sensor role credential with a chain leading back to the Sergeant.<br>• Check the digitally signed hash of the Sensor's code (generated by the Sensor's creator) that the Sensor carries with it.<br>Reputation:<br>• The sending and receiving Sentinels check each other's trust level prior to passing the Sensor. |
| Host | Monitor Sentinel | Sergeant | Blind Trust |
| Host | Reasonable and timely resolution of problems found on the host | Sentinel | Indirect. The Host will implicitly trust the Sergeant to monitor the Sentinel. |
| Host | Monitor Sensors | Sentinel | Blind Trust or Host could have process to check neighbor's view of Sentinel reputation. |

[11] To constrain the Sensors to readonly privileges, the Sentinels provide and store the pheromone on behalf of the Sensor.

| **Con-sumer / User** (trustor) | **Service to be Used** | **Service Provider** (trustee) | **Potential Foundation for Trust** |
|---|---|---|---|
| Host | Accurate and timely identification of problems | Sensors | Indirect. The Host will implicitly trust the Sentinel to monitor the Sensors. |
| Sensor | Provide reward when the Sensor has detected and reported on a problem | Sentinel | Reputation:<br>• Sending Sentinel checks receiving Sentinel's trust level prior to moving |
| Sensor | Routing to neighboring hosts | Sentinel | Reputation:<br>• Sending Sentinel checks receiving Sentinel's trust level prior to moving |
| Sensor | Accurate and timely indication of a path toward a host of interest (i.e., digital pheromone) | Other Sensors | Blind Trust<br>Policy enforcement:<br>• Indirect through Sentinel |

### Infrastructure Trust

Infrastructure trust pertains to the systems and networks upon which delivery of the service depends. CID will blindly trust the networks and is itself the mechanism for establishing host trust.

## 7.2 CID Trust Relationships Most Likely to Benefit from Trust Management

Many of CID’s trust relationships listed in the previous section can be handled efficiently and effectively through traditional security mechanisms such as authentication, digital signatures, and logging. The following relationships, however, would benefit from the addition of trust management techniques.

### Credential-Based Trust Management

All of CID’s delegation relationships require the definition of policy and creation and management of authorization credentials. Standard X.509 certificates could be used, but authorization credentials that specify finer-grained controls would be especially useful for Sentinel relationships and Sergeant-to-Sergeant cross-enclave trust negotiation.

### Reputation-Based Trust Management

Reputation-based trust management using a distributed trust model has been successfully used in communities of peers such as P2P systems, wireless sensor networks, and multi-agent communities to detect when members are providing malicious feedback, bad data, or are just not “pulling their own weight” in the community. The Sergeant-to-Sergeant cross-enclave relationship is a community of peers. Reputation would provide a mechanism for ensuring that Sergeants will be detected and isolated if they pass bad or even malicious Sensor logic to other Sergeants or if they take advantage of others’ experiences by using their shared Sensors without ever sharing their own useful Sensors for the good of the community.

Since Sentinels reside on the hosts they monitor, they are vulnerable to corruption. To detect such corruption, the Sentinels' reputation should be monitored. Reputation can be used by controlling entities (such as the Sergeant) to detect known, trusted entities (such as the Sentinels) that perhaps should no longer be trusted because of insider threat or deteriorating QoS. Here, the Sergeant is dependent on the service provided by the Sentinels under its control, similar to an employer/employee relationship[12].

Sensors are vulnerable because of their exposure to multiple hosts and because they must visit potentially-infected hosts. However, their characteristics – quantity, brief lifetimes, and minimal interactions – do not readily lend themselves to reputation-based trust management as this thesis discussed in section 5.3.

[12] Portions of this paragraph were originally published in Maiden WM, JN Haack, GA Fink, AD McKinnon, and EW Fulp. 2009. "Trust Management in Swarm-Based Autonomic Computing Systems." In *2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. IEEE Computer Society, Brisbane, Australia. © Copyright 2009 IEEE. Reprinted with permission.

# CHAPTER EIGHT

# DUALTRUST: A DISTRIBUTED TRUST MODEL FOR MANAGING THE TRUST OF AUTONOMIC MANAGERS

Swarm-based autonomic computing systems require a trust management framework that is scalable, lightweight, uses unobtrusive reputation evidence-gathering mechanisms, and is focused on the trustworthiness of the persistent autonomic elements rather than the more abundant and ephemeral sensor elements. If the trust management mechanism is too network-intensive, adds too much processing overhead, or encumbers agent adaptation, it will counter the benefits of swarm-based autonomic computing systems [34]. This thesis proposes to monitor the trust of the Sentinels as the creators of the Sensors and as the autonomic managers of the hosts on which the Sensors run. Focusing on the trustworthiness of the autonomic managers is more scalable and benefits both the autonomic manager community and the swarming sensors. This chapter introduces DualTrust, a trust model that reflects the dual nature of the autonomic manager’s horizontal peer relationships and vertical reporting relationship.

## 8.1 DualTrust Foundations

The DualTrust model is also “dual” in the sense that it uses both credentials and reputation as the foundation for determining trustworthiness. This section discusses the authentication, authorization credentials, and reputation evidence that form the foundation for trust in DualTrust.

### Authentication

Supervisors, Sergeants, and Sentinels are each assigned a public/private key pair for authentication, to conclusively confirm them as the source of a message, and to log their actions for non-repudiation purposes. Once they have been authenticated, Supervisors and Sergeants are treated as trusted entities. The Sergeant is installed on a trusted platform [42], [43] to provide hardware-based platform integrity and is protected with appropriate cyber security and physical security measures.

Each Sentinel has the public key of the Sergeant and the other Sentinels pre-loaded. Subsequent public key changes (such as the addition of a public key for a new Sentinel) are passed down to the Sentinels by the Sergeant.

Because the Sensors are numerous and ephemeral, the associated key management functions (e.g., key creation, constant distribution of public keys for new Sensors, and constant issuance of updated revocation lists) would add significant overhead for the certificate authority, the network, and the hosts, without a guaranteed corresponding increase in trust. Therefore, Sensors are not assigned an identifying key pair and are not otherwise uniquely identified.  Instead, Sensors carry a signed authorization credential as discussed in the next section.  The scalable approach for developing Sensor trust is to develop Sentinel trust since they are the creators and handlers of the Sensors. A Sensor can be trusted if the Sentinel that created it is trusted and if the Sensor's code has not changed since it was created.

