A TRUST SUPPORTIVE FRAMEWORK FOR PERVASIVE COMPUTING SYSTEMS

by

Dichao Peng

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Information Technology

Charlotte

2009

Approved by:

____
Dr. Zhaoyu Liu

____
Dr. Yuliang Zheng

____
Dr. Teresa Dahlberg

____
Dr. Yu Wang

____
Dr. Robert Cox

ABSTRACT

DICHAO PENG. A trust supportive framework for pervasive computing systems.
(Under the direction of DR. ZHAOYU LIU)

Recent years have witnessed the emergence and rapid growth of pervasive computing technologies such as mobile ad hoc networks, radio frequency identification (RFID), Wi-Fi etc. Many researches are proposed to provide services while hiding the computing systems into the background environment. Trust is of critical importance to protect service integrity & availability as well as user privacies. In our research, we design a trust-supportive framework for heterogeneous pervasive devices to collaborate with high security confidence while vanishing the details to the background. We design the overall system architecture and investigate its components and their relations, then we jump into details of the critical components such as authentication and/or identification and trust management. With our trust-supportive framework, the pervasive computing system can have low-cost, privacy-friendly and secure environment for its vast amount of services.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

With the rapid development of distributed and open infrastructures, including the internet and various types of wireless networks, computing systems are moving toward the next era, pervasive computing, in which various devices, software agents and services are all expected to seamlessly integrate and cooperate for achieving user objectives in an anywhere-and-anytime fashion.

The principle of pervasive computing is for computing devices to weave themselves into the fabric of everyday life until they are indistinguishable from it [78]. The computing nodes, either big as a laptop or small as an RFID tag, will use resources on demand for communication, information retrieval, and computation, in a manner of taking advantages of pervasive resources, instead of depending on personal computing platforms. Physical and temporal boundaries will fade away and people can access information transparently and freely.

So far, many research groups are making progress on smoothly and efficiently providing unanimous services by coordinating dynamic coalitions between heterogeneous devices. But due to the open infrastructure of pervasive computing spaces, to provide trusted coalition is different than traditional static infrastructures such as the Internet. In pervasive computing systems, supporting efficient and trusted dynamic coalitions without security and privacy violations is challenging due to the following difficulties:

- Limited system resources. Pervasive computing systems involve heterogeneous devices with limited computational power. Many devices (e.g. passive RFID tags) neither have enough power supply nor the computational ability to afford security functions, such as executing cryptographic algorithms, storing security policies, etc.

- Open air communication. “Always-on" wireless connectivity makes the network communication supporting pervasive environment more susceptible to eavesdropping and packet hijacking than any other computation environment.

- Dynamic population. Service providers as well as service consumers constantly move in and out of the space boundary, which results in the difficulty of evaluating the trustworthiness of a new member.

- Privacy violation. Pervasive computing devices are usually physically located around their users, which makes it possible for attackers to easily infer their privacies by collecting data from multiple resources.

Take this application scenario as an example: a student walks into a smart room and wants to send some documents on her PDA to print. But since this is the first time she uses the services in the smart room, her PDA and the printer do not know about each other. The PDA may have security concerns about whether the printer is dishonest, with some spyware installed to secretly copy the documents printed. The printer, on the other hand, wants to protect itself from malicious users who aims to exhaust its ink by sending lengthy jobs and also causes denial of services to other legitimate users. Although both the PDA and the printer have security polices to protect themselves, the trustworthiness between them should not easily fail solely because they are strangers to each other (which is very common in pervasive computing systems). In addition, the printer may have very limited computational resources to store its dynamic security policies and perform cryptographic functions. Thus to ensure the communication secrecy and data integrity becomes another challenge.

Our research provides a trust-supportive framework to address these challenges in pervasive computing systems. Our framework architecture relies on reconfigurable and on-demand-assembled security components collaborating with each other. We will explain the architecture and investigate the critical components such as user authentication & identification, trust negotiation and maintenance.

The rest of the thesis is organized as follows. Chapter 2 presents an overview of the trust-supportive architecture at the system level. We elaborate the role of our system within the pervasive computing environment and briefly explain the components that make up the system. Chapter 3 dives more into one of the key components, user recognition and discuss the major methods: user authentication and identification. We explain in details the challenges specific to pervasive computing and present our solutions. Chapter 4 discusses another critical component of how trust relations will be managed in our system. We elaborate the trust model, the process of trust negotiation and present a systematic algorithm for trust calculation. Related work is presented in Chapter 5 and Chapter 6 concludes the thesis.

Before we move to the next chapter, we would like to acknowledge the places where our publications appeared. The work on architecture design of the trust-supportive systems appeared as [49]. The work on user authentication/identification in chapter 3 appeared as [48, 50, 51, 52, 53].The work on trust management in chapter 4 appeared as [54].

CHAPTER 2: SYSTEM ARCHITECTURE OVERVIEW

In the context of Pervasive Computing, the dynamic computing devices cooperate to seamlessly provide services while hiding themselves in the background environment. Thus it is a nice feature of the system to be able to have on-demand resembled service components to fulfill the ever-changing environment and user requirement. We propose a middleware based approach to abstract and represent security requirements as several middleware components. These reconfigurable components reside on a layer of the actual pervasive computing devices, between the applications (either client or service) and the operating system and networks. These components provide reconfigurable security services for the application, such as user identification, trust establishment and management, data encryption, security service discovery, etc.

Our trust middleware architecture (Figure 2.1) is compatible with general middleware components. For example, in Gaia [68] reconfigurable middleware components aggregate to provide a service on demand. Our trust-supportive middleware components is in parallel with these service components and supports various security functions. Similar to general middleware architecture, on top of the architecture is the application layer, including both the service consumer and provider's applications. Beneath the application layer is the middleware framework, composed of both general service middleware and the security middleware, which includes the trust management framework that will be explained in the following chapters.

These on-demand-assembled security components compose of the set of security services that one pervasive computing system supports. For example, the encryption component provides data encryption with popular standard algorithms, as well as user specified algorithms; an intrusion detection component provides a device with diagnostic interfaces

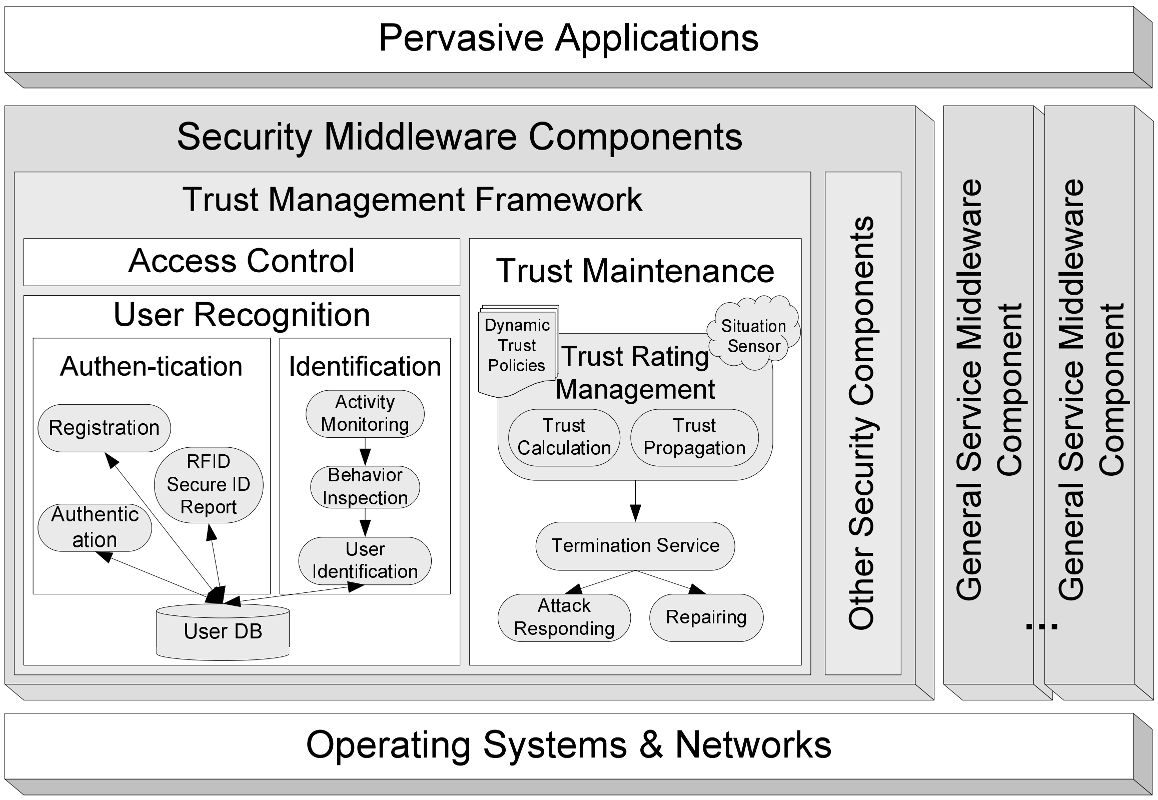


Figure 2.1: Trust Supportive Framework Architecture

to identify intrusions. Of all the security components, trust management is especially a challenging and important one because of the dynamic characteristic of pervasive computing [81]. It is a fundamental component dedicated to support establishing and maintaining trust between applications on heterogeneous devices. We will focus on the designing details of the trust management framework and elaborate its components.

The trust management framework handles service requests from application layer and instantiate a trust establishment agent to establish mutual trust. There are three components available for the mission: access control, user recognition and trust maintenance. Obviously trust maintenance plays a key role in this framework: it handles trust level calculation, trust relation propagation, adaptation to the environment context, exception responding and etc. But before the trust relation can be established, it is also critical to verify the user's identity so that the computing agents know exactly who they are trusting. We acknowledge this component as user recognition. Depending on whether there is available pre-shared profile, user recognition can be roughly divided into two categories: authentication and identification. The authentication component behaves the same as the widely adopted registration-authentication based security model, which takes advantage of the previously established user profile through registration to challenge and verify the user's identity. We will not dive into this topic due to the existing full-fledged researches. Instead, we are in more concern about user recognition in the case where no previous knowledge is available, which could be very common for pervasive computing systems. Finally with properly established trust, the access control component restricts resource access based on the trust level and applicable policies.

2.1 Authentication and Access Control

As we discussed, authentication and access control are among the most popular schemes adopted by many research approaches to secure pervasive computing systems.

In terms of access control component, we propose an access control system following the traditional Mandatory Access Control (MAC) approach. In our model, we assign each service a sensitivity label, suggesting the required level of trust for access. This sensitivity label is dynamically changing with the variation of environment and coalition relationships, according to the service's reconfigurable trust policy. For example, the smart card door locker service will be marked with higher sensitivity label during the night than when it is daytime. A visitor who can walk into a building freely in the morning may be denied by the door locker in the night without proof of adequate trust level. For another example, a user's PDA may set the air conditioner to her preferred temperature when she is the only one in the room. But when the smart room detects more than one people present, the air conditioner may require higher trust level to perform the same action since the temperature will affect the comfortableness of everybody in the room.

On the other hand, each client obtains a sensitivity label through trust establishment, which specifies its level of trust. In order to access a given service, the client must have a sensitivity level equal to or higher than the requested service.

Here the terms “service" and “client" are relative: a projector in the conference room can be a service when dealing with request from laptops for a slides show; but when it requests the smart room to turn off the lights, the projector becomes a client. Thus the projector can have a sensitivity label indicating its trust level, while its security policy may maintain another label as requirement for computing nodes who wants to use its display service.

Since this MAC access control component requires trust level as input, it relies on one of the two trust establishment models: traditional authentication method and/or user identification.

Although the term “authentication" varies from simple identity reporting such as passive RFID badge authentication, to the most complicated PKI-based user-password authentication, the underlying essence is the same: using previously established user profile to claim and prove identity. Authentication plays an important role to secure pervasive computing systems, especially in systems with less dynamic idiosyncrasy. It is a natural way of establishing trust by identifying a user device with previously known reputation and its user role within the system. But as we discussed, in some cases previously established user profiles are not available, where identification becomes handy, standing in parallel with authentication to address this problem.

2.2 User Identification

A well-established trust management system should be able to recognize a bad computing node (either a client or service node) and limit its access to the resources afterwards. But due to the nature of pervasive computing system of dynamic population, a user should not be rejected only because she is new to the computing space. Thus the attacker can move her device out of the pervasive computing space, modify the identity related information such as physical address of its networking interface and move back in again claiming herself as a newcomer. Or she may simply use another device so that the logged historical bad behavior of the old one will be useless: instead of being a notorious user with bad histories, she becomes a newcomer that should deserve a warm welcome.

We propose a user identification component to address this issue by recognizing a user even if the onboard device has been changed. The user identification component is designed based on the assumption that every user behaves uniquely during a connection session. The hours of day one usually comes, the displaying devices one prefers, the networking traffic compositions, the preferred access point (may suggest preferred position/location), the service set one uses, the temperature one prefers and so on, all the information together can help identify a user. Our user identification component creates profiles for each detected ill-behaved device and its user, with a matrix of the user preferences. When a self-claimed newcomer is trying to enter the pervasive computing space, a signature based inspection will be applied to the matrix of every attacker profiles. A dishonest veteran that claims itself as a newcomer will be denied for further services.

To recognize dishonest behavior that tries to harm the system and to support trust rating component's evaluation, the activity monitoring and logging component is dedicated to track user behavior and log this information for further inspection and evaluation. The designing principle of activity monitoring component is to make the logging information thin and efficient. To keep the logging database thin, only critical activities such as login failures should be stored for a reasonably long time (e.g. one year). Other logging information will only stay in the repository for a short time and then be deleted.

Based on the logging information collected by activity monitoring component, behavior inspection applies both abnormality based and signature based inspection methods to recognize security breaches. For example, if a supermarket customer is exploring the RFID product querying system by scanning hundreds of items per minute with an RFID reader, the abnormality analysis engine on activity monitoring component will generate an alarm, indicating the customer is behaving abnormally. This alarm may be applied to reevaluate client trustworthiness as a feedback to the trust rating management component.

2.3 Trust Maintenance

When a user's identity is successfully recognized, either by authentication or identification, the trust maintenance component will be activated to start monitoring and managing trust of both client and service. Since the service is at a public position, making it more vulnerable to malicious attacks compared to the client, the trust maintenance component is more service-oriented to protect service resources from malicious clients.

Lying in the center of trust maintenance is the trust rating management component. It adopts a rating based approach to assign a “trust value" to a given service on a specific computing node. The trust rating management component should be situation-aware and highly adaptable so that the dynamically changing environment such as time of the day can be taken into consideration for trust value calculation. Inside the rating management, there are two major components: trust calculation which quantify the trust level and trust propagation which enables trust calculation based on others' observations.

The trust calculation component follows a simple model that uses incentives to reward good behaviors while punishing bad ones. It provides a feedback interface for each service to evaluate the client's behavior in the transaction when the service session is completed. In addition to the service's evaluation, a client's rating is also adjusted by the server's monitoring component. The rating matrix of one node's trust value over others can be stored on one's local storage space; or the pervasive computing space repository can help cache the most recent trust rating matrix.

As trust calculation methods handles trust rating calculation based on one's own observation, the trust propagation component gathers trust information about a target node in order to evaluate its trust value even if no previous trust rating is available. This involves a complicated trust model we proposed to analyze the trust relation. More details on this topic will be discussed in Chapter 4.

To cut the detected users with insufficient trust value from further endangering the system, termination service is assigned to shut down an ongoing service session, under the circumstance that one's trustworthiness remarkably declines to an unacceptable level during the service session. The trust level drop may result from an observation of the node itself, or from a trusted node who report the target to be highly untrustable. Because the termination service shuts down ongoing services, which can seriously affects service quality if the user is innocent, it should only be summoned with high enough confidence.

In some cases complete prevention of bad behavior can be infeasible. When the system is compromised or damaged, the attack responding and repairing components are activated to minimize the loss. When the attack comes from a client, the responding and repairing components immediately shut down its service and build up a special profile for the attacker, including its conventional behavior and preferences, for future identifying the attacker again and even further lawsuit purposes. If the attack is from an ill-behaved service, the responding and repairing components may warn all distributed clients to lower the trust evaluation to prevent further loss.

To conclude our discussion on trust management framework, it is highly reliable on two critical components' functions: user recognition and trust maintenance. The former keeps track of a user’s true identity and make sure it is who it claims to be, while the latter evaluates a user’s trust value based on observations and recommendations. We are going to elaborate these two components in the next two chapters.

CHAPTER 3: USER RECOGNITION

To achieve successful trust management, the first and foremost task is to correctly recognize the users involved in the pervasive computing transactions. As we discussed before, due to the dynamic characteristic of pervasive computing systems, both the service providers and consumers can be constantly moving in and out of the computing space boundary, which may create a big challenge for recognizing the user's true identity. In addition to that, due to the fact that most communications are performed in the open air, eavesdropping becomes a viable option for the identity thieves. To make things even worse, the computing devices are usually equipped with very limited system resources such as CPU and memory, preventing them from performing complicated cryptographic calculations.

In this chapter we are going to address these challenges and elaborate our solution towards successful user recognition. We categorize the general term “user recognition" into two classes depending on whether there is pre-shared/registered information available to help achieve this task. In cases where such information is available, we refer to it as “authentication" while the term “user identification" will be used otherwise. According to this classification, technologies such as radio frequency identification (RFID) are falling in to the category of “authentication" rather than “user identification" as suggested in its name. After all RFID is more close to traditional authentication techniques compared to those abnormity based identity detection techniques.

In the following sections we are going to discuss our proposal towards recognizing users in pervasive computing systems under the situations of authentication and identification respectively.

3.1 Authentication

In the context of secure computer communications, authentication has been studied for many years since 1970s [41, 22, 58, 23, 59, 63, 30, 55, 45, 42, 75, 80, 31, 20, 44, 79, 84]. According to [58], authentication is defined as the process of verifying the identity of the communicating principals to one another. Since then use of cryptology to achieve authenticated communication in computer networks has been thoroughly studied. Protocols are proposed for the establishment of authenticated connections, for the management of authenticated mail, and for signature verification and document integrity assurance. Conventional and public-key encryption algorithms are studied and become the basis of authentication protocols. In the 1990s research on authentication reaches its climate when Kerberos v5 [60] gets published and widely adopted.

The technology of authentication seems to be full-fledged until new challenges have been posted by pervasive computing. Although modern authentication techniques provide resistance against eavesdropping over the wireless media of pervasive computing, they are too expensive to afford for some of the low-cost computing devices. For example, Kerberos requires public key cryptosystems to be available, which can be too costly for low-cost devices such as sensor network nodes [69] and RFID tags [36].

The budget problem of sensor networks is better than RFID and recent years have witnessed the trend of adopting PKI cryptosystems onto sensor nodes [56]. But RFID still can not afford such complicated authentication technologies due to its low budget: only when the price per tag can lower to \$0.10 per tag can RFID be financially accepted by the industry [77].

In this section we will first analyze the privacy and security challenges to RFID secure authentication and then present an attack model. Finally we will present our low-cost authentication protocol followed by a performance and security analysis.

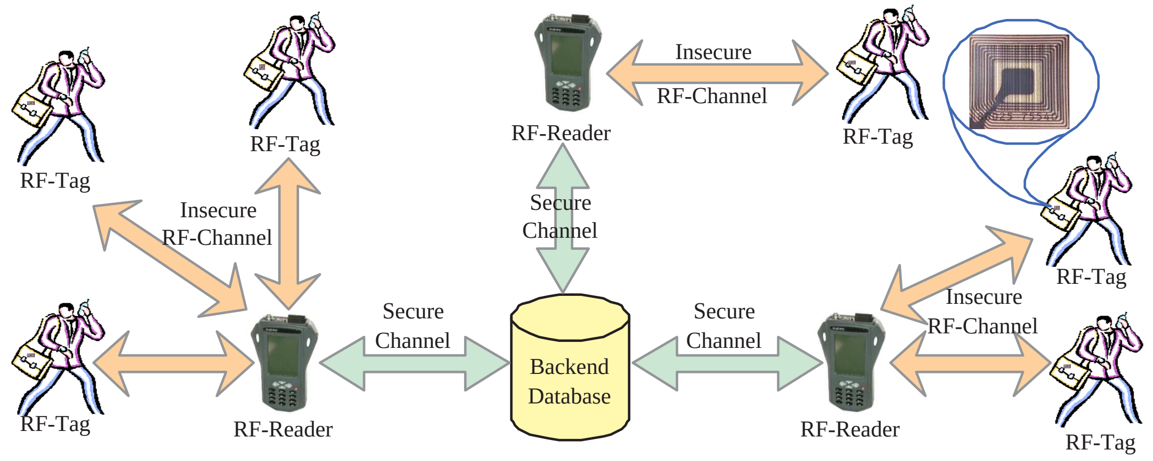


Figure 3.1: RFID System

3.1.1 Challenges and Privacy Issues

Radio Frequency Identification (RFID) technologies are widely regarded as the successor of optical bar codes and one of the forerunners of pervasive computing. Industries of manufacturing, supply chain management, and inventory control can benefit this technology to help reduce the costs wherever bar codes used to dominate. In a report released in May 2005, the US Government Accountability Office found that thirteen government agencies are using or plan to use Radio Frequency Identification tags [65].

The RFID system is a contactless recognition system composed of three parts: RFID tags, readers and a backend database (Figure 3.1). The tag (also called the transponder) consists of a passive integrated circuit and an antenna. Since it is not equipped with any power supply, the tag is quiet most of the time. Only when it receives a reader's query, the tag becomes activated and makes use of the energy absorbed from the reader's query signal to respond with this limited power. The reader is a device that sends query signal and identifies the tag from its response according to their communication protocol. The reader is usually connected to a centralized backend database running on a secure server, where all tags' information is stored. For security reasons, the data that a tag sends directly to the reader should not contain any identity information in plaintext. Only the authentic readers connected to the backend database can extract the tag ID from its response.

The communication channels between readers and the backend database can be protected by common security technologies and we will assume these channels to be safe and secure. On the other hand, the radio frequency communication channel between reader and tags is vulnerable to various attacks. Especially, because of the abundant power of the reader, the channel from reader to tag can be very easily overheard by adversaries.

The threat to user privacy is a major hurdle to the expansion of RFID industry. The US State Department is even considering backing off its RFID passport program due to the potential threat of unauthorized data access.

In many adopted RFID systems, the tag responds to the reader's query with its unique serial number, without verifying the reader's authenticity. This unique number can act as a clue for the adversaries to identify the tag carrier, thus threatening the user privacy. Due to this reason, many boycotts [1, 2, 6] are lunched against RFID technology. RF-Dump [8], a project in practice, has made the anxiety of privacy violation come true. With a normal RFID reader connected to a laptop via serial port, RF-Dump can collect tag ID without hacking anything.

Many approaches have been proposed to address the privacy issue. Most of them adopt similar schemes based on secret-sharing between tags and the database to achieve secure identity reporting. These approaches are designed to protect user privacy from various attacks such as eavesdropping. Most of them can solve the problem if there are no physical attacks performed by dedicated attackers, who can break into the tag memory by force and steal the shared secret information [61, 77]. With this derived information, the attacker can later differentiate this victim tag from thousands of tags' response so as to track the tag's carrier. Our research is motivated by the threat of physical attacks. We thoroughly analyze physical attacks to propose a secure identity reporting protocol to prevent various attacks. In the following section, we will present a threat model to identify these attacks.

3.1.1.1 Threat to Data Secrecy

The major threat to RFID system is the potential violation of data secrecy, which comes from the illegal reader's attempt to query tag identity. Technically the adversary's goal is to identify a victim tag from its peers to assist social analysis towards compromising the tag owner's privacy.

The attacks performed by the adversary can be generally divided into two phases. The first phase is the process of critical information collecting: the adversary chooses a victim tag to attack and try to collect any tag related information for later identification. During Phase II the adversary sends queries to unknown tags and collects their responses. She then analyzes the collected responses to distinguish the victim tag, with the help of information collected in Phase I. Phase I attack is usually carried out before Phase II, but not necessarily.

1) Phase I, Critical Information Collecting

During the attack preparation phase, the identity of the victim tag is known to the adversary, who is trying to break security obstacles to obtain critical information related to this specific tag. The attacker's goal in Phase I is to collect enough identifying information of the victim so as to recognize it from the wild later. According to the RFID systems architecture, we categorize the possible attacks into the following classes:

- Eavesdropping
- Message Hijacking
- Physical Attacks

Eavesdropping is the most cost-efficient and practical form of attack because the attacker only needs an RFID reader to passively listen on the communication channel between the authentic reader and a victim tag that she wants to track. The overheard message will then be analyzed so as to assist later re-recognition of the victim tag. The analysis of this message could be cryptanalysis such as off-line guessing the encryption keys (if applicable), trying to find out the relations of different responses from the same tags and etc. Since this analysis is a ciphertext-only-attack, it presents very little threat to most modern cryptographic algorithms. However, when combined with other attacks, eavesdropping may become powerful.

Most current researches assume that only the simplex channel from reader to tag is easy to be eavesdropped due to the low transmission power of tags. We believe that a robust protocol should also resist eavesdropping on the reverse channel, since the attacker may manage to achieve enough proximity to the tag in some application scenarios. For example, an attacker may pretend to check out a book at the library gateway just behind the person who she wants to track. Thus the communication between the tag on the victim's book and the library's RFID reader can be easily overheard by the attacker's device.

Additionally, the attackers can hijack the communication packets between reader and tag. They can not only eavesdrop on the message sending from the reader, but block the tag from hearing it by adding noise and generate a fake message to spoof the tag. This hijacked message may contain reader authentication information for tag management, depending on the protocol design. A special form of message hijack is record-and-replay attack, in which the attacker first blocks the tag from hearing the reader and later replays it in order to impersonate the reader. In addition, message hijack attacks can also be performed in the reverse direction (tampering the tag to reader packets). However, this attack will less affect the potential exposure of the tag's identity since only the legitimate reader is spoofed. However, we do not consider impersonating the tag is an active issue. There are two reasons for this: first, comparable attacks such as physically cloning a tag is always possible and second, for the RFID's precursor - optical bar codes, impersonating is much easier but not many complains have been heard about this issue.

Finally, a more dedicated attacker can even achieve physical access to a tag and compromise its memory. By the means of shaped charges, laser etching, iron-probe etc [70, 76], attackers can derive critical information such as identity number, authentication keys etc.

from the tag's memory. This attack was not given enough respect in the current researches. However, as we will discuss later, physical attack is a feasible option for dedicated attackers and the cost of attack is moderate.

2) Phase II, Query

During the Phase I attacks, the adversary tries to obtain critical information of the victim tag. At that time the tag's identity is known to the attacker and she wants to later identify the victim after sending it back to the wild. While collecting victim information is the main task of Phase I, Phase II is dedicated to query thousands of tags in the wild and distinguish the victim.

To achieve this goal, the attacker may install one or several fake readers at different places in the outside world, which follow the normal query protocol as legitimate readers does and collect tag responses. Then the attacker applies the previously collected information obtained in Phase I to distinguish the victim tag among thousands of tag responses.

This fake reader's query is different from eavesdropping attacks though they may share similar devices. With eavesdropping attacks, the adversary only passively listens on the communication channel between the tag and legitimate readers, while during the query the fake reader actively send messages to ask the tag to respond. Moreover, during eavesdropping attack the victim's identity is known to the attacker and in the query process, the attack has no idea if the queried tag is just the victim that she wants to track.

Since in most proposed schemes the reader's query is a general plaintext message with no authenticity information for the tag to verify, to generate a fake query is very simple and requires no attack techniques. In addition, the tag can not distinguish queries from legitimate readers or fake ones. So the attacker's query can never be prevented. To protect the tag identity from being leaked, protocols based on cryptographic algorithms should be proposed to set barrier to the attacks in Phase I.

3.1.1.2 Threat to Data Integrity

In addition to the majority attacks of Phase I and II that attempt to violate user privacy, an additional minor threat is the potential compromise of data integrity and availability, as discussed below.

The threat to data integrity comes from the adversary's attempt to tamper the data, either by hijacking the message or physically tampering the tag's memory. As X. Zhang et al. discussed in [85], we should put two aspects of integrity need into consideration: first is how well the protocol resists unauthorized modification (including physical attacks) and the second is how to detect such tampering. Theoretically, the first criteria can not be met because we can neither stop attackers hijacking the message in the open air, nor prevent people from physically analyzing tag's logic component, reverse engineering the IC chips or modifying the data in the memory. But it is possible to satisfy the second criteria and detect unauthorized modification, which is the best achievable data integrity.

3.1.1.3 Threat to Data Availability

Additionally attackers can also disable the tag to threaten the data availability. The simplest way to do that is to use a Blocker tag [38] to temporarily block the tags in a small area. A more destructive attacker can even use an electromagnetic weapon to physically damage a tag [36]. This Denial of Service attacks is a general problem to all cryptographic protocols and the discussion of preventing such attacks is beyond the scope of this paper.

3.1.2 A Secure Identity Reporting Protocol

In this section, we present a thorough analysis of physical attacks which motivates our research. Then we propose a secure identity reporting protocol to address this problem, together with other threats discussed in the previous section.

3.1.2.1 Physical Attacks Analysis: Motivation

1) Attack Scenario

Hereby we use this RFID scenario as an example: Bob wants to know what his friend Alice is doing everyday. Bob knows that recently Alice has borrowed an RFID embedded handbook from the local library. So he installed a few RFID readers in places where Alice might show up, since he knows the handbook is always carried along with Alice. A couple of weeks later, Alice returns the book to the library and Bob, of course, becomes its next “reader". He uses special devices to access the tag memory and successfully digs out the secret key of this tag. With the help of data collected by his RFID readers and this secret key, Bob can filter the data that his readers have collected over these days and clearly see where Alice has been. When she gets up and left home, how often she goes out for shopping, when and how long she takes part in a club etc. Alice’s private life is under surveillance by the RFID book she carries. If this attack is not severe enough, critical articles such as RFID tagged banknotes [37, 15, 83, 85] can be tracked in the same way to help criminals committing a robbery.

2) Attack Feasibility

According to the attack scenario, we define physical attacks in RFID systems as follows: attacks that use special devices to read tag memory and obtain/change identification-related information so as to compromise identity reporting protocols. Though attackers may also take advantages of these devices to physically access other item-related information stored in tag memory such as product price, book title, check out time, etc. that is not the physical attacks we are discussing about.

Generally speaking, all embedded systems without occasional human surveillance should give consideration to physical attacks. In addition, due to the price limit, RFID tags can afford no special provisions for secure storage areas where secrets are kept, which makes physical attacks even easier.

Physical attacks have been seen on various median to small sized IC chips such as USB tokens [40] and smartcards [17]. Up till now, no physical attacks on RFID chips have been published. However, since there is no significant difference between the IC design of RFID tags and smartcards at the chip level, similar attacks performed in [17] can be deployed on the RFID tags. For example, the attacker can also use fuming nitric acid to remove the resin coat of the tag, disconnect the microprocessor and attach one single micro-probing needle to the data bus. Then providing the address signal on the address bus, the secret EEPROM content can be accessed.

In many RFID applications, the tags can be temporarily possessed by the adversaries, e.g. in RFID library systems. Then the tags can be physically attacked and returned to the legitimate owner. This physical attack can be carefully performed so that the legitimate owner will not notice that it has been attacked. Even if the attacker has to damage a tag in order to access it memory, she can clone one since the data in tag memory is known.

Another simple but efficient way to achieve tracking is to attach a new tag to the victim other than "wasting time" to perform physical attacks to the legitimate tag. However, attaching a new tag to the item and returning it to the legitimate user plays a risk of being detected, especially when the item that carries tags are small and in "clear" form, e.g. bank notes. Correspondingly, physical attack can be used to track the victim without leaving evidence. So physical attack is still a safe option for the attackers, though the attacker has to pay higher cost.

From the discussion above, we can see that physical attack is a feasible attack with moderate cost. It has the potential to compromise user privacy in RFID systems. Although no physical attack incidents have been reported, as this technology's fast and widely deployment in, profitable gains will finally attract attackers' attention to put it into practice in the future.

3) Countermeasures against Physical Attacks

Generally there are two research directions towards preventions of physical attacks. One is hardware approach, applying tamper resistance property [12] to the system; the other is software approach, focusing on cryptographic ways to solve the problem.

Applying tamper resistance property to the tag is an effective way to prevent physical attacks. However, due to the current price ceiling of \$0.10 per tag [77], tamper resistance is too luxurious to be deployed on RFID tags [61].

To resist physical attacks by the means of software, an important rule is that no authentication keys should be shared between tag and the backend database, because the secret key in tag memory is susceptible to physical attacks and can be utilized to compromise the protocol and track the victim. However, some necessary information needs to be shared for the backend database to identify the tag. We focus on minimizing the impact of physical attacks on the shared secret to prevent long term tracking.

4) Resistance Classification

To better evaluate the system protection over physical attacks, we classify RFID identity reporting protocol's resistance into three levels:

- Level I (insecure): Adversaries can perform physical attacks once and track the victim tag forever. If we set the time of physical attack as a time line, either the queried data before or after this time can be used to track the victim tag.

- Level II (backward secure): Through physical attacks, adversaries can not trace the data back through past events in which the tag was involved before physical attacks. But they can still use the secret information, which was compromised by physical attacks, to pass subsequent authentication and track the tags afterwards.

- Level III (secure): Adversaries can not compromise the identity reporting protocol through physical attacks.

Level III is the most secure, but Level II resistance is enough in some practical scenarios. For example, in our scenario, since Bob can not predict which book Alice will borrow from the library, he has to collect the tag's response first and later physically attack the tag when Alice has returned the book. In such application scenarios, backward security is acceptable.

Physical attacks are not a concern in many existing protocols that follows shared secret schemes, thus their resistant lies in Level I in our categorization. However, a simple but efficient improvement can level up their resistance. In most of these protocols some secret information (e.g. an encryption key) is shared between reader and each tag for authentication purposes but it is never changed. To prevent physical attackers from "hacking once, tracking forever", the tag can periodically ask the legitimate reader to refresh the secret information (e.g. update the key) after the reader is authenticated. This scheme is similar to some web system periodically asks its users to change password. In this way the resistance to physical attacks can be enhanced. Though this does not solve the physical attacks problem, such an information refreshing scheme is useful in scenarios where tags are frequently reused, e.g. in library RFID systems.

3.1.2.2 A Secure Identity Reporting Protocol

In this section we will propose a secure identity reporting protocol that can resist various attacks such as eavesdropping, message hijacking and physical attacks. We will propose the protocol and analyze its performance. The security analysis of the protocol will be presented in the next section.

As discussed above, to design a secure RFID identity report protocol, a few hardware limitations should be considered.

- Only cheap hardware implementations can be afforded by RFID tags. The price per tag limits the number of logic gates for security purpose to a few thousands, which can only support cheap functions such as one way hash. Though recent hardware development implements AES into low-cost RFID tags [29], public key cipher, e.g. RSA, is not an affordable option for a low-cost tag for the IC industry in the foreseeable future.

- The communication channel between backend database and the reader is secure from eavesdropping. The backend database is secured by authentication and authorization

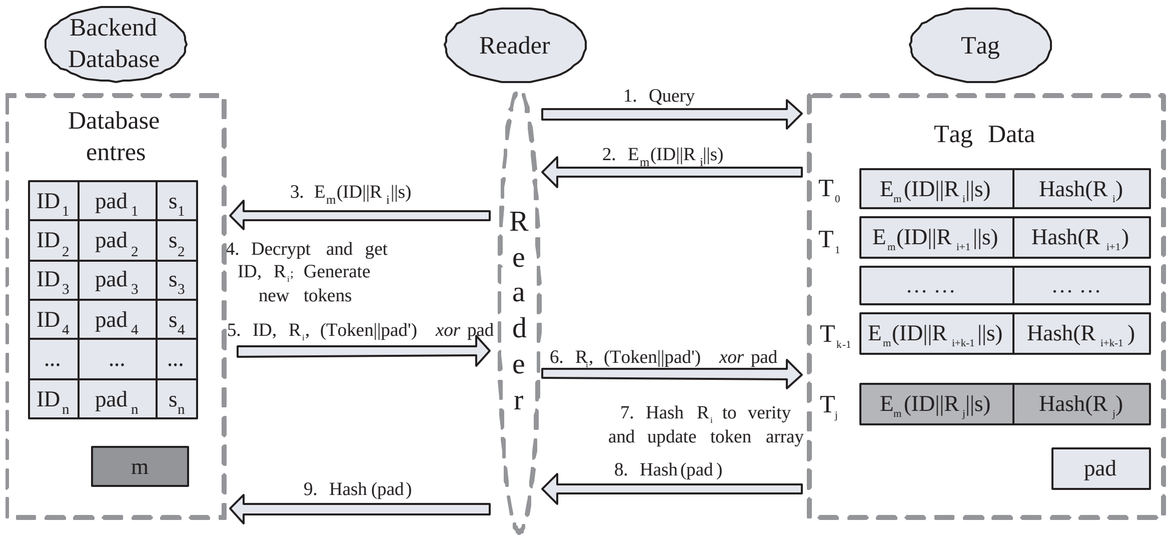

Backend Database
Reader
Tag
Database entres
ID1 pad1 s1
ID2 pad2 s2
ID3 pad3 s3
ID4 pad4 s4
IDn padn sn
m
1. Query
2. Em(ID||Ri||s)
3. Em(ID||Ri||s)
4. Decrypt and get ID, Ri; Generate new tokens
5. ID, Ri, (Token||pad') xor pad
6. Ri, (Token||pad') xor pad
7. Hash Ri to verity and update token array
8. Hash (pad)
9. Hash (pad)
Reader
Tag Data
T0 Em(ID||Ri||s) Hash(Ri)
T1 Em(ID||Ri+1||s) Hash(Ri+1)
Tk-1 Em(ID||Ri+k-1||s) Hash(Ri+k-1)
Tj Em(ID||Rj||s) Hash(Rj)
pad

Figure 3.2: A Secure Identity Report Protocol

technologies so that only authentic readers can request the database to extract the tag identity.