### Authorization Credentials

Two forms of authorization credentials are used in Dual Trust. First, Sentinels carry an authorization credential (such as a SPKI/SDSI credential [40]) containing the ID of the Sergeant and digitally signed with the Sergeant's private key to enable verification of source and content integrity. Sensors carry a similar authorization credential containing the ID of the creating Sentinel, and digitally signed by the creating Sentinel.

The second form of authorization credential is the Geography managed by the Sergeant. The Geography, the set of hosts that the Sensor agents are allowed to visit, functions like an inverse credential revocation list. If an agent has a Sentinel role credential, it only proves that the agent was granted the credential at some point in the past. However, the existence of the Sentinel in the current Geography (as received from and signed by the Sergeant) confirms its continued authorization. When a Sergeant removes an offending Sentinel from the Geography and publishes the revised Geography to the remaining Sentinels, the Sentinel's authorization is revoked. Sentinels check the Geography as part of the authorization process and will terminate Sensors received from the banned Sentinel and will no longer send Sensors to the banned Sentinel. Furthermore, if the Sentinel in question is not a neighbor in the Geography for an interaction that requires that the Sentinels be neighbors, then the interaction will not be authorized.

### Reputation Evidence

As part of the authorization process for a requested action, a Sentinel's reputation is checked due to its location vulnerability and centrality to CID operations. There are two types of reputation evidence used in DualTrust – complaints and quality of service (QoS) observations. The latter will be referred to as reputation evidence hereafter.

A complaint consists of the following elements:

`<By_SID, About_SID, DT, Context, Signed_Hash>`

where:

- `By_SID` is the ID of the Sentinel that is filing the complaint,
- `About_SID` is the ID of the Sentinel that committed the offense,
- `DT` is the date and time of the violation,
- `Context` is the category of the offense (Former_Member, Not_A_Neighbor, Low_Reputation, or Sensor_Integrity)
- `Signed_Hash` is a cryptographic hash that the originating Sentinel creates using the first four parameters and digitally signs (e.g., with the SHA-1 algorithm) with its private key. The signed hash enables in-transit tampering to be detected by the Sergeant and the Sentinel's signature proves who sent it for non-repudiation purposes.

Reputation evidence consists of the following elements:

`<By_SID, About_SID, DT, Context, [Pass/Fail], Signed_Hash>`

where:

- `By_SID` is the ID of the Sentinel that creates the evidence,
- `About_SID` is the ID of the Sentinel about which the trust evidence was gathered,

- `DT` is the date and time of the trust evidence,
- `Context` is the trust evidence category (Sensor_Integrity, Sensor_Resourcing, Sensor_Policing, etc),
- `Pass/Fail` is the feedback, where Pass = 1 and Fail = 0, and
- `Signed_Hash` is a cryptographic hash that the originating Sentinel creates using the first five parameters and digitally signs with its private key.

## 8.2 Architectural Design Constraints

The following requirements constrain the choice of architecture:

- It must focus on the trustworthiness of the persistent autonomic elements rather than the more abundant and ephemeral sensor elements.
- The Sergeant and the Sentinels must both be able to access Sentinel trust data.
- Complete trust data must be readily available. Depending on a subset of trust data experienced by the Sentinel and its neighbors is insufficient, because neighbors will only know about certain contexts; they will be slow to learn about the damage caused elsewhere by a Sensor created by a neighboring Sentinel.
- It must be lightweight and scalable.
- It must be fault tolerant. For example, it must maintain its stability and accuracy when systems are shut down or malicious trust data is provided.
- It must not encumber agent adaptation.

## 8.3 DualTrust Architecture for Evidence Storage and Distribution

This thesis proposes to monitor the trust of the Sentinels by using a trust model that reflects the dual nature of the Sentinel’s primary relationships in the ACS architecture -- horizontal peer relationships with other Sentinels (autonomic managers) and the vertical reporting relationship with the Sergeant.

### The Horizontal Aspect of DualTrust

Xiong and Liu’s PeerTrust [48], described earlier in this thesis, provides the inspiration for the distributed trust model used between the Sentinels. In PeerTrust, evidence is routed for storage just as P2P files are routed for storage, and trust evidence requests are routed for fulfillment just as P2P file requests are routed. Each peer stores a full set of trust data for one or more other peers, and each peer’s trust data is stored on one or more other peers to allow a voting process to be used to detect evidence tampering.

In CID, Sentinels have no need of a P2P-style routing mechanism since the Sentinels all belong to the same network security domain. Instead, they can directly contact the storing Sentinel to gather or write reputation data. The Sergeant, as part of its policy responsibilities, prescribes via the Geography the trust evidence storage locations for each node. The Geography is designed to minimize network hops between neighbors, which is also a desirable attribute for the nodes on which to store trust evidence.

This design has the benefit of making complete reputation data readily available to all peer entities and also to the Sergeant. This contrasts with another type of distributed trust model wherein direct trust observations are stored locally and must be augmented by recommendations from others. In such a model, a peer must send multiple trust requests to neighbors and via those neighbors to their neighbors, which adds to the network communication load and may still stop short of resulting in the retrieval of a complete set of trust data.  Such a design would also make it awkward for the non-peer Sergeant to retrieve trust data.

Figure 8.1 shows a reputation evidence collection scenario and the Geography that will be used for illustration throughout this chapter. The details of Sentinel X's internals pertaining to evidence collection are also shown.  The "checkerboard" represents the Geography, where each square in the Geography represents a Sentinel.  In this scenario, Sentinel X is preparing to send a Sensor to neighboring Sentinel Y.  Y's reputation evidence is stored on Sentinel Z.  The Geography also includes Sentinels A, B, and C and other Sentinels that are not labeled in the Figure.  The following steps illustrate the evidence collection scenario.

*Scenario:  Sentinel X collects reputation evidence prior to passing a Sensor to Sentinel Y.*

1. The Trust Evaluation module first checks its copy of the Geography which is stored in its Policy database to ensure that Sentinel Y is both a member of the Geography and a neighbor of X in the Geography.
2. If it is, it asks the Trust Evidence Collection module to gather Y's reputation evidence.
3. The Trust Evidence Collection module checks the Geography, stored in the Policy

database, to determine which Sentinel(s) (in this case, Z) store Y’s evidence.