Besides these hardware limitations, several designing principles should be followed to propose a secure identity reporting protocol. First of all, the tag's response to the reader should be ideally randomized. That is, no one should be able to tell whether two responses are from the same tag or not. Secondly, the protocol should scale with large number of tags in a system. For example, if the tag simply hashes its ID together with a random number and sends it back to the reader, the backend database has to exhaustively search every entry to find a match, which is not scalable to a large number of tags. Finally, after the reader successfully located the tag's data from its backend database, it should convince the tag of its authenticity to further manage the tag. A secure way for the reader to prove its authenticity to the tag should be developed. In this section we will propose a protocol concerning about these principles and the hardware limitations.

1) Initial Setup

Our proposed protocol relies upon a few tokens ($(T_0 ... T_{k-1})$) pre-stored in tag memory during setup (Figure 3.2). Each token is composed of two parts. The first part is the tag *ID*, a random number R_i and the token serial number s, encrypted by the backend database with its master key m. The second part is the one way hash of R_i. During setup, the backend database (the only party that knows m) generates k tokens $(T_0 ... T_{k-1})$ for each tag.

Besides the k normal tokens, one extra token T_j is generated the same way and stored on the tag. This extra token is for second channel recovery only and will not be sent out when queried. Besides, each tag shares the value pad, s with the database, which is used as a one time pad to mask the token update. The backend database consists of entries of each tag's ID and the corresponding pad, s, as well as other information related to each tag carrier, such as product name, producing date, last check out time, etc., depending on the application requirement (optional). The shared value pad is used as a one time pad to mask the token update, while s is the token serial number used to synchronize the token version to prevent tampering and many other attacks. The function of s will be discussed in detail in Section 3.1.3.2.

2) Protocol Execution

Same as most existing protocols, our protocol starts with the reader's plaintext query to tag. On receiving the query signal, the tag sends its first token T_0 back to the reader. When it receives the tag's response, the reader forward the token to the backend database and request a token decryption. The backend database verifies the authenticity of reader with normal authentication techniques and generates k new tokens using its master key m and a new pad' for token update. It then expand the existing pad to the length of $token||pad'$ and XOR it with them. This encrypted token update will be returned to the reader together with the extracted ID and R_i. The reader removes ID in the message and forwards the rest of message to the tag.

On receiving the token update, the tag will first check the validity of this message by hashing the authenticator R_i. If the hash of returned R_i equals to the corresponding value in token T_0, the tag will decrypt the token updates and place these new tokens as well as new pad' into its memory. Otherwise this message is discarded. To acknowledge the database that the pad is successfully updated, the tag sends the hash of pad back to complete the protocol.

3) Exception Recovery

If all the tokens $T_0 ... T_{k-1}$ are consumed by unsuccessful query with no valid token updates received, the tag will respond a message “main channel halt" to further query and the second channel becomes active. The second channel could be a physical contact interface [72] or a radio frequency channel with a greatly attenuated power that can only be heard within a few centimeters.

It is reasonable to assume that the attacker can not track the tag within such proximity that she can almost contact the tag. So the second channel can safely respond every query with the same token T_j (See the shadow part in Figure 3.2) without being afraid to be tracked by this token. The tag then waits until a reader send back the right authenticator R_j for the token T_j and recharge the tag with plenty of new tokens, following the standard identity reporting protocol in Figure 3.2. Through this second channel replenishment, the tag gets recharged again.

3.1.3 Performance and Security Analysis

3.1.3.1 Performance Study

The performance study in this section will show that our proposed protocol has light-weighted hardware complexity and good scalability, but with lower system reliability.

The first criterion to evaluate the protocol performance is the hardware complexity of the tag, which can seriously affect the price per tag and thus its practicability. In our proposed protocol, only two computational functions: one way hash and exclusive-OR, are implemented in the tag. The entire encryption/decryption burden is left for the backend database to provide the secrecy of tag ID. The tag follows very simple logic to perform transmitting/receiving message, hashing and updating memory. Though some redundant bits are used to store multiple tokens, the hardware complexity is moderate and the price per tag still remains at a low level.

Secondly for the criterion of protocol complexity, our approach adds three more communication rounds (message 6, 8 and 9) to most existing schemes. More communication round may suggest lower system reliability, or specifically, lower successful rates of identity reporting. Further implementation research is requisite to evaluate the degree of system reliability degradation.

For the scalability criterion, in our approach the computational complexity on the database side is O(1), as our database always does only one decryption per authentication to derive the value of ID. Compared with the O(log(n)) complexity in [57] and O(n) complexity in [61, 66, 77] (where n is the number of tags within a system), our protocol is scalable with the gigantic size of RFID system in practice.

Another benefit of our approach can make its performance more attractive. Though not economical, some RFID systems require item-related information (e.g. product name, last check-out time) stored on tag other than in the centralized database. It is not a good practice to store such information in plaintext because the attacker can physically attack the tag and access this data. In our approach, this item-related information can be stored in cipher text within each token, (i.e. $E_m(ID||R_i||s||ItemInfo)$) which can provide confidentiality of the data.

3.1.3.2 Security Analysis

In this section we will evaluate the resistance of possible attacks of the proposed protocol according to our threat model. As we discussed, the potential attacks that may threaten user privacy are eavesdropping, message hijacking and physical attacks. In addition the attacker can also compromise the data integrity and availability by other means such as denial of service attacks.

1) Resistance against Physical Attacks

The main advantage of our protocol is its resistance to physical attacks. In our scheme, the only data stored in tag memory is the array of tokens and the *pad*. We assume Carol, the intruder, can compromise the tokens and use them to analyze the response from tag.

For the first part of token: $E_m(ID||R_i||s)$, provided the encryption algorithm E is secure (e.g. AES-128), it is infeasible for her to get either ID or R_i without knowing the master key m. Thus she is not able to track the victim by extracting ID. However, Carol can memorize all the k tokens and send query to see if these tokens returned back so as to track it for a maximum of k times. If the tag meets legitimate readers before been tracked k times, all tokens will be refreshed and Carol will lost the track of victim tag. We call this fruitless attack “physical read tracking", which is totally different from regular physical attacks that compromise the authentication keys once and track the tag forever.

The above discussion is based on the premise that Carol can only read tag memory to compromise user privacy. If Carol can tamper the token content (i.e. attack against the data integrity), she may generate fake tokens to replace the existing ones in order to gain completely control over the victim tag. As we discussed in Section 3.1.1, this tampering attack could not be prevented under the assumption that physical attack is possible. The best resistance to tampering attack is to detect it. In our protocol, the backend database can not derive a valid ID by decrypting the fake token generated by the tempering attackers so that the attack can be detected when the tag meets the authentic reader.

2) Eavesdropping Attacks

In our proposed protocol, message 6 is a meaningful reader to tag message, which can be eavesdropped by the attackers. This message is composed of two parts, an authenticator R_i and the padded token update. Because the token update is encrypted by a one time *pad*, we assume the eavesdropper can not remove the pad to derive the token update as long as the *pad* is secret.

However, according to our security assumption, the *pad* on tag memory can be compromised by physical attacks, with a higher cost. Thus physical attacks combined with eavesdropping can successfully track a victim because subsequent *pad*s can be decrypted by the current *pad* thus forms a vicious circle. Though this attack the adversary can track the victim for $k-1$ times per eavesdropping (she has to leave at least one token available for the legitimate reader to query). However, this combined attack has a very high premise that the attacker has to eavesdrop on every single session when the tag communicates with any legitimate reader. Once the attacker missed one communication session, she will lose the trail of *pad* thus lose the track of the tag, which can make the physical attack very fruitless. Moreover, another active way to defeat this combined attack is that the database can track the number of remaining tokens on each tag. If one tag shows abnormally small number of remaining tokens, the database will send a warning to the reader and ask it to refresh the tokens in a more eavesdropping-resistant way, such as using the second channel.

3) Message Hijacking

As discussed in our threat model, a more dedicated eavesdropper can hijack message 6 by replacing the token updates with a fake one. Since the authenticator R_i in the hijacked message can pass the validity check, the tag will update the fake tokens according to the hijacked message. This message hijacking attack can successfully compromise the data integrity, but it brings very little threat to user privacy. Without knowing the current *pad*, the attacker can not determine the token value after the hijacked message is padded with *pad*, thus she is not able to track the tag.

A more powerful attacker who is able to obtain the *pad* by physical attacks can strengthen the power of message hijacking. She can hijack the token update message and replace the new tokens with exactly the same existing tokens in the tag (i.e. prevent the tag from updating tokens). Later when her fake readers find a tag sending T_0 to respond query, the attacker knows it is from her victim. She will again prevent the tag from updating tokens in the same way since the authenticator R_i for T_0 is already known to her. Thus the attacker can circularly track the victim.

However, this combined attack also has a high premise that the attacker has to hijack the token update message in every single session when the tag communicates with any legitimate readers. Otherwise she will lose the control of the victim tag, until another physical attack is performed again to obtain the *pad*. This can make the combined attack fruitless.

As we discussed in Section 3.1.1, the best resistance to tampering attacks is to detect it because we can neither prevent attackers physically tampering the tag memory nor stop them hijacking a message in the air. The discussed combined attack to our protocol can be detected by tracking the token serial number s in each token. The token serial number s is a growing version of the token and the s of last successful authentication is stored together with each tag's ID in the database. When the backend database detects a tag returns one token with an incorrect serial number s, it can inform the reader that the tag has been attacked. In that case the reader can take actions to prevent further tracking by communicating with the tag in a hijacking-resistant way (e.g. using the second channel).

4) Denial of Service Attacks

One potential threat to our token based scheme is the denial of service attacks: an adversary can send a mass of queries to quickly consume the tokens. Although user privacy is not compromised, this process can disable tags from responding, thus threatening the data integrity.

This denial of service attack is very similar to the DoS attacks discussed by Ari Juels in [36]. In [36] Ari argues that DoS is not considered as an active issue because there exist other simple but effective physical ways to achieve comparable DoS attacks. For example, an adversary with an electromagnetic weapon need not resort to breaking down the protocol in order to disable tags, especially if the tags are protected by some countermeasures of unintentional query, as will be explained in the next section.

Though DoS attack is not an active issue, RF-tags can be *unintentionally* scanned by readers that are not associated with their designated reader. (E.g. a reader of Company A may inadvertently read tags of Company B.) This unintentional query, though not belonging to any forms of attacks, has the same effect as DoS attacks. In our proposed approach, we resort to other schemes to minimize the impact of unintentional query, as discussed below.

One possible solution to unintentional query is to use different query message for different association. For example, in airport luggage flow control systems, different airline companies can query tags with their own query message. In this way a tag will never respond to unintentional query from unknown readers of other companies.

This scheme of association-specific query fulfills the privacy requirement in some application scenarios, but may have potential threat in some other scenarios. For instance, a person carrying 3 books of library A, 2 books of library B and 4 books from library C may suffer from probabilistic tracking: the attacker can send query of library A, B and C to see if another person also return 3, 2, 4 responses of the three libraries respectively. If such responds are found, the attacker can conclude with high confidence that this person is the victim that she wants to track. More detailed discussion of this “set of tags characterizing a person" has been highlighted in [18].

If the unintentional query is inevitable, we should provide means to recover the tag. As we discussed Section 3.1.2.2, the second channel is used to recharge a tag that died from unintentional query. When the tag is running out of token, it will respond “main channel halt" in the main channel and opens the second channel as an interface to recharge the tag with new tokens.

To conclude this section, we presented our threat model of RFID systems and thorough analysis of physical attacks to RFID tags. Based on the threat model we propose a secure identity reporting protocol to address the possible attacks. In our proposed protocol, the tag responds to readers with pre-stored one-time tokens. The tokens contain the tag’s encrypted ID that can only be decrypted by a legitimate reader. The reader sends back dynamically created new tokens to the tag. The new tokens are encrypted by a one-time pad, which is also dynamically updated by the reader. We analyze that our scheme can resist physical attacks, in addition to other security attacks. The performance analysis shows that our protocol is scalable for large RFID systems with huge volume of tags.

3.2 Identification

In the previous section, we discussed authentication techniques and dive into RFID for a secure authentication solution. In this section we are going to cover identification, where no pre-shared information is available for the system to verify the identity of a user.

As we discussed in Chapter 1, dynamic population in pervasive computing system may suggest that the infrastructure has to be friendly to new users [33]. The service providers should put basic initial trust on a stranger client, who might turn out to be a malicious attacker later. This will encourage attackers whose fame (in terms of trust level) in the community has gone notorious to come back and claim themselves as new users, which wipes bad reputation to neutral very easily.

In addition to that, inside threat [34, 71] coming from compromised nodes and malicious outsider attackers can compromise the user identity by controlling the user's computing devices. Especially if the compromised nodes had a high trust level within the computing space before the attack, compromising user identity can result in serious chaos the trust management system.

Besides the challenges, pervasive computing systems also have some nice features that can help us to achieve trusted coalition. For example, there are usually some powerful nodes (compared to ad hoc networks) that can help to perform distributed management and broadcast the result to benefit the community. Some pervasive computing systems may even have some centralized control services that can be trusted by other nodes. Also the population of the pervasive computing system is relatively low than Internet, which make it easier to achieve profile management of different groups of users.

Take this application scenario as an example. Two famous bank robbers want to steal some valuable jewelry that is secured in the bank. First they want to take a look inside the bank to assist their next action. So they just claim themselves as new bank customers and walks into the bank. Their PDAs then start to communicate with the computing nodes inside the bank lobby via the pervasive computing environments provided to assist bank customers. By talking with the other nodes, their PDAs may collect some important information such as how video cameras are installed in the system, where is the blind point of the infrared sensor, when is the best time to take the action, etc. But because the robbers will not do anything harmful to the bank at this moment, and their true identities as famous robbers are hidden behind the “new users", the bank security system will not take special notice upon them.

As another attack scenario, the robbers may try to hijack one bank manager’s identity (e.g. stealing her ID card, threatening her family, etc.) and easily find and steal the valuables by taking advantage of the fake identity. In both scenarios the failure to detect and prevent the robbers are due to the failure of identity recognition of the robbers. Thus the trust management system that works upon the failed identification system can not protect the bank property. Traditional intrusion detection techniques will not be able to solve the problem since the attackers have not started the process of robbing the bank yet.

In this section, we assume that user identity is already compromised by either of the two methods above (i.e. stealing identity or claiming as new user). We are going to propose our user behavior identification schemes based on behavioral matching and profile management systems to identify those identity violations and fire an early warning. Our user behavior identification approach can be roughly divided into three correlated parts: system logging/monitoring, user behavior profile establishment/maintenance, and malicious user detection. These modules will be built within the trust supportive framework [49] among pervasive computing nodes.

3.2.1 User Behavior Identification and Profile Establishment

As we stated before, we will assume that user identity is already compromised by stealing identity (e.g. controlling the victim nodes) or simply claiming as new users. As shown in Figure 3.3, our proposed user behavior identification approach can be roughly divided into three correlated parts: system logging/monitoring, user behavior profile establishment/maintenance, and malicious user detection. In order to identify these identity

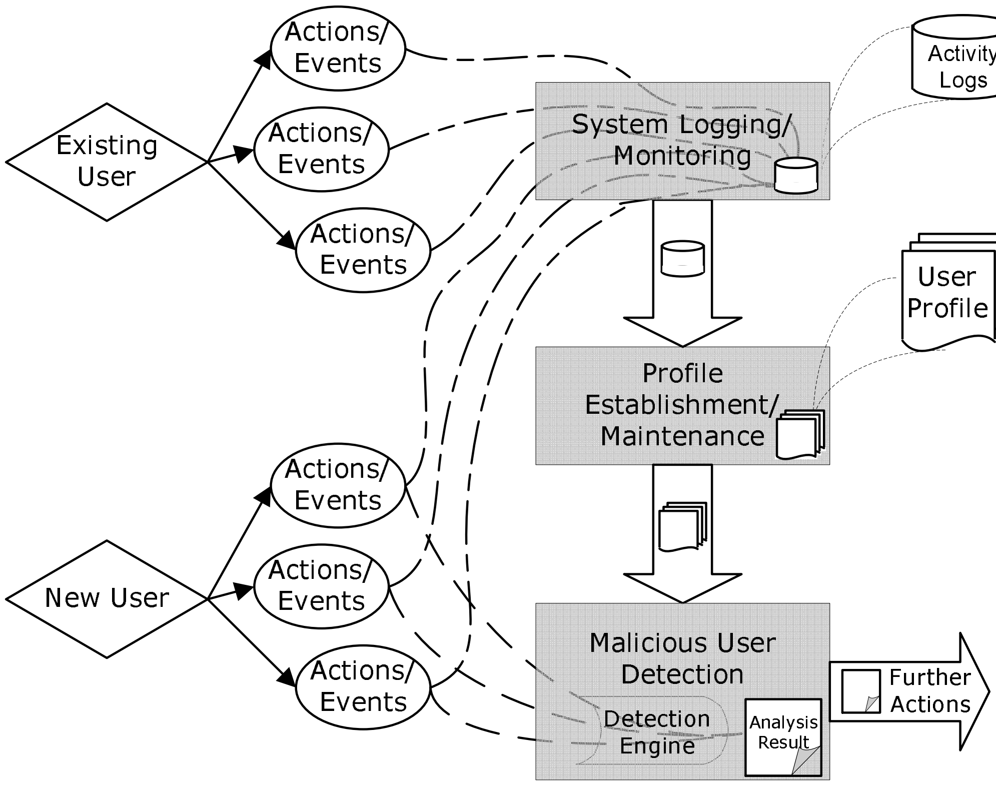


Figure 3.3: User Identification Approach Overview

threats, the first task is to gather information related to user identities.

3.2.1.1 System Logging and Monitoring

The system logging and monitoring module is responsible to collect information of daily user actions and events. This can be roughly divided into two parts: logging normal daily system events and monitoring potential malicious actions.

1) Logging System Events

Traditionally system logs automatically record events in a certain scope in order to provide an audit trail that can be used to diagnose problems. Different operating systems and multitudinous computer programs include different forms of logging subsystem. Some operating systems provide a syslog service [9], which allows the filtering and recording of log messages to be performed by a separate dedicated subsystem.

In a pervasive computing space, a computing node may also perform similar functions to track down system activities and events generated by different users. Since these logs will be utilized to analyze user identities, they will focus on taking information related to various user behavior signatures as inputs, such as:

- User login events. Different users may have discrepancy on their active time period within a pervasive computing space. So their login and logout to the pervasive computing systems can serve as important log information for further identification purposes. Information such as their login time of a day, login frequencies will be recorded into the logs.

- System resource utilization. Users will most likely try to access services that they are interested in. For example, in our scenario the robbers may frequently access the digital camera while normal bank customers are barely interested in that.

- Networking signatures. The networking traffic is also specific that can help identify a user or even a group of users. For example, a user with high packet internet groper(PING) value is very likely to have laggy network connection next time. Besides, the traffic style such as transport protocol one is using and the preferred access point one usually connects to can also be signatures of specific users.

- Other signatures. In addition to the signature information above, there is also other information that can be logged, such as which operating system one is using, which antivirus software, web browser one has installed, which email service one is connecting through, which news group one is actively interacting with.

The inputs types of system logs should not be static. Rather, they should be dynamically tuned according to the application scenario. For example, in our scenario the system resource utilization collects more behavior signatures than the networking signatures. If one type of logs is considered more useful within a system, its corresponding information collection module should be allowed to use more system resources to get more detailed data.

2) Monitoring Potential Malicious Actions

As we discussed, the system logging is responsible to collect user identity related information though logging normal system events generated by daily user actions. In addition, if a user behaves abnormally/maliciously, the system should also take it down since the way in which a group of attackers try to attack the system could be idiographic than others.

Generally the monitoring of malicious actions can be traced by traditional intrusion detection systems (IDS), such as Snort [10, 67]. But the traditional IDS acts like a black box, which takes user action/events inputs and gives out intrusion warnings. In order to profile similar attackers when they come back in the future, we also need to record the information of how one is performing attacks. This could be done by exploring into detail of the IDS black box and extracting useful information to identify an attack.

With the detailed daily system logs and the attacking information collected by the dedicated IDS, the information collection layer will pass down the collected data to the profile management layer for further process.

3.2.1.2 Profile Establishment and Management

With the profile information collected by the system logging and monitoring components, the profile establishment and management module is responsible to parse them and build up profiles for malicious users.

The process of establishing user profile is shown in Figure 3.4. As we discussed, the pervasive computing service with a login interface usually will also allow users to register for a new account and earn some initial trust to use the service. So if the information collected belongs to an existing user, the profile management system will directly find the profile that it belongs to and update corresponding data. If the user claims itself as a new user, the profile maintenance module will first create a new entry in the profile database for this user. But when the malicious user detection module recognizes that the user profile signatures matches existing malicious user behaviors, it will send a feedback to the profile management module and request a profile merge.

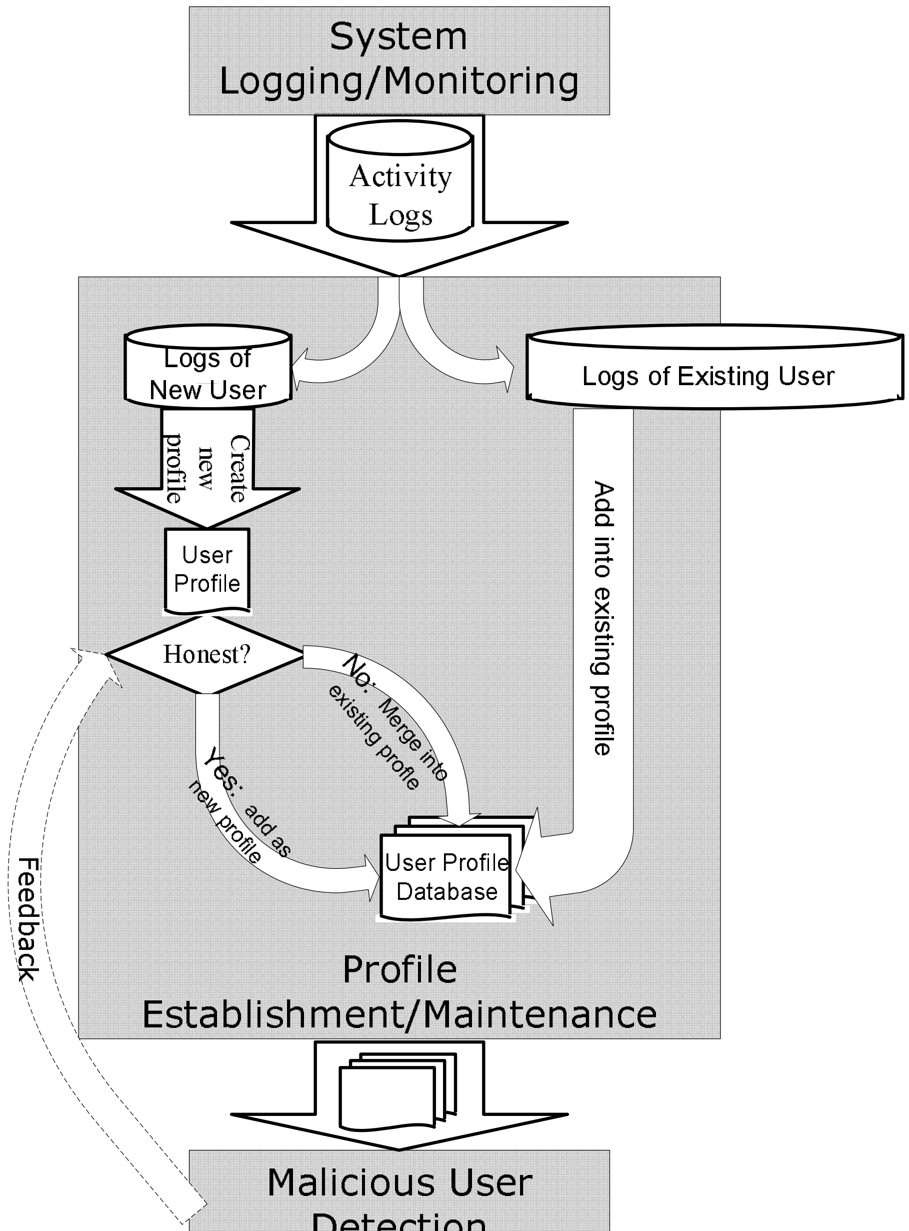


Figure 3.4: Profile Establishment and Management

There are some on-going researches on user behavior modeling towards establishing user profiles. Nathan and Sandy at MIT's Media Lab proposed a system called Eigenbehaviors to use limited user behavior information to predict the daily behavior and determine the social allegiances of study participants [26]. Our user behavior modeling will follow the major trend of current researches and model the user behavior at the data attribute level. A user's profile will include the distribution of each behavioral data type, e.g. login time, system resource utilization.

Although each user may have its own profile in the database, it is neither infeasible nor necessary to match an unknown user to one specific user's profile in the database, especially if the user pool is very large. Instead, the user profile management system will classify similar malicious user profiles and treat profiles of the same category as a whole for signature matching. For example, in our scenario the bank robbers are classified in one profile. Their system resources utilization and the critical information that they are interested can be their behavior signatures. So instead of identifying another user as one specific former attacker, the system can simply identify the user that is similar to this attacker category, which is more feasible and practical.

Though the logic of profile management system is simple, the inside management of profile database is tricky. First, the user signature information of the same type may have conflict with each other. For example, one attacker logs in twice, once with an 80% connection of HTTP protocol but the other time only 10% is HTTP data packets. To make the profile more accurate, the system should record more samples of same type of information in a long run. If the data is consistent for a long while, the piece of inconsistent data will be simply ignored; but if the data is varying from time to time, this information type will be not very useful for recognizing a user and will be given a low weight for the final user identification evaluation calculation. Moreover, the profile database is growing as time goes on. It is necessary to keep it in a reasonable size by regularly deleting some out-of-date profiles.

3.2.1.3 Suspicious User Detection

When the profile database is well updated and has collected enough user profile information, the suspicious user detection module is going to identify suspicious users by matching their actions to existing malicious user profiles, which was obtained from IDS, or the training data with specified attacker identities. This matching process involves pattern matching techniques as is used in most intrusion detection systems, composed of a series of algorithms. As a simple example, the pattern matching process can use the following two different algorithms to model the matching process.

Because each user profile contains various types of information, such as user login events, network usage signatures, frequently used system resources etc., there should be different matching algorithms for different data types. Here we introduce Matching Score (MS) to quantify this likelihood. A Matching Score is a real number between 0 and 1. The closer MS is to 0, the more likely the two compared users are of the same identity. MS of different data types should be calculated in different methods:

For quantitative data types, such as the network traffic composition, the matching score will be calculated as in Equation 3.1. Here MS_i stands for the matching score of this data type. P_i means the profile data value of the existing malicious user group and N_i is the data value of the current new user. Max_i and Min_i stands for the possible maximum and minimum value of this data type. For example, one existing malicious user group has a signature of 90% network traffic to be UDP (e.g. due to some special attacking technique they use). And the current user under test has only 5% UDP in total traffic. We know the possible value of this data type varies from 0% to 100%, so the matching score is calculated as $\frac{|90\%-5\%|}{100\%-0\%}$=0.85, which means this new user has a 0.85 *unlikelihood* to belong to the specified malicious user group in terms of this data type.

$$MS_i = \frac{|P_i - N_i|}{Max_i - Min_i} \tag{3.1}$$

For discrete data types, such as preferred system services that the user is accessing, the matching score is calculated as in Equation 3.2. In this equation, Eq(a,b) is a evaluation function of two discrete data: if a and b has the same value, Eq(a,b)=0; otherwise Eq(a,b)=1. So the output of $Eq(P_i,N_i)$ reflects the unlikelihood that the user's action signature (N_i) is same with the matched profile signature (P_i). However, since this data type is discrete, the possibility that N_i happened to be the same as P_i is not neglectable. So we adjust the matching score by adding the probability that the two to be the same by chance, which is $\frac{1}{Total(P_i)}$, where $Total(P_i)$ is the total number of possible values of this discrete data type. But since the matching score should be a value between 0 and 1, we refine this result by using a Min() function to make the result conform to the MS range. For example, one existing malicious user is using Linux operating systems and the user under test is using Mac OS X. So here $Eq(P_i,N_i)$=1 since they have different values. Suppose the matching system will recognize Windows, Linux, Mac OS, BSD, Solaris, and other unix-like OS as a whole. So there will be 6 possible values of the adjustment part in total: $\frac{1}{Total(P_i)}$=$\frac{1}{6}$. Finally we do the Min function and derive the matching score of this data type: $MS_i = Min(1, 1+\frac{1}{6})$=1.

$$MS_i = Min(1, Eq(P_i,N_i) + \frac{1}{Total(P_i)}) \tag{3.2}$$

We have discussed how to calculate the matching score of a new user's signature on one data type (MS_i). Usually there are multiple types of signatures in a user's profile. A method is needed to calculate the final matching score combining all these profile signatures. Because different signature types have different importance in terms of identifying a user, we assign each signature type a weight (W_i) to reflect its impact. The final matching score of one user identity's similarity to a given user profile is calculated as Equation 3.3. Here MS_i stands for the result of matching score calculated in data signature type i. Since each MS_i is in the range of 0..1, the result of MS will also have to be a real number between 0 and 1. The closer MS is to 0, the more likely the testified user has the same identity with the target user profile, and vice versa.

$$MS = \frac{\sum MS_i * W_i}{\sum W_i} \tag{3.3}$$

The system will have a threshold value (e.g. 0.2) of acceptable MS value. When a new user comes in, the user identification system will try to analyze its behavior signatures according to the active attacker profiles (one by one) in its profile database. If the user signature matches any of the attacker profiles, it will be blocked from further services and its profile database will be updated for better identifying this user in the future.

3.2.2 Simulations and Performance Studies

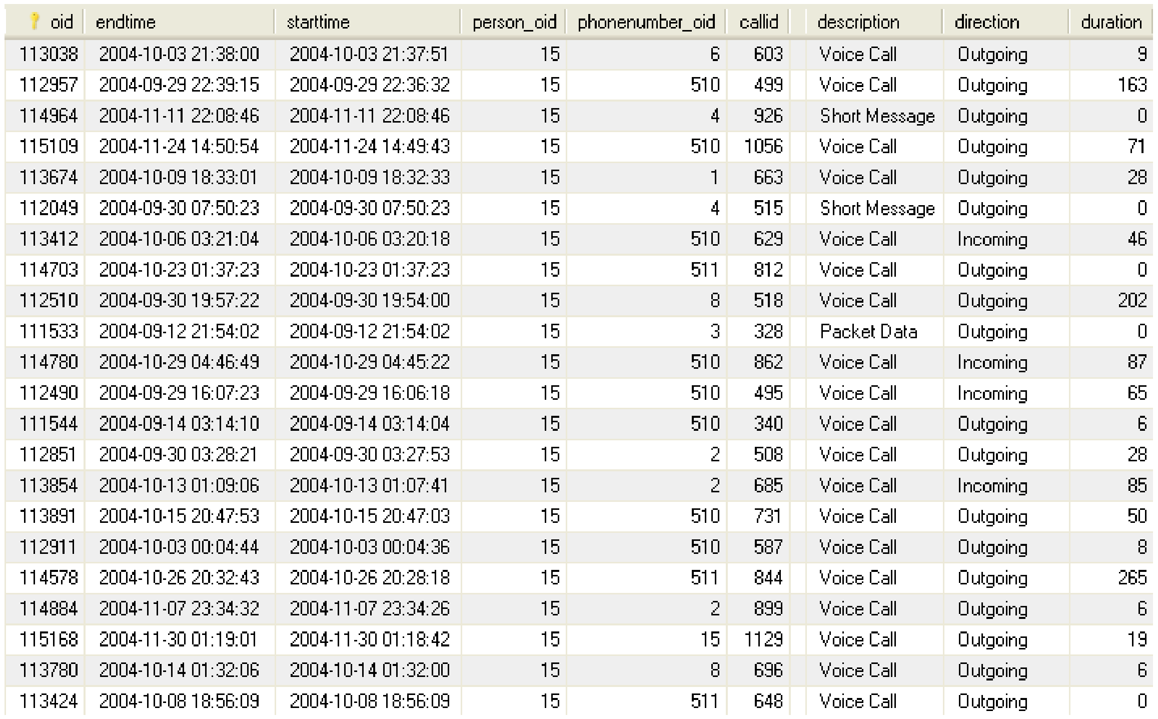
To test the performance of our behavior based user recognition model, we conducted a series of simulations. The simulation is done based upon a dataset collected in real world applications. Ideally, the dataset would cover a middle-sized user base over a large time period, with rich user behavior related context data. Should users and their behavior be recorded by the dataset, we can testify if our user recognition model can successfully recognize a specific user from their profiles established following our model.

3.2.2.1 Simulation Setup

In order to evaluate the performance of our user identity recognition model, we utilized a trace of node contacts from the MIT Reality Mining project [7, 27, 28].

The Reality Mining project represents the largest mobile phone experiment ever attempted in academia, which collected an unprecedented amount of data on human behavior and group interactions. The data is anonymous and made available to the general academic community. By the end of the experiment, this dataset contains over 500,000 hours (60 years) of continuous data on daily human behavior. Figure 3.5 shows a partial and sample view of the data collected by Reality Mining project.

The dataset consists of one hundred Nokia-6600 smart phones pre-installed with several pieces of software they have developed as well as a version of the context application which



oid	endtime	starttime	person_oid	phonenumber_oid	callid	description	direction	duration
113038	2004-10-03 21:38:00	2004-10-03 21:37:51	15	6	603	Voice Call	Outgoing	9
112957	2004-09-29 22:39:15	2004-09-29 22:36:32	15	510	499	Voice Call	Outgoing	163
114964	2004-11-11 22:08:46	2004-11-11 22:08:46	15	4	926	Short Message	Outgoing	0
115109	2004-11-24 14:50:54	2004-11-24 14:49:43	15	510	1056	Voice Call	Outgoing	71
113674	2004-10-09 18:33:01	2004-10-09 18:32:33	15	1	663	Voice Call	Outgoing	28
112049	2004-09-30 07:50:23	2004-09-30 07:50:23	15	4	515	Short Message	Outgoing	0
113412	2004-10-06 03:21:04	2004-10-06 03:20:18	15	510	629	Voice Call	Incoming	46
114703	2004-10-23 01:37:23	2004-10-23 01:37:23	15	511	812	Voice Call	Outgoing	0
112510	2004-09-30 19:57:22	2004-09-30 19:54:00	15	8	518	Voice Call	Outgoing	202
111533	2004-09-12 21:54:02	2004-09-12 21:54:02	15	3	328	Packet Data	Outgoing	0
114780	2004-10-29 04:46:49	2004-10-29 04:45:22	15	510	862	Voice Call	Incoming	87
112490	2004-09-29 16:07:23	2004-09-29 16:06:18	15	510	495	Voice Call	Incoming	65
111544	2004-09-14 03:14:10	2004-09-14 03:14:04	15	510	340	Voice Call	Outgoing	6
112851	2004-09-30 03:28:21	2004-09-30 03:27:53	15	2	508	Voice Call	Outgoing	28
113854	2004-10-13 01:09:06	2004-10-13 01:07:41	15	2	685	Voice Call	Incoming	85
113891	2004-10-15 20:47:53	2004-10-15 20:47:03	15	510	731	Voice Call	Outgoing	50
112911	2004-10-03 00:04:44	2004-10-03 00:04:36	15	510	587	Voice Call	Outgoing	8
114578	2004-10-26 20:32:43	2004-10-26 20:28:18	15	511	844	Voice Call	Outgoing	265
114884	2004-11-07 23:34:32	2004-11-07 23:34:26	15	2	899	Voice Call	Outgoing	6
115168	2004-11-30 01:19:01	2004-11-30 01:18:42	15	15	1129	Voice Call	Outgoing	19
113780	2004-10-14 01:32:06	2004-10-14 01:32:00	15	8	696	Voice Call	Outgoing	6
113424	2004-10-08 18:56:09	2004-10-08 18:56:09	15	511	648	Voice Call	Outgoing	0

Figure 3.5: A Sample View of Reality Mining Dataset

provides rich user behavioral and context data. Seventy-five users are either students or faculty in the MIT Media Laboratory, while the remaining twenty-five are students at the MIT Sloan business school. Of the seventy-five users at the lab, twenty are master students and five are MIT freshman. The information collected includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status (such as charging and idle), which comes primarily from the Context software installed on the Nokia smart phone. The study generated data collected by one hundred human subjects over the course of nine months and represent approximately 500,000 hours of data on users' location, communication and device usage behavior. Upon completion of the study, the dataset is anonymized and made public in MySQL database format to the general academic community.