4. The Trust Evidence Collection module proceeds with any direct reputation observations it can make and writes the evidence it observed to Z.
5. The Trust Evidence Collection module checks its Trust Data store to determine if it contains any previously collected evidence, $Ev_x$, for Y. It notes the latest timestamp, DT, of the evidence data for Y.
6. The Trust Evidence Collection module requests evidence for Y newer than datetime DT from Z.
7. The trust evidence for Y obtained from Z, $Ev_z(Y)$, is written to the local Trust Data store.
8. The Trust Evidence Collection module then notifies the Trust Evaluation module that the evidence data is available for evaluation.
9. The Trust Evaluation module calculates the reputation score as described in section 8.4.

To reduce the amount of trust evidence being transported over the network, Sentinels store evidence they previously retrieved, so they only need to retrieve the evidence gathered since the previous request for evidence about that node. The Sergeant uses policy statements to authorize Sentinels that already have a reputation score or trust evidence for a given peer, to skip the evidence request if the score is no older than a specified interval and the score for the peer is above a specified threshold that is higher than the authorization threshold. For example, if the authorization threshold is .90, then the Sergeant may permit Sentinels to use a cached reputation score if the score is > .95

and was calculated no more than 15 minutes ago.

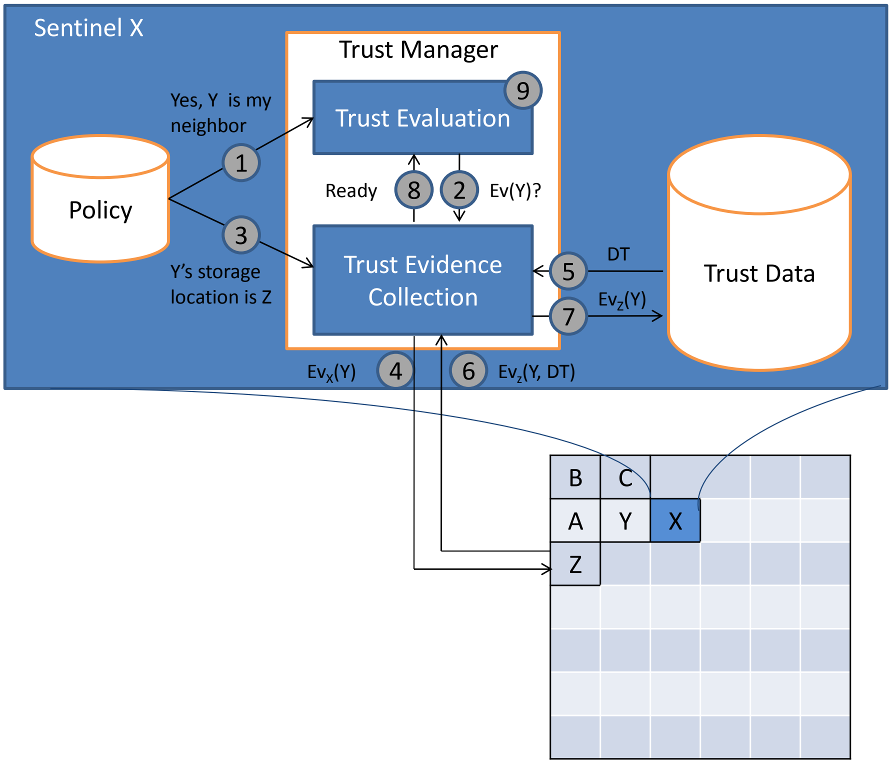

Figure 8.1: Reputation evidence collection scenario.

To minimize network traffic even further, the Sentinel can request just the reputation score (signed for non-repudiation and to prevent undetected, in-transit tampering) from the storing Sentinels. If all scores are above the threshold required for authorization, the average score can be used. Otherwise, it would be necessary to retrieve the evidence records from each storing Sentinel, using the voting mechanism to determine the best data to use. If a storing Sentinel is found to have tampered with the data, reputation evidence with the context of Evidence_Tampering will be sent to the

Sentinels that store the malicious Sentinel’s trust evidence.

When a Sentinel’s host shuts down or the Sentinel gets taken out of the CID system, the Geography is updated, along with the trust storage mappings, and re-distributed to the remaining Sentinels. The removed Sentinel’s reputation evidence storage responsibilities are assigned to other Sentinels which must gather evidence data (unless it already has it due to previous queries) in order to establish an initial reputation evidence base for their newly-assigned Sentinel. For instance, if Sentinel Y’s reputation evidence is stored on Sentinels A, B, and Z, and then Z is shut down, the Sergeant may assign Sentinel C to store Y’s reputation evidence. C will need to gather reputation evidence for Y from A and B to form its initial reputation evidence base. In cases where the data from A and B don’t agree, C will consider A and B’s reputation scores (i.e., as an indication of credibility) to determine which to believe. If A has a few newer records than B, A’s newer records are retained by C.

### The Vertical Dimension of DualTrust

In a hierarchical distributed system, the higher-level nodes of the hierarchy provide a natural location at which to place the trust evidence collection and evaluation role, so long as scalability and single-point-of-failure issues are taken into account. As far as the latter, adding this role to the Sergeant’s workload does not increase its criticality to the overall system any further. It could, however, affect scalability because a hierarchical trust model minimizes overall communication but focuses it on a comparatively few nodes near the top of the hierarchy. To minimize scalability issues, this thesis recommends a complaint-based model that parallels the process where consumers register complaints about untrustworthy businesses with the Better Business Bureau [5]. Aberer and Despotovic [1] recommend a complaint-based model when it can be expected that trust usually exists and malicious behavior is the exception, which can be expected to be the case in a managed, intra-domain environment. Because positive feedback is not recorded, the increase in network traffic due to trust feedback is greatly reduced, making this model relatively lightweight. The complaint mechanism is reserved for actions that are clearly malicious in nature and therefore require the immediate attention of the Sergeant. Example contexts include Authorization_Violation, Sensor_Integrity, and Evidence_Tampering. Only complaints that need to receive the immediate attention of the Sergeant would be sent directly to the Sergeant. The Sergeant will first check the reputation score of the reporting Sentinel as an indicator of its credibility and also check the reputation of the allegedly offending Sentinel. Trustworthiness is not calculated in the complaint model; instead, a complaint triggers a policy-based response. The Sergeant may remove the offending Sentinel from the Geography and publish the revised Geography to the remaining Sentinels, wait for additional complaints to corroborate the initial complaint, or, if the offending Sentinel recently alerted the Sergeant that it was dealing with problems on its host, the Sergeant may choose to wait for a period of time while Sensors continue to characterize the problem (see Figure 8.2). The latter option reflects the fact that the Sentinels are trusted separately from their hosts, as described in Section 5.2.

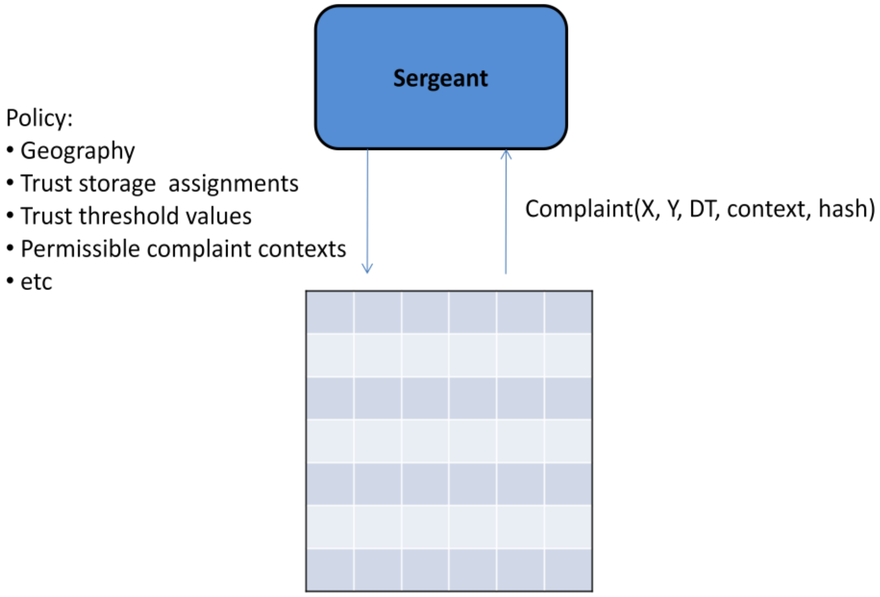
Sergeant
Policy:
• Geography
• Trust storage assignments
• Trust threshold values
• Permissible complaint contexts
• etc
Complaint(X, Y, DT, context, hash)

Figure 8.2: The Sergeant's hierarchical relationship to the Sentinels.

### Global Trust Awareness

Global situational awareness is a key attribute in CID. DualTrust keeps the Sergeant constantly apprised of the global trust situation, first because of the complete trust data stores maintained by the assigned Sentinels which the Sergeant can periodically check (i.e., a pull mechanism) for each Sentinel and also because the complaint-based model immediately informs the Sergeant (i.e., a push mechanism) when serious trust breaches occur.

## 8.4 Trust Evaluation

Trust must be evaluated for the context and quality of service that is required [11]. For example, an e-commerce vendor may not be good at shipping products promptly, but may provide high-quality products and a generous return policy. For each of these contexts (e.g., shipping time, product quality, and return policy), there is value in separately monitoring the quality of service that is of interest (e.g., prompt, high-quality, or generous). The vendor's quality of service in one of these areas is not necessarily reflective of their quality of service in the other areas, and the type of quality metric that is of interest may be different as well (prompt vs. high-quality vs. generous).[13]

In CID, the context for which we want to measure trustworthiness is the macro-level context of infrastructure defense. Infrastructure defense requires that all entities involved have the highest level of integrity. Any indication of lack of integrity is significant in a security application, regardless of the context. Therefore, all trust evidence for a given Sentinel is incorporated into a single reputation score to indicate the Sentinel's integrity in defending the infrastructure. However, trust evidence is stored according to the context in which it occurred so the contribution of each context to the overall reputation score can be weighted to reflect the importance of that context to the overall trust (i.e., integrity) value.

Because trust changes over time, only evidence gathered in the last delta time period or in the last n interactions of a given context are included in the calculation. This prevents former good behavior from camouflaging current bad behavior. The choice of a delta time period or n interactions is configurable by the Sergeant for each context.

[13] The first three paragraphs in this section are based on material that was originally published in WM Maiden. *Trust Management Considerations for the Cooperative Infrastructure Defense Framework: Trust Relationships, Evidence, and Decisions*, Pacific Northwest National Laboratory Technical Report PNNL-19117, Pacific Northwest National Laboratory, Richland, Washington, 2010. Available at http://www.pnl.gov/main/publications/external/technical_reports/PNNL-19117.pdf.

The reputation score for a Sentinel, S, is calculated as a context(c)-weighted average using only the trust evidence that is newer than a given time delta and was reported by other Sentinels with a reputation score equal to or greater than the threshold specified by the Sergeant. The reputation of a Sentinel, R(S), is the sum, across all contexts, of the number of Passes (vs. Fails) that the Sentinel has received for context c multiplied by the context-specific weight, W, divided by the total number of trust evidence records for the Sentinel for context c. The context weights, W(c), have values between 0 and 1; the ∑W(c) must be 1. This equation produces a reputation score normalized to be between 0 and 1. When the system starts, it will take awhile to gather the trust evidence data necessary for this equation. In the meantime, if the number of evidence records received for a given Sentinel and context is 0, then the Sergeant's reputation threshold value will be used in place of P(S,c) / T(S,c).

$$R(S) = \sum_{c=1}^{n} \frac{P(S,c) * W(c)}{T(S,c)} \; where \, \frac{P(S,c)}{T(S,c)} = Threshold \; if \; T(S,c) = 0$$

As an example scenario, consider that the Sensor being passed by Sentinel X to Sentinel Y was created by Sentinel C. Before accepting the Sensor, Y checks C's reputation. The Sentinel that stores C's reputation has the evidence data shown in Table 8.1.

Table 8.1: Reputation evidence for Sentinel C

| By SID | About SID | DT | Context | Pass/Fail |
|---|---|---|---|---|
| X | C | <DT1> | Sensor_Integrity | 1 |
| X | C | <DT2> | Sensor_Resourcing | 0 |
| Z | C | <DT3> | Sensor_Policing | 1 |
| Z | C | <DT4> | Sensor_Performance | 1 |
| A | C | <DT5> | Sensor_Integrity | 1 |
| A | C | <DT6> | Sensor_Resourcing | 1 |
| B | C | <DT7> | Sensor_Policing | 1 |
| B | C | <DT8> | Sensor_Performance | 1 |
| X | C | <DT9> | Sensor_Integrity | 1 |
| X | C | <DT10> | Sensor_Resourcing | 0 |

If all evidence is used, the calculation results are as shown in Table 8.2.