For our simulation, we extracted ten types of user information from the dataset as our profile signatures:

- Activities per day. This is a log of how many times a user use the phone per day, including utilizing all phone functions from making phone calls to simply checking the time. This information is collected by the phone software that logs the event that the phone exits idle state.

- Intimate users per day. The dataset contains information about how many bluetooth devices are in the intimate range of the current user. We use this number of users as a user behavior signature.

- Tower switch per day. This is the number of how many base station one user will switch between for a single day. This suggests the mobility of a user, frequency of movement and location preferences.

- Calls made per day. How many phone calls a user made for a single day is clearly a signature of that user.

- Voice call percentage. This is a signature of phone usage pattern, suggesting how many percentage of a user's connection usage is used for voice call (other than data packages and text messaging).

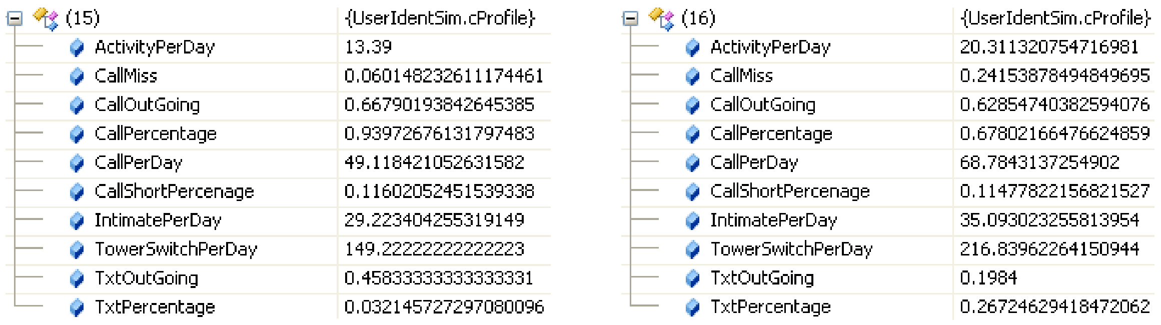
- Outgoing calls percentage. This logs the percentage of phone calls made that are outgoing (other than incoming or missed).

- Missed calls percentage. Similarly this tracks the percentage of incoming phone calls that is missed by the user.

- Short phone calls percentage. This is a signature tracking the percentage of phone calls made that lasts less than three minutes, which suggests the user's social preference on phone calls.

- Text messaging percentage. This is the equivalent of Voice call percentage, suggesting the percentage of phone usage on text messaging.

- Outgoing text messaging percentage. Similar to Outgoing calls percentage, this logs the percentage of text messages that are received other than sent.



(15)	{UserIdentSim.cProfile}
ActivityPerDay	13.39
CallMiss	0.060148232611174461
CallOutGoing	0.66790193842645385
CallPercentage	0.93972676131797483
CallPerDay	49.118421052631582
CallShortPercenage	0.11602052451539338
IntimatePerDay	29.223404255319149
TowerSwitchPerDay	149.22222222222223
TxtOutGoing	0.45833333333333331
TxtPercentage	0.032145727297080096

(16)	{UserIdentSim.cProfile}
ActivityPerDay	20.311320754716981
CallMiss	0.24153878494849695
CallOutGoing	0.62854740382594076
CallPercentage	0.67802166476624859
CallPerDay	68.7843137254902
CallShortPercenage	0.11477822156821527
IntimatePerDay	35.093023255813954
TowerSwitchPerDay	216.83962264150944
TxtOutGoing	0.1984
TxtPercentage	0.26724629418472062

Figure 3.6: A Sample Profile of One User's Behavior

With these user behavior signatures extracted from the dataset, we can build profiles for all the one hundred users. A sample profile of two users is displayed in Figure 3.6.

3.2.2.2 Performance Study

Under the simulation setup, we did a series of performance study to evaluate our user identity recognition model. We use the Matching Score algorithm described in Section 3.2.1.3 to calculate the matching score of one user's behavioral similarity to another. We consider the first few months as the profile training period, during which the profile data of all the ten profile parameters are collected and/or calculated. Starting from the next day after this training period, we randomly pick a user and generate her identity signatures of the same ten profile parameters. Then her signature is compared to the other one hundred user's profiles collected in the last few months: The one with lowest matching score is the match found by our algorithm. Now we will reveal the true identity of the randomly picked user and see if the matching is correct. This "pick and match" process is then repeated and we collect the success rate of our algorithm in terms of correctly identifying the true identity of the user. Sometimes for the specified time period, there can be no data of the specified user. This is normal because the user may fail to turn on the phone for some reason. In such cases the study for this user is terminated and it will not affect the final result.

The simulation result is displayed in Figure 3.7. As we can see from the Figure, our user recognition model achieves a successful recognition rate of over 80% when enough data

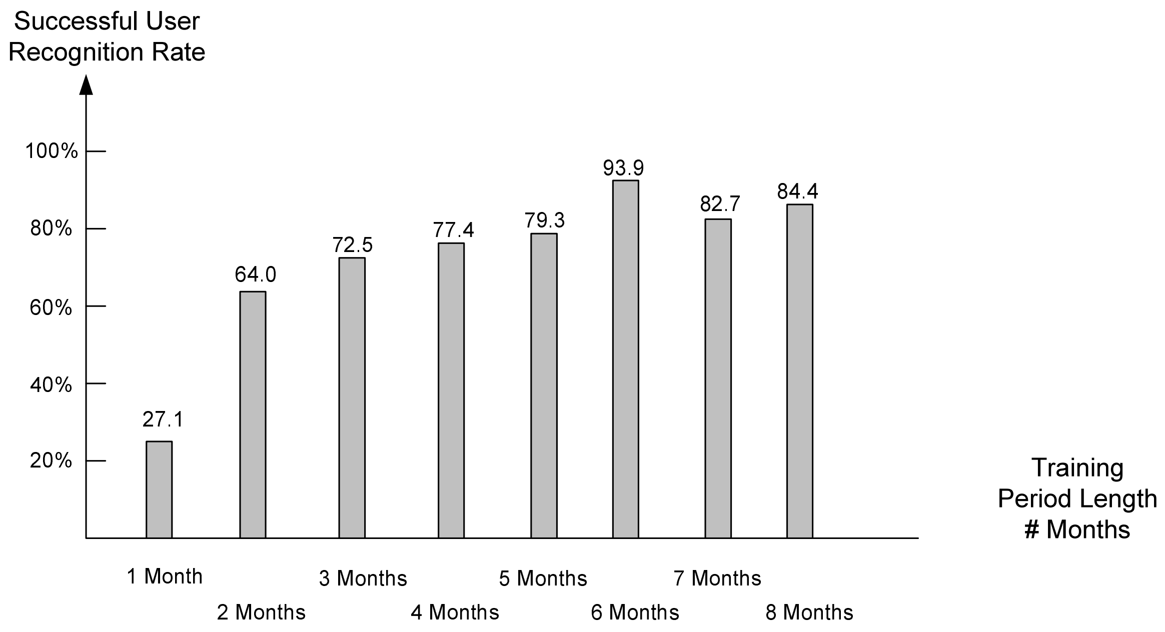


Figure 3.7: Simulation Result based on Training Period Length

is collected to build up profiles. The figure also shows that while the profile is gradually established, the recognition success rate is increasing accordingly. We noticed that the result of 6-months profile data is higher than the others, even better than the 7-months and 8-months' performance. We further investigated the problem and find out that this is because of the lack of data profiles for the 6-month simulation: a lot of users do not have any record during the time period we are investigating. Due to this lack of user data, the total number of simulations during this time period is dramatically decreased, which could have caused the result to become more random than the others.

3.2.3 Dynamic Reactions to Suspicious User Behaviors

The direct influence of the user behavior identification process is to trigger a reevaluation of the suspicious user's trust level. The trust management system [49] will take the result from the user behavior identification system and broadcast the result to the community and advise other nodes to downgrade the trust level of the suspicious user. While it is the other nodes' own decisions of whether to trust this early warning or not, this will have negative effects of the trust evaluation of the suspicious user as a whole.

Besides the trust management, the system also needs to take some countermeasures to address the potential threat. The first reaction of the system is to keep close watch on the suspicious users. For legal issues, it is not legitimate to directly stop the suspicious user from further services before they start to put the attack into practice. For example, in our scenario, the potential bank robbers should not be rejected from the pervasive computing services from the bank lobby space. But the system can start monitoring the suspicious users right after their behaviors matching the malicious user signatures. With the powerful security system resources dedicated to monitor a few suspicious nodes, it is much easier to take a timely first reaction once the suspicious user starts to perform illegal actions.

In addition to the monitoring, the reaction module should also collect possible crime evidence generated by the suspicious users, such as the actions they take, the resources they access, the data they send and receive, etc. These forensics logs can record very detailed information about each suspicious user without taking too much system resources.

Finally the reaction system can also feed the suspicious user with some poison information that is harmless to normal users. For example, in our scenario, the bank system can generate fake information about the bank security setup to mislead the potential robbers in order to protect its property.

To conclude this section, we introduced a user behavior identification system to find out suspicious users by logging malicious user behavior into a profile database and comparing user behavior signatures with these profiles. Our simulation shows that our user recognition model can help identify a user from its behavioral profile. Our reaction module will take actions to prevent further possible malicious actions by those suspicious users. The user behavior identification system, together with the authentication technologies, can be very helpful for pervasive computing system to recognize a user's true identity in order to help we manage the trust relations.

CHAPTER 4: TRUST MANAGEMENT

In the previous chapter, we discussed about our solution towards user recognition, which paves the way for a successful trust management. In this chapter, we will focus on the definition of trust and propose a trust model for pervasive computing systems. Then we will discuss the trust relation propagation and present our algorithm. Finally we will discuss attack schemes against the trust model and show our simulation result of our trust model's robustness against such attacks.

4.1 Trust Model

In this section, we present the details of our trust model. As we discussed previously, various trust models have been developed from different application perspectives in current research literature. A major challenge for a trust model is that trust is application-dependent and hence we need trust models that provide customized trust information to applications. The following desired properties need to be considered in a trust model:

- Trust is specific to the service being provided by the trustee entities. For example, entity A may trust entity B on providing a reliable web service, but not a file storage service. So trust depends on the tasks that each party is expected to perform in the context of a particular application.

- Trust is a measurable belief. There are different degrees of trust. For example, party A may trust party B more than A trusts party C for the same service. This property emphasizes the need for a suitable metric to evaluate trust. Such metric is based on evidence, experience and perception. The measurement can be quantitative (e.g., a probability assignment) or relative (e.g., by means of a partial order).

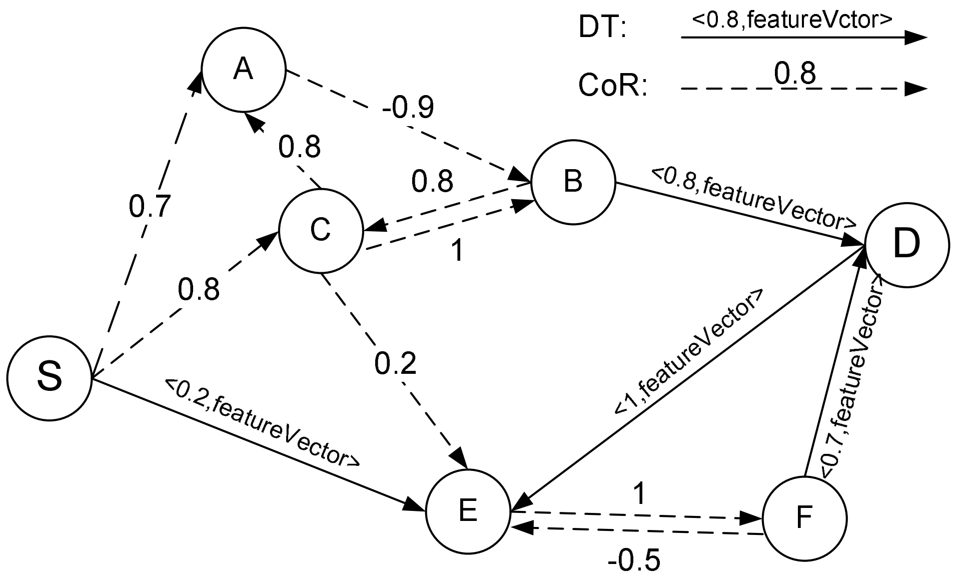


Figure 4.1: A Global Trust Graph

- Trust has limited time duration and evolves over time. The fact that party A trusted party B in the past does not guarantee that A will trust B in the future. B's performance and other relevant information may lead A to re-evaluate the trust on B. During a service interaction, the more A realizes that A can depend on B for service X, the more A trusts B. On the other hand, A's trust in B may decrease if B proves to be less dependable than anticipated by A.

A good trust model needs to be designed in such a way to fulfill all the properties above.

4.1.1 Trust Model Overview

The core of our proposed trust model is based on the notion of trust relationships among collaborative entities. Entities in our model can be of various types and be specified at different levels of granularity. For example, an entity can be an organization, a department within an organization, or a single individual. We model the trust relationship as a directed trust graph as shown in Figure 4.1. In the model, there is an edge from entity A, referred to as the trustor, to entity B, referred to as the trustee, if and only if A has a trust relation with B. A trust graph is not necessarily same as the connection graph of a network, such as the routing graph of an ad hoc network or connectivity graph of a P2P network. Though being neighbors on the connection graph may help two nodes establish trust relations easily, the trust graph can be completely distinctive from the connection graph.

In most application scenarios there is a concept of a global trust graph. The edges and vertices of the global trust graph is real-time changing. Most of time, it is neither feasible nor necessary for a single entity to maintain the global trust graph unless there is a central entity. Instead, each entity maintains a local trust graph. The local trust graph is a subgraph of the global graph at some point of time, but only contains trust relations that an entity is aware of. There are two types of edges in a trust graph representing two basic types of trust, Direct Trust (DT) relationship (in solid line on Figure 4.1) and Confidence of Recommendation (CoR) relationship (in dashed line).

CoR represents the degree of confidence on a trustee to provide accurate recommendation. CoR is labeled with a trust value. A trust value is a measurable degree of trust. It is a real number within [-1, 1], where 1 means the most trustable and -1 means the most distrusted. The trust value is the measurement of an entity's "honesty" degree on recommendation. There are other approaches that use a real number from 0 to 1 as a trust value. Mathematically speaking, these two representation are equivalent through a one-to-one mapping between them.

DT represents a direct trust relationship between a trustor and a trustee for various services provided by the trustee. DT is constrained by some conditions and service aspects, such as the trustee granularity, preconditions, and events that can change the trust relation. We propose to use feature vector to specify and describe such constrain information in a DT. In our trust graph, DT is labeled with <trustValue, featureVector>. Similar to CoR, the trust value is ranged between -1 and 1. The feature vector records all relevant aspects affecting the trust relationship represented by the edge. In particular, a feature vector is an abstract description of various aspects of the trust relationship, including the following items:

- Trustee Specification. This item specifies the granularity of the trustee. The gran-

```
<!ELEMENT FeatureVector (TrusteeSpecification, Precondition,
TemporalConstrains)>

<!ELEMENT TrusteeSpecification (GranularityLevel, Trustee)>
<!ELEMENT Precondition (ConditionDescription*)>
<!ELEMENT TemporalConstrains (ExpirationTime, BlackoutTime)>
<!ELEMENT GranularityLevel (#PCDATA)>
<!ELEMENT Trustee (#PCDATA)>
<!ELEMENT ConditionDescription (#PCDATA)>
<!ELEMENT ExpirationTime (#PCDATA)>
<!ELEMENT BlackoutTime (#PCDATA)>
```

Figure 4.2: DTD for Trust Feature Vector

```
<!ELEMENT TrustEvaluationFunction(Function, TrustHistory)>

<!ELEMENT Function (#PCDATA)>
<!ELEMENT TrustHistory (HistoryDataItem)>
<!ELEMENT HistoryDataItem (Record, RecordTime, TrustValue)>
<!ELEMENT Record (GoodTransaction, TotalTransaction)>
<!ELEMENT RecordTime (#PCDATA)>
<!ELEMENT TrustValue (#PCDATA)>
<!ELEMENT GoodTransaction(#PCDATA)>
<!ELEMENT TotalTransaction(#PCDATA)>
```

Figure 4.3: DTD for Trust Evaluation Function

ularity level could be a computing node, one of node's services, or even one trust requirement of a service.

- Preconditions. When a trust evaluation process is initiated, the preconditions are evaluated first. Those conditions are the prerequisite that must hold for the trust relation. A parser can be used to help the application understand these conditions and evaluate them accordingly.

- Temporal Constrains. The temporal constrains specifies the valid lifetime of the trust relation. Specifically, it describes the expiration time and the blackout time of the trust relation.