Table 8.2: Reputation of Sentinel C using all evidence

| Context | Sum of Context Scores | # of Context Scores | Context Weight | R(C) |
|---|---|---|---|---|
| Sensor_Integrity | 3 | 3 | 0.4 | 0.40 |
| Sensor_Resourcing | 1 | 3 | 0.1 | 0.03 |
| Sensor_Policing | 2 | 2 | 0.2 | 0.20 |
| Sensor_Performance | 2 | 2 | 0.3 | 0.30 |
| **Total** | | | | **0.93** |

However, if the Sergeant's policy is that evidence can only be used if it was reported by a Sentinel with a reputation score greater than or equal to 0.85, then the calculation, using the reputation scores in Table 8.3, would be changed as shown in Table 8.4.

Table 8.3: Reputation scores of evidence data reporters

| Sentinel ID | Reputation Score |
|---|---|
| A | 0.79 |
| B | 0.85 |
| X | 0.95 |
| Y | 0.94 |
| Z | 0.80 |

Table 8.4: Reputation of Sentinel C using evidence from Sentinels with reputation scores >= 0.85

| Context | Sum of Context Scores | # of Context Scores | Context Weight | R(C) |
|---|---|---|---|---|
| Sensor_Integrity | 2 | 2 | 0.4 | 0.40 |
| Sensor_Resourcing | 0 | 2 | 0.1 | 0.00 |
| Sensor_Policing | 1 | 1 | 0.2 | 0.20 |
| Sensor_Performance | 1 | 1 | 0.3 | 0.30 |
| **Total** | | | | **0.90** |

To improve performance, reputation scores can be cached. However, the cached score must be invalidated when new reputation evidence is available or new context weights are distributed by the Sergeant.

## 8.5 Reputation-Enhanced Relationship Scenarios

This section considers how several trust-enhanced Sentinel relationship scenarios would unfold. It considers what trust evidence is checked, how trust evidence is collected, and how and when reputation scores are used.

This section assumes that Sentinel authentication has already occurred and both Sentinels have a Sentinel role-based authorization credential signed by the Sergeant.

Whether a Sentinel is being asked to *grant access to a resource* or is verifying *trust in a service*, it will perform three types of checks in the following order – (1) authorization, (2) any applicable trust evidence-generating checks, and (3) reputation. Trust-evidence generating checks are done prior to requesting the Sentinel's reputation evidence data so the new evidence can be included in the calculation of the reputation score.

If an authorization violation is found, a complaint is immediately filed with the Sergeant. No remaining authorization checks are performed since the authorization checks are ordered such that the broader checks are completed first, and a violation of the first renders the latter of no additional significance. The trust evidence-generating checks and reputation checks are also not performed in this case since the entity whose trustworthiness is being checked is not even legitimate.

**Horizontal Trust Relationship 1: A Sensor is passed from the sending Sentinel to the receiving Sentinel**

Before sending a Sensor to the receiving Sentinel (a neighbor in the Geography randomly selected by the Sensor), the sending Sentinel's policy enforcement point will perform the following check to determine if it can trust the receiving Sentinel. It will proceed to send the Sensor only if merited, and will otherwise require the Sensor to choose a different destination. It is not necessary to check the receiving Sentinel’s authorization because the Sentinel was randomly selected by the Sensor from the latest Geography received from the Sergeant.

*Sender’s Check 1: Does the receiving Sentinel have a good reputation?*

Trust evidence for the receiving Sentinel is gathered and a reputation score is calculated. The sending Sentinel will continue the transfer only if the receiving Sentinel’s reputation score is above a minimum threshold determined by the Sergeant’s policy.

On the other side of the transaction, the receiving Sentinel, before accepting a Sensor from a sending Sentinel, must consider whether it has sufficient trust in the sending Sentinel. (See Figure 8.3.) The receiving Sentinel checks the sending Sentinel’s current authorization and reputation. If any of the checks fail, the receiving Sentinel will either refuse the Sensor or terminate it upon arrival depending on the Sergeant’s policy.

*Receiver’s Check 1: Is the sending Sentinel a member of the current Geography?*

This is an authorization check that can be performed locally since the Sentinel receives the Geography (including updates) from the Sergeant. If the sending Sentinel is not a member of the current Geography, the receiving Sentinel will file a complaint about the sending Sentinel with the Sergeant and will then either refuse the Sensor or terminate it upon arrival, depending on the Sergeant’s policy. The context for the complaint is Former_Member. Authorization findings are not stored as trust evidence because other Sensors are able to perform the same conclusive authorization check using their local copy of the Geography without having to gather trust data or perform a calculation.

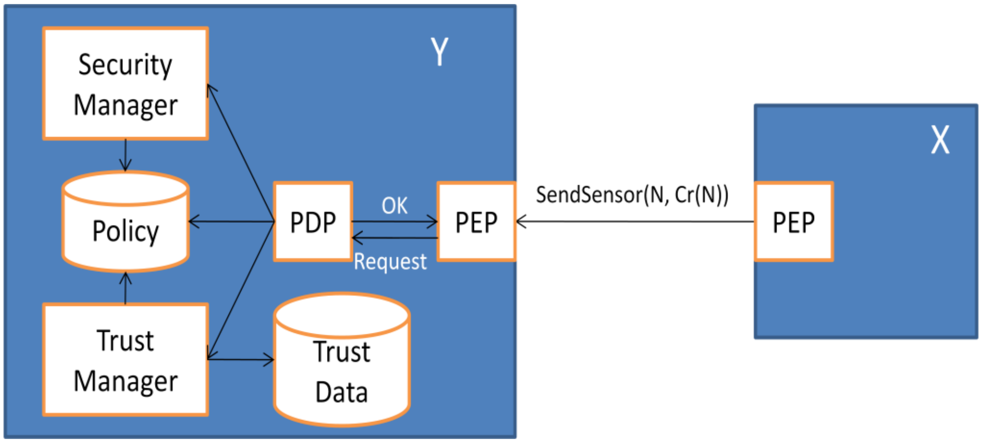

Figure 8.3: High-level architecture.
The Policy Enforcement Point (PEP) for Sentinel Y, through its Policy Decision Point (PDP), checks the sending Sentinel’s authentication and authorization credential with the Security Manager and its Geography membership and reputation with the Trust Manager before accepting a Sensor created by Sentinel N from Sentinel X.

*Receiver’s Check 2: Is the sending Sentinel a neighbor in the current Geography?*

This too is an authorization check that can be performed locally. If the sending Sentinel is not a neighbor in the current Geography, the receiving Sentinel will file a complaint about the sending Sentinel with the Sergeant and will either refuse the Sensor or terminate it upon arrival, depending on the Sergeant’s policy. The context for the complaint is Not_A_Neighbor.

*Receiver’s Check 3: Is the cryptographic hash of the Sensor’s serialized code the expected hash value?*

The receiving Sentinel stores the reputation evidence (Pass or Fail), and depending on policy, may also send a complaint to the Sergeant. In both cases, the context is Sensor_Integrity. The context weight for Sensor_Integrity should reflect the significance of code integrity.

*Receiver's Check 4: Did the sending Sentinel provide adequate resources to the Sensor?*

Assume that the Sensor carries arrival timestamps from its two most recent hosts. Based on the timestamps, the receiving Sentinel calculates whether the lag from when the Sensor was sent to the previous Sentinel to the present time exceeds a configured threshold. This would indicate that the sending Sentinel did not provide adequate resources (CPU cycles, etc.) to the Sensor to allow it to run in a timely fashion. This could be due to an overloaded system or due to maliciousness; therefore, the threshold value should be set high enough to not regularly punish the Sentinel of a busy system. The threshold is configured and distributed as part of the Sergeant's policy. The receiving Sentinel stores the trust evidence (Pass or Fail) with the context set to Sensor_Resourcing.

*Receiver's Check 5: Does the sending Sentinel have a good reputation?*

Trust evidence for the sending Sentinel is gathered and a reputation score is calculated. The sending Sentinel's reputation score must be above a minimum threshold determined by the Sergeant's policy.

*Horizontal Trust Relationship 2: Hosting Sentinel executes Sensor created by the creating Sentinel*

Because the Sensor's code hash was checked upon receipt by the receiving Sentinel (called the hosting Sentinel in the context of this trust relationship), this section assumes that the code has not changed and, therefore, any issues with Sensor execution reflect on the creating Sentinel rather than the sending Sentinel. Before allowing the Sensor to execute, the hosting Sentinel's policy decision point considers the creating Sentinel's current authorization and reputation.

*Host's Check 1: Is the creating Sentinel a member of the current Geography?*

If the creating Sentinel, as determined from the Sensor's authorization credential, is not a member of the current Geography, the hosting Sentinel will infer that the creating Sentinel is not trustworthy and will terminate the Sensor. This is not reported to the Sergeant as an authorization violation because the creating Sentinel is not responsible for the fact that the Sensors it created are still active once it has been removed from the Geography. However, if the current Geography was received from the Sergeant prior to the Sensor handoff from the prior Sentinel (based on the timestamps the Sensor carries), the hosting Sentinel knows that the sending Sentinel should have terminated the Sensor. It writes trust evidence for the sending Sentinel (Pass or Fail) using the context of Sensor_Policing.

*Host's Check 2: Does the creating Sentinel have a good reputation?*

Trust evidence for the creating Sentinel is gathered and a reputation score is calculated. If the score is above a minimum threshold determined by the Sergeant's policy, the executing Sentinel proceeds to run the mobile Sensor agent's code.

*Host's Check 3: Did the Sensor perform in a trustworthy manner?*

After the Sensor code has been run, the Sentinel provides reputation feedback for the creating Sentinel based on factors such as the following. If the Sensor fails any of these checks, trust evidence is recorded with feedback=Fail; otherwise, feedback is set to Pass. (Alternatively, trust evidence could be stored for each of these separately, but this would increase network traffic.) The context is Sensor_Performance.

- Did the attempted or actual privileges of the Sensor exceed what was allowed?
- Did the attempted or actual resource consumption of the Sensor exceed what was allowed?
- Did the Sensor falsely report data that the executing Sentinel knows is not true?
- Did the Sensor disrupt the host in some way?

The mechanisms by which these factors are determined are outside the scope of this thesis, which simply assumes that such a mechanism exists.

**Vertical Trust Relationship: Sergeant entrusts Sentinel with carrying out policy**

The Sergeant must be able to depend upon the Sentinels to carry out its policies, so it exercises a continual oversight function much like a parent/child or employer/employee relationship. If a Sentinel is found to not be in compliance with policy, the Sergeant has the option of removing the Sentinel from the Geography until the problem is resolved, such as by cleaning, re-configuring, and rebooting the host. If the Sergeant has received recent information from the Sentinel itself that its host is under attack, the Sergeant may choose to delay its response to allow Sensors to visit the Sentinel to help characterize the problem.

The Sergeant has multiple methods by which to determine whether a Sentinel is in compliance with policy:

- The Sentinel’s global reputation (based on trust evidence periodically gathered from the Sentinels and calculated) falls below the threshold set by the Sergeant. The interval at which the Sergeant calculates a global reputation score for each Sentinel is part of the Sergeant’s policy. Additionally, the Sergeant is also prompted to calculate the global trust of a Sentinel when another Sentinel reports that its trust has fallen below the allowed threshold (i.e., context is Low_Reputation).
- The Sergeant has received one or more complaints about authorization violations and the reporting Sentinel’s reputation score (which indicates its credibility) is above the threshold set by the Sergeant.
- Anomaly detection mechanisms identify policy compliance issues.

Anomaly detection has been shown to improve the accuracy of reputation scores [38]. For example, mobile monitoring agents could gather anomaly information. Mobile monitoring agents are outside the scope of this thesis, but briefly, the concept is that they could be specialized Sensor agents sent by the Sergeant (rather than a Sentinel) that would carry encrypted results and return to the Sergeant after a specified interval or a specified number of Sentinel visits. Their authorization credential would show they were created by the Sergeant. The agent would need to carry a policy version number or policy timestamp to prevent it from comparing old Sergeant policy against a newer version sent by the Sergeant to the Sentinel since the agent was dispatched. Just as other Sensors possess one of many classifiers, the mobile monitoring agents would have one of many policy-monitoring classifiers. Once the monitoring agent returns to the Sergeant, the Sergeant could either take immediate action based on the agent's payload of trust evidence and/or could choose to store the trust evidence in the Trust Data store.