The challenge is to define feature vectors for trust relationships within a distributed service infrastructure. We introduce an XML based language to describe feature vectors. We use Document Type Definition (DTD) schema language [13] to define the feature vector format (Figure 4.2). According to the DTD template, we use XML based languages [14] to specify the constrains in a feature vector in a uniform and standard way so that they can be transmitted and understood by system entities. This feature enables trust propagation for the case when no prior trust between trustor and the trustee is available.

In addition to these constrains, the trustor also specifies its trust evaluation function on exactly how the trust value of a DT and a CoR is calculated. The trust evaluation functions are specified in a separate XML document. This separation facilitates trust propagation. While the trust value and its constrains need to be distributed to other nodes to build up their local trust graphs, the evaluation algorithm of how trust is calculated is usually kept secretly. The Document Type Definition template for trust evaluation function is shown in Figure 4.3.

4.1.2 Trust Value Calculation

As we discussed, trust is a measurable belief. There are different degrees of trust. The measurement can be based on evidence, experience and perception. The measurement results can be quantitative (e.g., a probability assignment) or relative (e.g., by means of a partial order). In our model, we use trust value to represent the degree of trust. Most current research on trust value evaluation are from some mathematical models, such as probability-based models and information theory models. The trust values from such models cannot always satisfy all desired properties. In order to address this problem, extra factors and parameters outside the models need to be introduced. Such an introduction somehow weakens the purpose and validity of the mathematical models.

Unlike most current research, our trust value formulae start from the basic properties, and consider both the objective and subjective parts. An example of objective parts is the probability of good actions, similar to the current probability-based trust models. An example of the subjective part can be the amount of reward (penalty) of a good (bad) action. We incorporate the following desired properties into our trust value evaluation:

- Trust value evolves over time and is time sensitive. More recent actions should have

$$\text{CoR} = \begin{cases} \text{Min}(\text{CoR}_0 * \beta^{\Delta t} + \text{RWD}*(1+\text{RAT}),\ \text{CoR}_0+(1-\text{CoR}_0)*\text{CVG}) \\ \textit{if recommendation==good} \\ \\ \text{Max}(\text{CoR}_0 * \beta^{\Delta t} - \text{PNT}*(2-\text{RAT}),\ -1) \\ \textit{if recommendation==bad} \end{cases}$$

$$\text{RAT} = \frac{\text{NumOf(GoodRecommendations)}}{\text{NumOf(TotalRecommendations)}}$$

Figure 4.4: Calculation of CoR

more impact on the trust value.

- Trust value should increase with good actions and decrease with bad actions. Furthermore, in most application scenarios, good actions should increase the value slower and bad actions should decrease the value faster.

- Trust value should encourage a good trust history. An entity, with a better trust history than another entity, will have more increase on trust value with a good action, and less decrease on trust value with a bad action.

- Trust value evaluation can be customized by applications. For example, the penalty of a bad action in a security-sensitive service can be more severe than in ad hoc routing service.

We present a trust calculation method in compliance to these properties. We first discuss the trust value formula for the CoR. There are several inputs for the CoR evaluation: the previous CoR value, the quality of this recommendation (good or bad), the overall good recommendation rate (good recommendations over all recommendations), the time interval between the current evaluation and previous evaluation, and the trust value from the recommender to the recommendee. Our proposed CoR formula is shown in Figure 4.4.

In the formula, CoR_0 stands for the cached pervious CoR value. β is the forgetting factor for time in the range of (0, 1], which is used to help entities “forget" out-of-date trust events and put more weights to recent actions on trust evaluation. Δt is the time interval between current time and the time when CoR_0 is cached. RWD is the reward factor for a good recommendation and PNT is the penalty factor for a bad one. A typical value for RWD is 0.1 and 0.2 for PNT. RAT is the ratio of historical good recommendation over all recommendations, and CVG is a parameter to control the convergence speed of CoR.

When the trustee of one CoR made a good recommendation (by either giving positive recommendation to good nodes or negative recommendation to bad nodes), the trustor will try to reward the trustee. The first part of the formula is the multiplication of its cached previous CoR value by the amount of $\beta^{\Delta t}$, which helps “forget" the previous value a little bit. The second part is $RWD * (1 + RAT)$, the reward of this good recommendation. Since RAT is the good over all recommendation rate, ranging from 0 to 1, a good node with $RAT = 1$ will have two times more reward in trust value than a bad node with $RAT = 0$. This helps make our calculation scheme satisfy the third property. Finally since the rewarding of a trust value may cause the result be out of the valid range of CoR ([-1, 1]), we apply a cap $CoR_0 + (1 - CoR_0) * CVG$ on this calculated result. CVG is a tunable parameter within (0,1], which controls the convergence speed of CoR when it is growing. A typical value for CVG is 0.1, which means that whenever the CoR grows, it can only increase by 10% of the difference between the current CoR and 1. If CVG is set to 1, the cap of CoR becomes 1, thus the CoR value can grow to 1 within limited rewards. The calculation scheme for penalizing a bad recommendation is similar to rewarding. The only difference is that we use -1 as the lower cap for the calculation result.

The parameters β, RWD, PNT and CVG can be adjusted by the application. For example, when a good node may change its performance from time to time due to environmental factors, we may consider adjusting the forgetting factor so that recent actions take much more weight. On the other hand, if computing nodes are constantly moving in and out of the system, we need to increase the base reward and penalty amount, so that a newcomer can quickly establish enough trust. Applications can customize the values of these parameters based on their needs or even adjust them dynamically during run time based on the trust system performance. For example, if an entity senses that there is an increase of malicious users, it can increase the penalty of a bad action, making malicious users being detected more quickly.

The trust value calculation of DT will be similar to that of CoR. The additional consideration is service-related. A services can introduce an application-specific factor for the reward and penalty.

4.2 Trust Relation Establishment

The previous section defined trust relationship together with a trust model. In this section, we will discuss the trust relationship establishment in a general context: we will investigate how the system determines the trustworthiness of a client after collecting and evaluating the trust evidence about the client. Trust establishment between two entities is grounded in one's evaluation on another's ability, benevolence and integrity, and the trust evaluation is based on trust evidences that describe the client's intention or behavior pattern. Three types of client trust evidences are identified to be used for trust establishment: credentials, environment context, and behavior records. We will analyze these types of trust evidences in the following sections by presenting our automated trust negotiation strategy.

4.2.1 Active Automated Trust Negotiation

The main challenges for automated trust negotiation are how to define, maintain, and acquire the interoperable trust negotiation strategies and protocols. In this section, we first look at the general considerations for defining , maintaining, and acquiring the trust negotiation strategies, then presents an effective approach, active trust negotiation scheme, for automated trust negotiation.

As we mentioned, the trust negotiation process requires both the client and the system must have the interoperable negotiation strategies and protocols in place before the negotiation starts. Due to the computing space or domain limitations, the client usually does not have all the trust negotiation strategies when the client intends to negotiate for accessing the resources in the system. In this case, the client needs to acquire them from the system or other resources when trust negotiation is on demand. Otherwise, the trust negotiation cannot proceed.

The interoperable trust negotiation strategies are in a family, and any two of them can be used by the client and the system to ensure the trust negotiation process to proceed. One special case is that the client and system are using one same trust negotiation strategy so that the interoperability is definitely ensured. This property is what we expect and provides an option for use to choose trust negotiation strategies.

The trust negotiation strategy and protocol for a resource in the system is mainly determined by the trust control polices of the resource. In other words, each trust policy in the system with negotiation ability should specifies a trust negotiation strategy to be used when trust negotiation is on demand. If the client intends to negotiate to access a resource in a system, both the client and system can use the same trust negotiation strategy specified by the trust control policy for the resource in the system.

For the performance and security reason, the negotiation strategies and protocols for the resources are usually defined and maintained in the same system as the policies reside. Because trust negotiation strategies are used locally to help maintain and control the release sequence of local credentials, they must be under the control of local environments for efficiency and security. One effective and dynamic way for the client to get the trust negotiation strategy from the system is to download the trust negotiation strategy on-line when the client cannot find an interoperable trust negotiation strategy and protocol locally. This leads to our active automated trust negotiation scheme, which operates in two stages: setup stage - trust negotiation strategy download phase, and execution stage - trust negotiation execution phase.

The active trust negotiation scheme includes two stages: trust negotiation strategy download and trust negotiation execution. The process in each stage poses challenges in the implementation. In the following subsections, we will discuss the general procedures in each stage and the main issues concerned.

In this stage, the client and system will go through the following procedures in sequence:

1. The client requests to access a resource in the system, and both the client and system realizes that trust negotiation is needed.

2. The system notifies the client what trust strategy and protocol is used for the trust negotiation, and the client is searching for an interoperable trust strategy and protocol locally.

3. If the client does not find an interoperable trust negotiation strategy and protocol, it requests to download the trust strategy from the system so that the client and system share the same trust negotiation strategy and protocol. Else, the client will use its local interoperable trust negotiation strategy.

4. Both the client and system notify each other after loading the trust negotiation strategies, respectively.

During this setup stage, the following issues should be put into consideration for security and compatibility:

- Download process should be secure enough so that it ensures that the downloaded codes from a system do not include the malicious codes. This needs a secure and reliable download protocol.

- The trust strategy and protocol being downloaded should be applicable and compatible with the client system, i.e. the client system can understand what the strategy and protocol denote and use them to execute the trust negotiation.

After the client has found an local interoperable trust negotiation strategy or downloaded the trust negotiation strategy from the system, both the client and system will load the trust negotiation strategy and protocol and start the trust negotiation. The client and the system will go through the following procedures.

1. Both the client and the system exchange credentials as requested by the trust control polices, which is guided by the trust negotiation strategy and is using the trust negotiation protocol for messages transfer.

2. If the trust negotiation code executer (normally the client) is not confident of the code security, it may use sandbox technologies to help protect itself from malicious code.

3. If the client's credentials meet the requirement of the trust polices for accessing the requested resource, the system grants the access rights for the client and the client will perform the allowable actions on the resource. Else, no trust can be established.

Trust negotiation strategies and protocols are related to the trust control policies in the system. In our case, trust negotiation strategies will be downloaded to the clients, so one important thing to be considered is to make the trust negotiation strategies compatible with the local trust control policies in the client.

Although there are still more research work to be done in providing secure and efficient trust negotiation strategies, it is not our focus of our research. We assume that there are valid trust negotiation strategies being stored in the system and specified by trust control policies.

4.2.2 Agent-based Infrastructure for Trust Negotiation

The concept for the active automated trust negotiation provides a way for trust negotiation for the mobile entities in pervasive computing environments theoretically. As we mentioned, there are several issues concerned such as download protocols, trust policy language and trust negotiation strategy encoding. In this section, an agent-based infrastructure

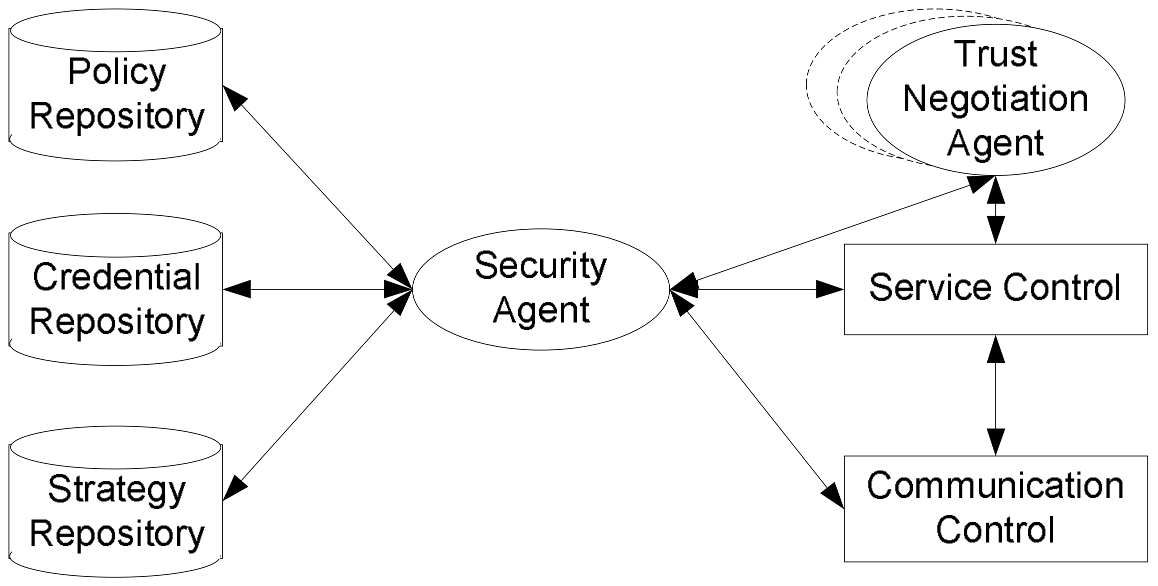


Figure 4.5: Overview of Agent-based Infrastructure for Trust Management

for trust management is proposed to implement the active trust negotiation scheme and provide the corresponding solutions to the related issues.

We first present the overview of the framework and the main components in this framework, then concentrate on the trust negotiation agent design and the detailed mechanisms to resolve the implementation issues for the active trust negotiation scheme in the infrastructure.

Figure 4.5 illustrates our agent-based infrastructure for trust management in pervasive computing environments. As described below, this infrastructure is aimed to realize trust management with different trust management strategies for different application scenarios such as trust negotiation, trust recommendation and etc. In this section, we will focus on trust negotiation and discuss the design of trust negotiation agent and the implementation of the active trust negotiation scheme.

The main components in this trust management infrastructure include service control, communication control, security agent, various repositories, and different trust agents such as trust negotiation agent, trust recommendation agents and so on.

Service control works as a general controller and is the main interface to communicate with the external entities. All the service requests from clients should go through the service control. The service control will consult with the security agent for the security enforcement. It is service control that initiates the corresponding trust management processes based on the consulting result from the security agent. The service control requests the communication control to allocate the communication protocols and channels for specific tasks or specific components in the systems (e.g. trust negotiation is done through the negotiation communication channels). In this way, some components (e.g. trust negotiation agents) will communicate with the external entities through the specified channels. The service control also keeps all the current clients and manages communication session information.

Security agent enforces the security of the local system. It maintains the policy repository, the credential repository and the strategy repository and uses the trust control polices and digital signatures of credentials to perform trust management. All the regular authentication and authorization are supported by the security agent. The security agent also delegates the trust evaluation tasks to the specific agents based on different trust evidences types and application scenarios. By synthesizing the evaluation results from all the available agents, it concludes the trustworthiness of the clients and makes the decision for granting the access rights for the client so that the client can perform on the resources.

Communication control is a component for allocating the communication channels and protocols. It usually receives the requests from the service control or the security agent for allocating the communication channels and protocols and monitoring the data traffics. The commonly used channels and protocols are specified by the service control when the system starts. Specific agents such as trust negotiation agents are allocated specific channels respectively.

The policy repository stores the rules and polices which are used by the security agent to enforce the system security. The credential repository keeps the credential information and the account information for the system itself and clients. The negotiation strategy repository keeps all the trust negotiation strategies and protocols, which are specified by the trust control policies and used for trust negotiation. All these repositories and the information stored in them are controlled by the security agents and can be queried by the other agents.

These agents are the specific agents for handling the trust evaluation tasks for different application scenarios and trust evidence types. Since different application scenarios need different methods for collecting trust evidences and use different criteria for evaluating trust evidences, different agents are used to implement these different trust establishment mechanisms respectively. Since our focus is on trust negotiation process, we will describe its detailed implementation and how it realizes our two-stage active trust negotiation strategies.

Trust negotiation agent in the trust management infrastructure is responsible for trust negotiation. It receives the delegated task from Security Agent and Service Control for negotiating to establish trust relationship with the other party. Following our active trust negotiation scheme, trust negotiation agent will do the following main things:

- Downloading the trust negotiation strategies
- Loading the trust strategies locally
- Executing the trust negotiation based on the trust strategies.

The detail implementation are related to trust control policies and languages, trust negotiation protocols, trust negotiation strategies, secure download protocol, and negotiation agent design. In the following subsections, detailed implementation considerations are presented for each component and protocols.

Trust negotiation process is to exchange the credentials (i.e. trust evidences) for trust establishment. It gets involved in the request for credentials and the release of credentials, which in turn needs the trust policies to specify what the requested credentials should be, and relies on the trust negotiation strategies and protocols to determine how to guide and control the request and release of the credentials.

Trust negotiation must start with some common non-critical credential exchanges, then proceed on critical credential exchanges guided by trust control policies and trust negotiation strategies. In other words, trust negotiation is not a zero knowledge based process. Trust control policies, trust negotiation strategies and protocols are related to how to describe the credentials or handle the credentials, so compatible languages are needed for semantic consistency when describing the credentials and action types. Therefore, the following assumptions are made for simplicity.

- The trust negotiation discussed is not a zero knowledge based process. The client and the system must have some common base or non-critical credentials (e.g. known identifications) to start with.
- The same trust policy language is used for defining trust or access control policies so that the concept or vocabulary in the trust control policies can be understood by the trust negotiation strategies.
- The same protocol language is used for defining the trust negotiation protocol and encoding the messages, whose concept and vocabulary are the same as (or compatible with) the ones in trust control policies.
- These trust negotiation strategies and protocols are defined and stored in the systems and downloaded to the client.

Trust control policy of a resource in the system specifies the context and the required credentials for the client to present for trust evaluation. The system can grant the client to access the resource if the client's credentials meet the requirements specified by the trust control policy. In trust negotiation process, in order not to release the content of trust control policies of the system totally, the trust control policies should be designed to request the client's credentials gradually from the non-critical ones, less critical ones to the more critical ones. Meanwhile, the trust policies in the client may also specify the gradual request sequence for some system credentials before the client's credentials are released to the system. The trust control policies are also called *disclosure polices (or negotiation policies)*, which define the gradual disclosure sequences of the contents of policies and credentials for trust negotiation. We still call it *trust control policies*.

Trust control polices are defined by policy languages. For simplicity, we assume that trust policies are defined using one same policy language in our implementation. An XML-based trust policy language similar to the trust Language TRN-X defined in [25] or TPL in [16] can be an appropriate option due to its applicability and flexibility.

Trust negotiation protocol defines the format for trust negotiation messages, which includes the requests for credentials, the replies for the requests, and the command to start or halt the trust negotiations as well as the contents of messages such as credentials and policies. We assume that protocols are written in XML to encode all the commands, credentials and polices in the messages, which match the trust policy language in both vocabulary and semantics. The negotiation protocols are stored in the system together with the trust negotiation strategy and will be downloaded to the clients.