Policy indicators that mobile monitor agents could check include the following:

- Do policy (including Geography) changes logged as received by the Sentinel match the Sergeant's log of what was sent to the Sentinel?
- Does the Geography in use by the Sentinel match the Sergeant's current Geography?
- Is there a specific element of prescribed policy that the Sentinel has not implemented?

- Are specific aspects of the Sentinel’s host state inconsistent with the Sentinel’s log of policy actions?
- Are recorded response times within range, such as when a particular policy change was sent versus when it was implemented according to the log?
- Is the ratio of Sensors forwarded to vs. received from the Sentinel higher than average by a given margin and not substantiated in the Sentinel’s logs? This would indicate that the Sentinel is terminating an unusual number of Sensors.

# CHAPTER NINE

# CONCLUSIONS AND FUTURE WORK

## 9.1 Conclusions

This thesis has introduced the DualTrust model for managing the trust of the autonomic managers for the protection of both the autonomic manager community and the swarming sensors. The DualTrust model meets the architectural requirements listed in section 8.2. DualTrust focuses on the more persistent elements of the autonomic computing system, the autonomic managers, because reputation evidence can be gathered over a longer time and is therefore more meaningful for persistent elements than for ephemeral elements. There are also considerably fewer autonomic managers than swarming Sensors, so this focus addresses scalability as well.

Because the evidence storage mechanism stores complete reputation evidence for a given peer replicated across a few selected nodes, it meets the requirement for complete reputation evidence data to be readily available without having to request it from all of the Sentinels. It also enables the Sergeant to have ready access to reputation evidence even though it is not one of the peers. The replication of the evidence provides for fault tolerance, including tolerance for malicious behavior.

The design of the evidence distribution mechanism minimizes network traffic for scalability. To further improve scalability, an option is presented for having the storing Sentinel calculate and distribute reputation scores rather than reputation evidence.

With the exception of the optional, specialized monitoring agents, the DualTrust model does not constrain adaptation of the swarm or the swarming Sensors in any way. One reason that Sensors can be created and adapted freely is that Sensor trust is handled through trust in the creating Sentinels and through Sensor code integrity rather than requiring the overhead of identification and key pairs for each Sensor.

## 9.2 Future Work

Additional research is needed to address the pheromone trust relationship, secure the Sensor authorization credentials, implement DualTrust, and run performance tests using the evidence distribution model and the score distribution model.

In addition, DualTrust is believed to be applicable to other architectures that consist of a set of peers with a hierarchical manager, but this should be researched and verified.

# BIBLIOGRAPHY

[1] K. Aberer and Z. Despotovic. "Managing Trust in a Peer-2-Peer Information System," *Proc. of the Tenth International Conference on Information and Knowledge Management (CIKM)*, 2001.

[2] I. M. Atakli, H. Hu, Y. Chen, W. S. Ku, and Z. Su. "Malicious node detection in wireless sensor networks using weighted trust evaluation," *Proc. of the 2008 Spring Simulation Multiconference*, pp. 836-843, 2008.

[3] A. Abdul-Rahman and S. Hailes. "A Distributed Trust Model," *Proc. of the 1997 Workshop on New Security Paradigms*, ACM Press, pp. 48-60, 1998.

[4] F. Azzedin and M. Maheswaran. "Evolving and Managing Trust in Grid Computing", *Proc. of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '02)*, pp. 1424-1429, 2002.

[5] Better Business Bureau, http://www.bbb.org/us/.

[6] M. Blaze, J. Feigenbaum and J. Lacy. "Decentralized Trust Management", *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pp. 164-173, 1996.

[7] Venkat Balakrishnan, Vijay Varadharajan, Uday Tupakula, and Phillip Lucs. “TEAM: Trust Enhanced Security Architecture for Mobile Ad-hoc Networks”, *Proc. of the 15th IEEE International Conference on Networks,* pp. 182-187, 2007.

[8] F. Cornelli, E. Damiani, S. D. di Vimercati, S. Paraboschi, and P. Samarati. “Choosing reputable servents in a P2P network”. *Proc. of the 11th International Conference on World Wide Web (WWW '02).* ACM, New York, NY, 376-386, 2002.

[9] D.W. Carman, P.S. Kruus, and B.J. Matt. *Constraints and Approaches for Distributed Sensor Network Security,* NAI Labs Technical Report #00-010. 2000.

[10] D.M. Chess, C.C. Palmer, and S.R. White. “Security in an Autonomic Computing Environment”, *IBM Syst. J.,* vol. 42, pp. 107-118, 2003.

[11] I. Dionysiou. *Dynamic and Composable Trust for Indirect Interactions*, Ph.D. Dissertation, Washington State University, School of Electrical Engineering and Computer Science, Pullman, Washington, 2006, http://research.wsulibs.wsu.edu:8080/dspace/bitstream/2376/551/1/i_dionysiou_072406.pdf.

[12] Wenliang Du, Lei Fang, and Peng Ning. “LAD: Localization Anomaly Detection for Wireless Sensor Networks”. *Proc. of the 19th International Parallel and Distributed Processing Symposium (IPDPS’05)*, 2005.

[13] M Dorigo, V. Maniezzo, and A. Colorni. “The Ant System: Optimization by a Colony of Cooperating Agents”, *IEEE Transactions on Systems, Man, and Cybernetics,* Part B: Cybernetics, vol. 26, pp. 29-41, 1996.

[14] Eschenauer, Laurent and Gligor, Virgil D. “A Key-Management Scheme for Distributed Sensor Networks”. *Proc. of the 9th ACM Conference on Computer and Communications Security*, 2002.

[15] Laurent Eschenauer, Virgil D. Gligor, and John Baras. “On Trust Establishment in Mobile Ad-Hoc Networks”, *Proc. of the Security Protocols Workshop*, 2002.

[16] E. Friedman and P. Resnick. “The Social Cost of Cheap Pseudonyms,” *J. Economics and Management Strategy*, vol. 10, 1998.

[17] M.C. Fernandez-Gago, R. Roman, and J. Lopez. “Survey on the Applicability of Trust Management Systems for Wireless Sensor Networks”, *Third Int’l Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, 2007.

[18] T. Grandison. *Trust Management for Internet Applications*, Ph.D. Dissertation, University of London, England, 2003. Available at http://pubs.doc.ic.ac.uk/trust-managem-for-internet-app/trust-managem-for-internet-app.pdf.

[19] T. Grandison and M. Sloman. “A Survey of Trust in Internet Applications”, *IEEE Communications Surveys and Tutorials*, vol. 4, no. 4, pp. 2-16, 2000.

[20] Jereme N. Haack, Glenn A. Fink, Wendy M. Maiden, David McKinnon, and Errin W. Fulp. “Mixed-Initiative Cyber Security: Putting humans in the right loop”, *Mixed-Initiative Multiagent Systems Workshop*, 2009. Available at http://u.cs.biu.ac.il/~sarned/MIMS_2009/papers/mims2009_Haack.pdf.

[21] Lei Huang, Lei Li, and Qiang Tan. “Behavior-Based Trust in Wireless Sensor Network”, *APWeb Workshops*, Springer-Verlag, 2006.

[22] J. Hoffmeyer. “The Swarming Body”, Semiotics Around the World. *Proc. Fifth Congress of the Int’l Assoc. for Semiotic Studies*, pp. 937-940, 1994.

[23] IBM Corporation: *An Architectural Blueprint for Autonomic Computing*, (4th ed.), IBM Corporation, New York, 2006. Available at http://www-01.ibm.com/software/tivoli/ autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf.

[24] W. Junior, T. Figueriredo, H.-C. Wong, and A. Loureiro. “Malicious Node Detection in Wireless Sensor Networks”, *Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS’04)*, 2004.

[25] W. Jansen and T. Karygiannis. *NIST Special Publication 800-19 – Mobile Agent Security*. National Institute of Standards and Technology, Gaithersburg, Maryland, 2000. Available at http://csrc.nist.gov/publications/nistpubs/800-19/sp800-19.pdf

[26] Johnson, D.B., Maltz, D.A., and Hu, Y. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, IETF MANET Internet Draft, 2003.

[27] J.O. Kephart and D.M. Chess. “The Vision of Autonomic Computing”, *Computer*, vol. 36, pp. 41-50, 2003.

[28] Chris Karlof and David Wagner. “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures”, *First IEEE International Workshop on Sensor Network Protocols and Applications*, 2002.

[29] K.-J. Lin, H. Lu, T. Yu, and C. Tai. “A Reputation and Trust Management Broker Framework for Web Applications”, *Proc. of the 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Service*, pp. 262-269, 2005.

[30] Huaizhi Li and Mukesh Singhal. “Trust Management in Distributed Systems”, *Computer*, IEEE Computer Society, February 2007.

[31] C. Lin, V. Varadharajan, Y. Wang, and V. Pruthi. “Trust Enhanced Security for Mobile Agents”, *Seventh IEEE Int’l Conf. on e-Commerce Technology*, pp. 231-238, 2005.

[32] C. Lin, V. Varadharajan, Y. Wang, and V. Pruthi. “Security and Trust Management in Mobile Agents: A New Perspective”, *2nd Int’l Conf. on Mobile Technology, Applications and Systems*, pp. 1-9, IEEE Press, New York, 2005.

[33] C. Lin and V. Varadharajan. "Trust Enhanced Security – A New Philosophy for Secure Collaboration of Mobile Agents", *Int'l Conf. on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1-8, IEEE Press, New York, 2006.

[34] W.M. Maiden, J.N. Haack, G.A. Fink, A.D. McKinnon, E.W. Fulp. "Trust Management in Swarm-Based Autonomic Computing Systems", *Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, pp.46-53, 2009.

[35] W.M. Maiden. *Trust Management Considerations for the Cooperative Infrastructure Defense Framework: Trust Relationships, Evidence, and Decisions*, Pacific Northwest National Laboratory Technical Report PNNL-19117. Pacific Northwest National Laboratory, Richland, Washington, 2009. Available at http://www.pnl.gov/main/publications/external/technical_reports/PNNL-19117.pdf.

[36] "How PGP works", http://www.pgpi.org/doc/pgpintro.

[37] Asad Amir Pirzada and Chris McDonald. "Establishing Trust in Pure Ad-hoc Networks", *27th Australasian Computer Science Conference*, 2004.

[38] N. Stakhanova, S. Basu, J. Wong, and O. Stakhanov. "Trust Framework for P2P Networks Using Peer-Profile Based Anomaly Technique", *Proc. of the Second International Workshop on Security in Distributed Computing Systems (SDCS)*, pp. 203-209, IEEE Computer Society, Washington, DC, 2005.

[39] K. Seamons, T. Chan, E. Child, M. Halcrow, A. Hess, J. Holt, J. Jacobson, R. Jarvis, A. Patty, B. Smith, T. Sundelin, and L. Yu. "TrustBuilder: Negotiating Trust in Dynamic Coalitions", *Proc. of the DARPA Information Survivability Conference and Exposition (DISCEX'03)*, vol. 2, pp. 49-51, 2003.

[40] SPKI, http://world.std.com/~cme/html/spki.html.

[41] A. A. Selcuk, Ersin Uzun, Mark Resat Pariente. "A Reputation-Based Trust Management System for P2P Networks", *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, pp. 251-258, 2004.

[42] Trusted Computing Group, http://www.trustedcomputinggroup.org/trusted_computing/benefits.

[43] Trusted Computing Group, http://en.wikipedia.org/wiki/Trusted_Computing_Group.

[44] H.K. Tan and L. Moreau. "Certificates for Mobile Code Security", *Proc. 2002 ACM Symposium on Applied Computing*, pp. 76-81, ACM, New York, 2002.

[45] Uwe G. Wilhelm, Sebastian Staamann, and Levente Buttyán. "On the Problem of Trust in Mobile Agent Systems", *Proc. of the Symposium on Network and Distributed System Security*, pp. 114-124, 1998.

[46] D. Wolpert and K. Tumer. *An Introduction to Collective Intelligence*. NASA-ARC-IC-99-63. NASA Ames Research Center, California, 1999, http://arxiv.org/PS_cache/cs/ pdf/9908/9908014v1.pdf

[47] Mingdi Xu, Ruiying Du, Huanquo Zhang. “A New Hierarchical Trusted Model for Wireless Sensor Networks”, *2006 International Conference on Computational Intelligence and Security*, pp. 1541-1544, 2006.

[48] L. Xiong and L. Liu. “PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities”, *IEEE Trans. Knowl. Data Eng.*, vol. 16, pp. 843-857, 2004.