Trust negotiation strategies are the algorithms for controlling the credential disclosure based on the trust control policies. So this requires the trust negotiation strategies to interpret the trust control polices, maintain the acquired credentials and arrange the credentials disclosure sequence during trust negotiation. Since a trust negotiation strategy is an algorithm, it should be implemented as an integral program that can be executed by the calling environment. A client will download the program, load it into the local running environment and execute the program. Also, the trust negotiation strategies should be under the control of trust negotiation agents and security agent.

For this purpose, we extend the concept of Active Capability given in [46, 47] and define Security Capsule to encode the trust negotiation strategies. A Security Capsule is an executable code that encodes the negotiation strategy and protocol. It includes the data types and algorithm of the trust negotiation strategy and can be loaded and executed by

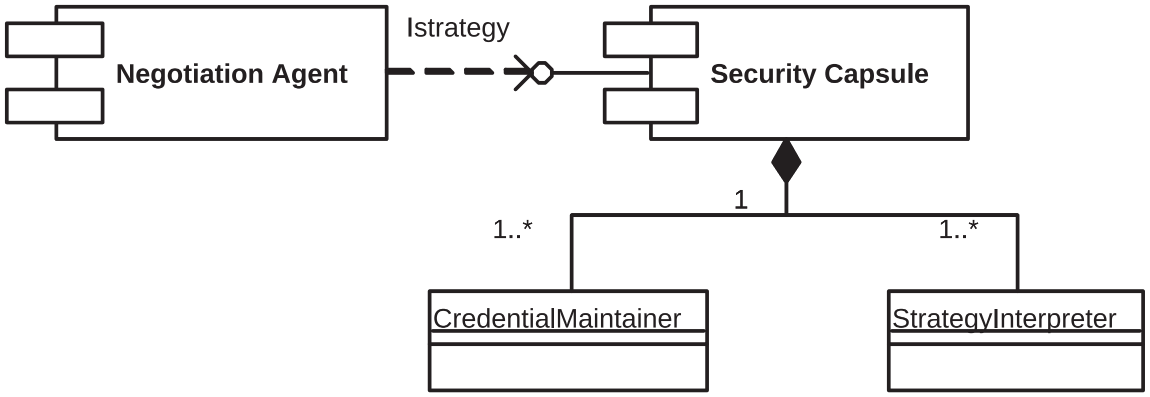


Figure 4.6: The Relationship between Trust Negotiation Agent and Security Capsule

the negotiation agent. For the compatibility and integrity of the trust negotiation strategy, Security Capsules implement a common interface with the negotiation agent. The negotiation agent queries or receives the information about the credential request or disclosure sequence to form the negotiation messages according to the negotiation protocols. In this way, the Security Capsule is transparent to negotiation agents and can be loaded to any trust negotiation system if the negotiation agents import the common interface.

The relationship between the negotiation agent and Security Capsule is illustrated using UML component diagram in Figure 4.6, in which *Istrategy* represents the common interface that is implemented by all the Security Capsule components, *StrategyMaintainer* object is responsible for interpreting the trust control policies and controlling the credential release based on the control polices, and *CredentialMaintainer* is responsible to maintain the external credentials provided by the counter party and the local credentials that have been released to the counter party.

In general, the Security Capsules have the following advantages by extending the concept of Active Capability:

- Mobility - they can be downloaded from the system on-line dynamically.
- Ready for use - they are programs that can be executed when being loaded into the run-time and can be reused by moving to any systems or clients.

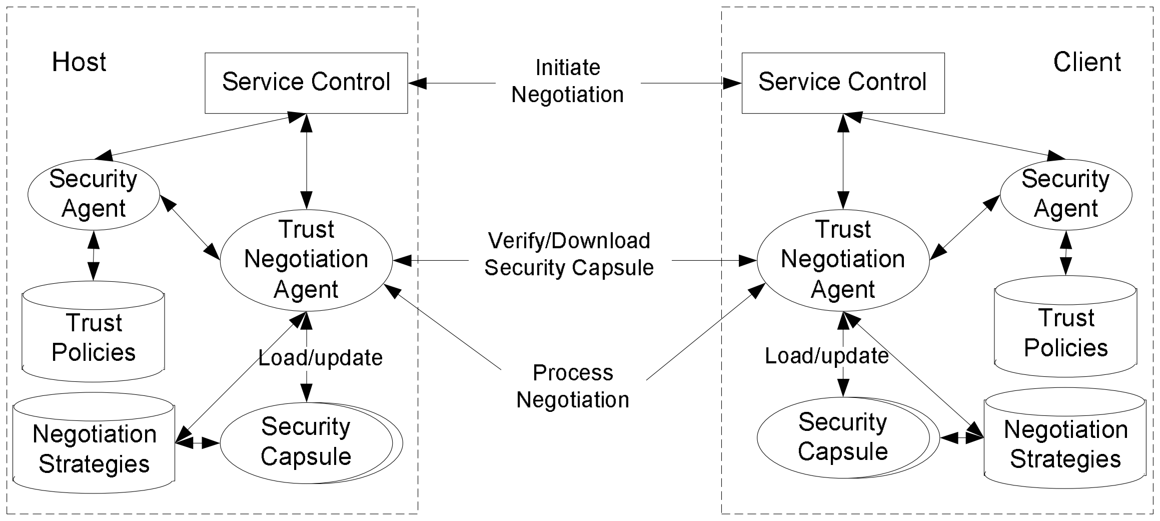


Figure 4.7: Negotiation Process

- Configurability - a Security capsule interacts with Negotiation Agent through the common interface. Any updating in the internal objects is transparent to the Negotiation Agent. So a Security Capsule can be used for different applications by reconfiguring it if it is designed so.

When the client downloads the security capsules, it should ensure that security capsules do not contain any malicious codes. This involves the following two step operations.

- The download protocol should ensure the integrity of the downloaded security capsule, which requires the protocol to have a checking mechanisms for verification. The security hash method of MD5 can provide this functionality.

- The security capsule can only communicate with the negotiation agent locally through the common interface and it cannot leak any information and compromise the local environments. This can be realized by the local security policies and security domain such as the sandboxes.

- The security capsule will be verified using some bench trust control policies for its validity.

Trust negotiation agent is responsible for trust negotiation task in our agent-based trust management infrastructure. During the two-stage trust negotiation process, when the Service Control receives a request for access to a resources, it consults with the security agent and concludes that trust negotiation is needed, then delegate the task to the trust negotiation agent. Then the trust negotiation agents from both sides will check whether the trust negotiation strategies required by the trust policy can be found locally. If the client cannot find a compatible one locally, its trust negotiation agent will request download the security capsule for the trust strategies from the host. When the trust strategies are present with the security capsules and ready for use, the trust negotiation agents will start to execute the negotiation process and begin exchanging credentials, which is guided by the trust negotiation strategies. The general trust negotiation process is illustrated in Figure 4.7.

As illustrated in Figure 4.7, the client most times needs to download the trust negotiation strategies. Therefore, the trust negotiation agent needs to maintains a reliable download protocol in addition to the trust negotiation protocols. All these protocols should be compatible with the vocabulary and semantics of the trust control policies and trust negotiation strategies, which are usually defined using the policy languages.

4.2.3 Trust Maintenance

Besides an effective trust negotiation scheme, the next important research is to ensure trust maintenance after trust relations are established. In pervasive computing systems, because the nodes are moving in and out the computing space constantly, this dynamic characteristic requires mighty trust maintenance schemes to guarantee that the established trust is not abused.

The proposed trust maintenance and management module is composed of several collaborative components (Figure 4.8), which resides on different computing devices and works in a on-demand-assembling fashion to fulfill the security demand of certain services. Some of the components identifies a user, monitors its behavior to detect potential threats and intervenes the service execution; while others manage the established trust in a

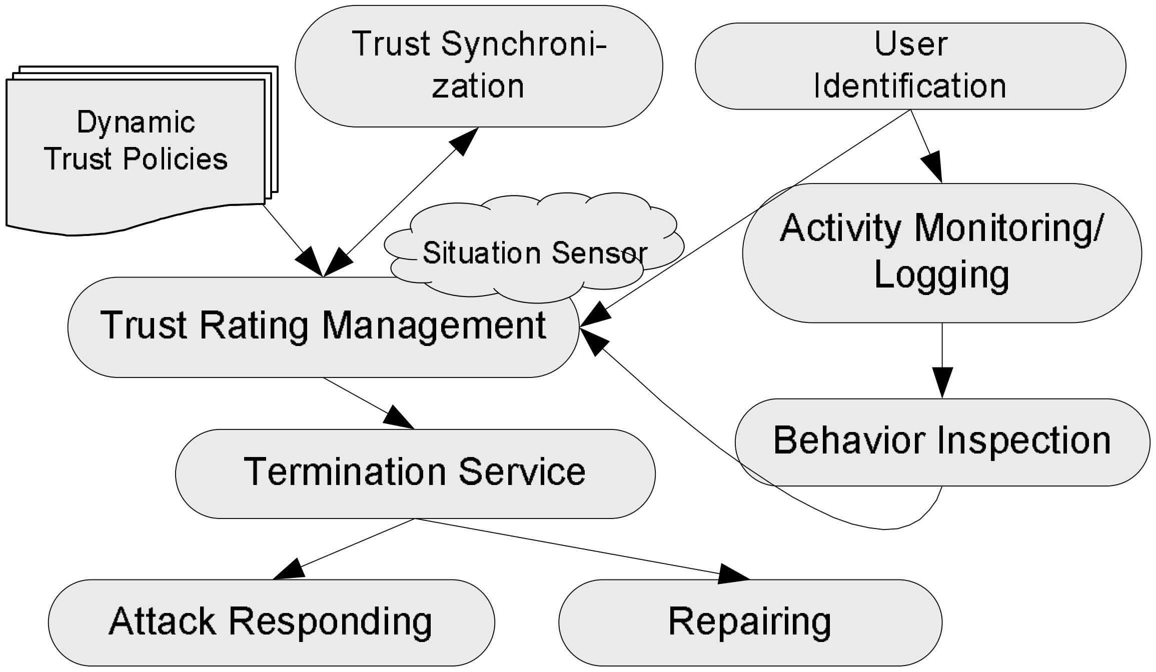


Figure 4.8: Trust Maintenance and Management

distributed manner.

Trust synchronization component keeps the trust repositories on different virtual servers up-to-date. To ensure the system stableness and reliableness, one pervasive computing space may adopt multiple virtual servers running at the same time, each virtually providing all available services. These proxies may host trust establishment with several clients at the same time. Since there is high probability that a newly enrolled client may immediately access other services residing in the same space, the client profile on each virtual server needs to be synchronized with each other to avoid the client repeatedly negotiating trust with these services.

Trust rating component uses incentives that reward good behaviors while punishing bad ones. Trust rating system provides a feedback interface for each service to evaluate the client's behavior in the transaction when the service session is completed. Besides the service's evaluation, a client's rating score is also adjusted by the virtual server's monitoring component. The rating score is distributed in the system repositories. The client may also request a copy of its rating certificate signed by the pervasive computing space and keep it as a future credential for other services. Correspondingly, a low trust rating of a client can have negative effects on its future resource accessing or even result in the shut done of the ongoing service.

To cut the realtime-detected malicious users from further endangering the system, *termination service* is designed to shut down an ongoing service session, under the circumstance that one's trustworthiness remarkably declines to an unacceptable level during the service session. The trust level drop may result from an abnormal behavior, detected by the behavior inspection component, or from user identification component suggesting that the client tries to hide its dishonest history. Because the termination service shuts down ongoing services, which can seriously affects service quality if the user is innocent, it should only be called with high enough confidence.

Complete prevention of bad behavior is be infeasible. When the system is compromised or damaged, the *attack responding and repairing* components are activated to minimize the loss. When the attack comes from a client, the responding and repairing components immediately shut down its service and build up a special profile for the attacker, including its conventional behavior and preferences, for future identifying the attacker again and even further lawsuit purposes. If the attack is from an ill-behaved service, the responding and repairing components notify all distributed virtual servers to block it from clients and perform further inspections.

Because pervasive computing environments involve very dynamic population, it is difficult to identify a user or her devices. Attackers can change its identity after its reputation (in terms of trust rating scores) has fallen lower than initial value [24]. We propose a *user identification* component to address this issue to recognizing a user. The user identification component is designed based on the assumption that every user behaves uniquely during a connection session. The hours of day one usually comes, the displaying devices one prefers, the networking traffic compositions, the preferred access point (may suggest preferred position/location), the service set one uses, the temperature one prefers and so on, all the information together can uniquely identity a user. Our user identification component creates profiles for each detected ill-behaved device and its user, with a matrix of the user preferences. When a self-claimed newcomer is trying to enter the pervasive computing space, a signature based inspection will be applied to the matrix of every attacker profiles. A dishonest veteran that claims itself as a newcomer will be denied for further services.

To recognize dishonest behavior that tries to harm the system and to support trust rating component's evaluation, the *activity monitoring and logging* component is dedicated to track user behavior and log this information for further inspection and evaluation. The designing principle of activity monitoring component is to make the logging information thin and efficient. To keep the logging database thin, only critical activities such as login failures should be stored for a reasonably long time (e.g. one year). Other logging information will only stay in the repository for a short time and then be deleted.

Based on the logging information collected by activity monitoring component, *behavior inspection* applies both abnormality based and signature based inspection methods to recognize security breaches. For example, if a supermarket customer is exploring the RFID product querying system by scanning hundreds of items per minute, the abnormality analysis engine on activity monitoring component will generate an alarm, indicating the customer is behaving abnormally. This alarm may be applied to reevaluate client trustworthiness as a feedback to the trust rating management component.

Trust maintenance and management components may be physically residing on distributed computing devices, collaborating to provide trust support. Though the trust maintenance as a whole is a on-demand-assembled component in the architecture, its subcomponents are themselves optionally re-combinable. According to the requirements of a service session, these subcomponents will aggregate to form a trust maintenance component to fulfill the security demand.

4.3 Trust Propagation

The previous section discussed the establishment of a trust relation. In this section, we will focus on how an existing trust relation can propagate on the trust graph and be utilized by third parties other than the trustor and trustee. In distributed service infrastructures, a single node's observation on one specific trustee is not sufficient to make a decision. Even sometimes there are no existing direct trust relation between the trustor and trustee before they start a transaction. Therefore trust propagation is needed to calculate the trustee's trust value based on the trustor's local trust graph.

Trust propagation includes two aspects: concatenation and aggregation. Concatenation is the process to generate a trust value along one single trust path, starting from the trustor all the way to the trustee. Aggregation, on the other hand, combines the evaluation results of different trust paths and derives the final results of trust evaluation.

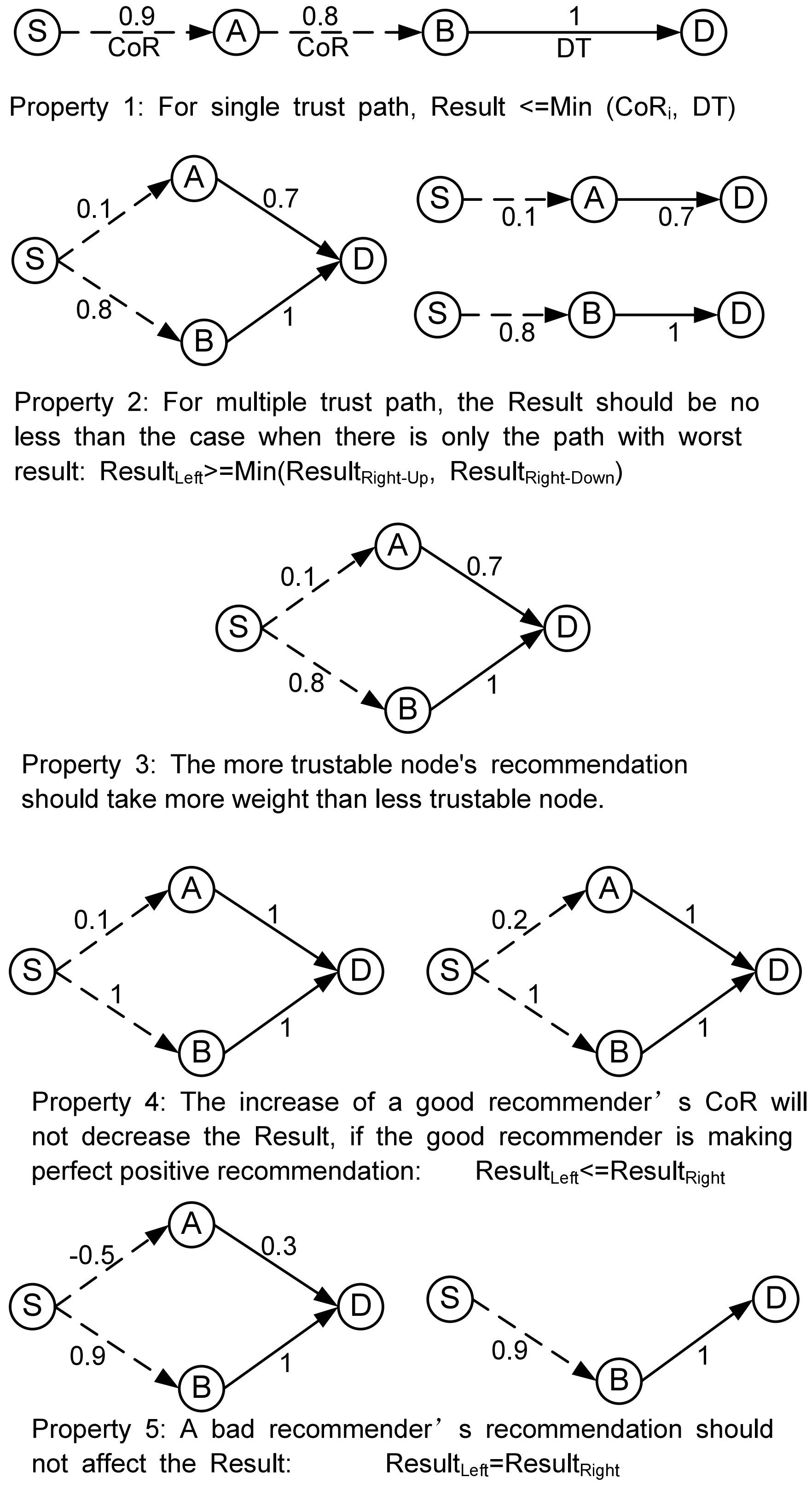
4.3.1 Desired Properties of Trust Propagation

Before proposing a trust propagation algorithm, we first discuss the following desired properties that a trust propagation approach needs to satisfy:

- When there is only a single trust path, the result of trust propagation should not be more than any of the trust values along the path. As shown in Figure 4.9, the calculated result of the first diagram should be no more than CoR_{A-B}, which is the smallest trust value along the single path.

  The meaning of this property is that a propagated trust should not have higher trust value than any of the intermediate trust value. Otherwise it would be overconfident on some entity according to the intermediate recommendations.

- When there are multiple trust paths, the result of trust propagation should be no less than that of the case when only the worst trust path exists. As shown in the second diagram of Figure 4.9, the evaluation result of left trust graph should be more or equal than the value calculated only from path S-A-D.



Property 1: For single trust path, Result <=Min (CoR$_i$, DT)

Property 2: For multiple trust path, the Result should be no less than the case when there is only the path with worst result: $\text{Result}_{\text{Left}}$>=Min($\text{Result}_{\text{Right-Up}}$, $\text{Result}_{\text{Right-Down}}$)

Property 3: The more trustable node's recommendation should take more weight than less trustable node.

Property 4: The increase of a good recommender’ s CoR will not decrease the Result, if the good recommender is making perfect positive recommendation: $\text{Result}_{\text{Left}}$<=$\text{Result}_{\text{Right}}$

Property 5: A bad recommender’ s recommendation should not affect the Result: $\text{Result}_{\text{Left}}$=$\text{Result}_{\text{Right}}$

Figure 4.9: Desired Properties for Trust Propagation Algorithm

This property comes from the understanding that more positive evaluation of trust will not decrease the final trust value by combining them with the existing evaluation.

- When there are multiple trust paths, the recommendation of more trustable node should take more weight than that of less trustable node. In the third diagram of Figure 4.9, the path S-B-D should take more weight than S-A-D, since B is more trustable than A in the view of S.

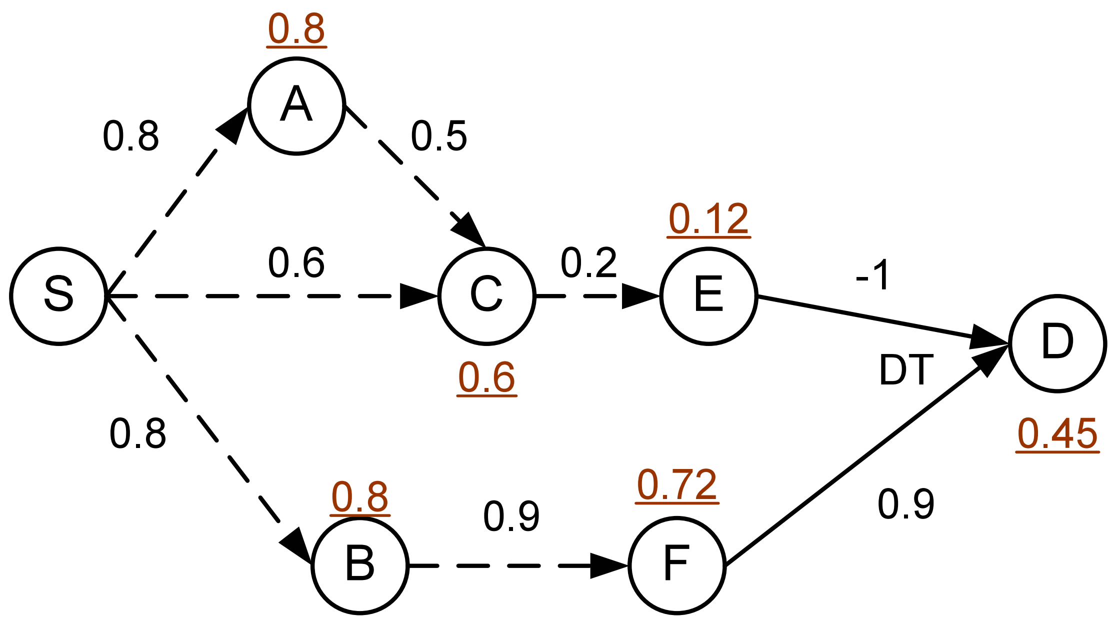
  The underlying principle for this property is simple: the more reliable the source is, the more weight it takes in the final consideration.

- The increase of a good recommender's CoR will not decrease the result of trust propagation, if the recommender is making a perfect recommendation. In the fourth diagram of Figure 4.9, due to the increase of CoR_{S-A}, the evaluation result of the right trust graph should be more or equal than the left one.

  This property simply suggests that when a recommendation node's reliability is increasing, its suggestion should not take the propagation result down if this suggestion is perfect (with a trust value of 1). This simple but reasonable property can not be fulfilled by many current trust propagation approaches.

- A recommender's advice should not affect the result at all if the recommender is not trustable. In the fifth diagram of Figure 4.9, due to the negative CoR_{S-A}, A's recommendation has no value for S. So A's recommendation should not affect the result of propagation. The two trust graphs should have the same evaluation result.

  An intermediate node with negative trust value suggests that the trustor does not believe in it. Thus any recommendation from this node should not be considered. There are some debates on whether we should take this intermediate node's recommendation in the opposite way or should we simple ignore it. We think ignoring this distrustable recommendation is more similar to human behaviors: although your en-



$$Trust_{S\text{-}D}=RP_D=(W_E*DT_{E\text{-}D}+W_F*DT_{F\text{-}D})*Max(RP_E, RP_F)$$

$$W_E=\frac{RP_E}{RP_E+RP_F} \qquad W_F=\frac{RP_F}{RP_E+RP_F}$$

Figure 4.10: Trust Propagation Example

emy's enemy could be your friend, people usually do not rely on the enemy's words to find out if someone is your friend.

4.3.2 Trust Propagation Algorithm

In this section we propose an algorithms to calculate the trust value along the trust paths. This algorithm should meet the properties discussed above. First we need to generate a subgraph from the local trust graph of the trustor, which contains all paths connecting the trustor to the trustee. Different DTs generate different subgraphs. The last hop of the trust paths must be a DT while the rest are CoRs. An example is shown in Figure 4.10. Node E and F have direct trust on node D. Their opinions can be opposite to each other. Also node S also has its CoRs pointing to E and F indirectly, suggesting how E's and F's opinions are trustable. We use this subgraph as an example to discuss our proposed trust propagation algorithm. The algorithm combines the trust values (CoRs and DTs) to yield the trust value of node D in the view of S.

Figure 4.11 presents the pseudo code for our trust propagation algorithm based on modified weighted addition (A simple weighted addition formula can easily be shown that it does not satisfy all the above properties). We define an attribute named Recommendation Point (RP) for every node in the graph (underlined numbers besides each node in Figure 4.10). The RP of one node represents its recommendation validity in the view of node S. Thus S's RP by default is 1. The trust value propagation algorithm is the procedure for S to calculate other nodes' RP from S all the way to D. In this algorithm we simply treat DT the same way as we do for CoR, unless specifically pointing out. A node's RP is calculated as follows:

- When there is only one single edge pointing to the node, this node's RP is calculated by multiplying the previous node's RP and the trust value of the CoR. For example, $RP_A = RP_S * CoR_{S-A} = 1 * 0.8 = 0.8$.
- If there are multiple edges pointing to a node, this node's RP is calculated as follows. First we check if the trustors of these CoRs are of same distance from node S. If they are not, we simply ignore the CoRs from nodes who are further away from S. This "shortest path dominance" rule comes from the intuition that people puts short links over long ones when taking recommendations. Our simulation discussed later shows this assumption is reasonable. In Figure 4.10, to calculate node C's RP, we see that both node S and A have CoR pointing to C. However, since A is one hop away from S while S is 0 hop away from itself, the opinion of A is simply ignored. So $RP_C = RP_S * CoR_{S-C} = 0.6$.

```
Function CalcRP($node_D$)
{
  If ($node_D$==$node_S$)
     return 1;

  Find all the $node_i$ satisfying the following conditions:
       1)$node_i$ trusts $node_D$ and
       2)HopsBetween($node_S$,$node_i$) equals to the minimum hops
         between $node_S$ and any $node_i$

  For all the found $node_i$ do
  {
     $W_i$=CalcRP($node_i$)/(CalcRP($node_1$)+⋯+CalcRP($node_n$));
  }

  $RP_D$=($W_1$*$Trust_{1-D}$+$W_2$*$Trust_{2-D}$+⋯ +$W_n$*$Trust_{n-D}$)*Max(CalcRP($node_i$));

  If($RP_D$<0)
  {
     For all $Trust_{D-j}$
        Delete $Trust_{D-j}$ from the trust graph
  }

  return ($RP_D$);
}
```

□ *$Node_S$ is the node calculating the RP.*
□ *$Trust_{i-j}$=the trust value that node i trusts node j, can be CoR or DT*
□ *HopsBetween(A,B) calculates the shortest hops between node A and B*

Figure 4.11: Pseudo Code for Trust Propagation Algorithm

- If one node's RP is negative, its opinion will be ignored. This is in consistent with Property 5 in Figure 4.9. As a result all the edges starting from this node with negative RP will be deleted from the trust graph. This does not mean that the trust relations with negative trust values should be deleted. A negative CoR does not necessarily imply the trustee of this CoR will have negative RP, especially when there are multiple nodes which put positive reviews on this trustee. In such cases this trustee's recommendation is still useful as long as its calculated RP is positive.

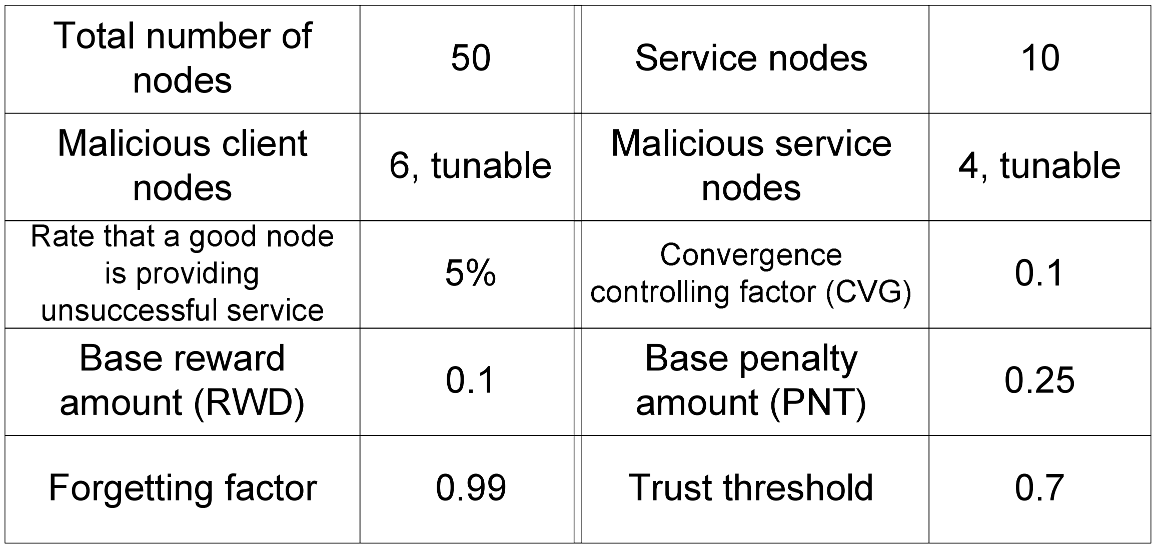
- If there are multiple trust paths to a node with the same shortest length, we use the following algorithm to combine them for calculating its RP: first we do a weighted addition of all the CoR values, with the RPs of the previous hop nodes as the weight; then this result is multiplied by the maximum value of the previous hop nodes' RPs. For example, to calculate the RP of node D, we first add the two DT values from node E and F, with the RP of E and F as the weight; then the result is multiplied by RP_F, since it is bigger than RP_E. Finally when we reach the trustee D, the resulted trust value is the RP of D ($Trust_{S-D} = RP_D$).

Using the examples in Figure 4.9, it can be shown that the algorithm satisfies all the properties in the figure. Also it can be seen that our approach does not endorse the notation that enemy's enemy is a friend. This notation would be true from the view of a probability model. Compared to many current trust propagation algorithms, this modified weighted addition algorithm can fulfill the desired properties we present above.

4.4 Threat Models and Simulation Results

In this section we will evaluate the performance of our system in a distributed service environment. We will analyze several strategies that malicious entities can use to compromise the trust relationships in the environment and evaluate the performance of our trust model against these attacks.

In our simulation, we build a service-oriented computing environment, with a total of



Total number of nodes	50	Service nodes	10
Malicious client nodes	6, tunable	Malicious service nodes	4, tunable
Rate that a good node is providing unsuccessful service	5%	Convergence controlling factor (CVG)	0.1
Base reward amount (RWD)	0.1	Base penalty amount (PNT)	0.25
Forgetting factor	0.99	Trust threshold	0.7

Figure 4.12: Simulation Parameters

50 nodes, 10 of which are service nodes (Figure 4.12). These service nodes can provide up to 5 different types of services. While each service can be provided by multiple nodes, every single service node can also provide multiple services. The default number of bad nodes within the system is set to 10, 4 of which are service nodes and 6 are client nodes. The number of bad nodes is tunable in our simulation in order to test the robustness of our trust model against large number of malicious nodes. The base reward (RWD) and penalty (PNT) amount for a transaction is set as 0.1 and 0.25 respectively and the forgetting factor is set to 0.99. The threshold trust value for a node to be regarded as “trustable" is 0.7. In our simulation, we make the good service node having 95% chance to perform successful transactions, while the bad node will act differently in different attack scenarios.

In the following subsections, we will analyze our simulation results and discuss the impact of simulation parameters’ variations on the results.

4.4.1 No Trust Model

When there is no trust model, the bad service nodes simply do unsuccessful transactions while the good ones perform successful transactions with a high possibility (95%). A client has to randomly pick up a service provider that can do the transaction it needs. The simulation result shows that the transaction success rate is decided by the number of malicious service nodes and their coverage of service types. When there is no bad service node, the success rate is 95%. If the services nodes are all malicious, the rate goes down to 0%. In our simulation set-up, when 4 service nodes are malicious in the default setup, the success rate varies around 60%, depending on the distribution of these bad nodes among different service types. More detailed data are shown in Figure 4.13, for different configurations.

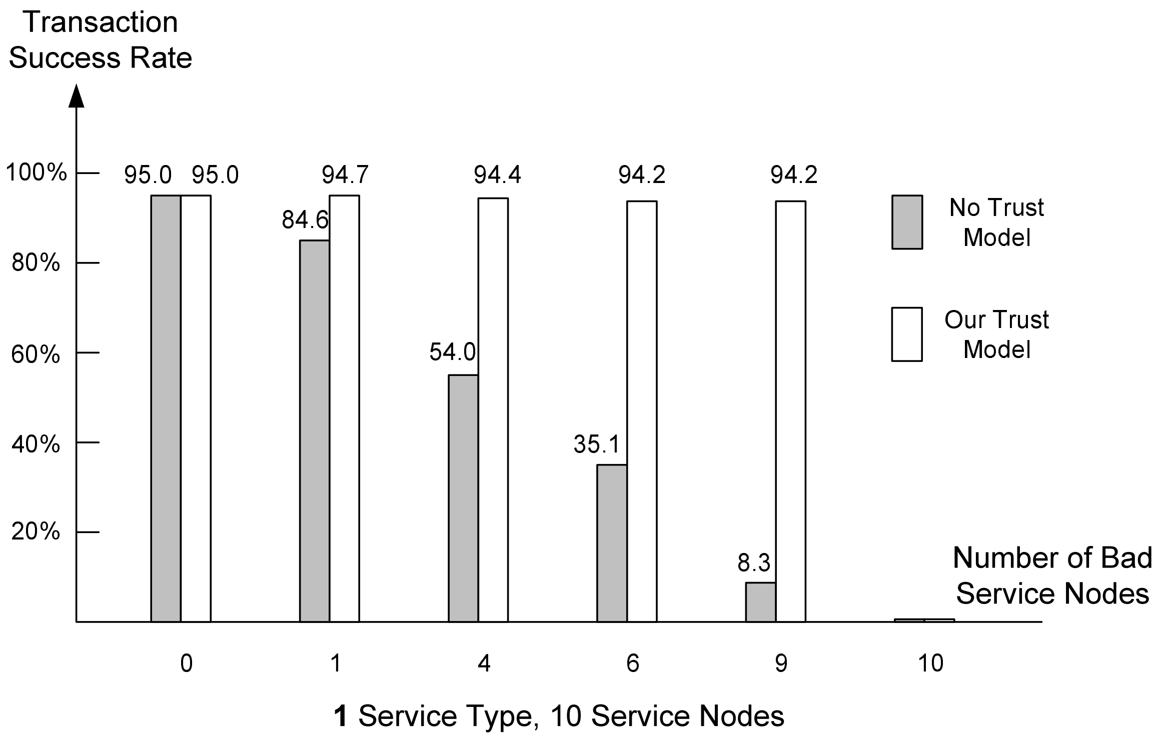
4.4.2 Simulation Result of Our Trust Model

In this subsection, we discuss the simulation of our trust model under a simple malicious environment. We assume the bad nodes act independently: a bad service node simply does unsuccessful transactions for others, and a bad client simply gives useless references for others.

4.4.2.1 Transaction Success Rate

We initialize the system with zero trust value between the 50 computing nodes, which means there is no pre-established trust relation at the beginning. So for the first few transactions, the clients have to do a blind guessing to select a service, without knowing whether the service node is good or bad. But when more service transactions are made, more trust relations will be established and the trust graph is becoming more and more stabilized. The simulation shows that after about 2000 service transactions are made, the system reaches a stabilized state and the transaction success rate becomes stable. The observed stable transaction success rate is around 94%. This suggests that even if 4 out of 10 service nodes are malicious, our trust model can help clients to reach a high transaction success rate. In the following discussions, the transaction success rates are all collected after the system becomes stable, unless specified otherwise.

Figure 4.13 shows the simulation result of our trust model, compared with the case when there is no trust model. First we did a simulation when all 10 service nodes are providing the same type of service. As shown in the figure, when the number of bad service



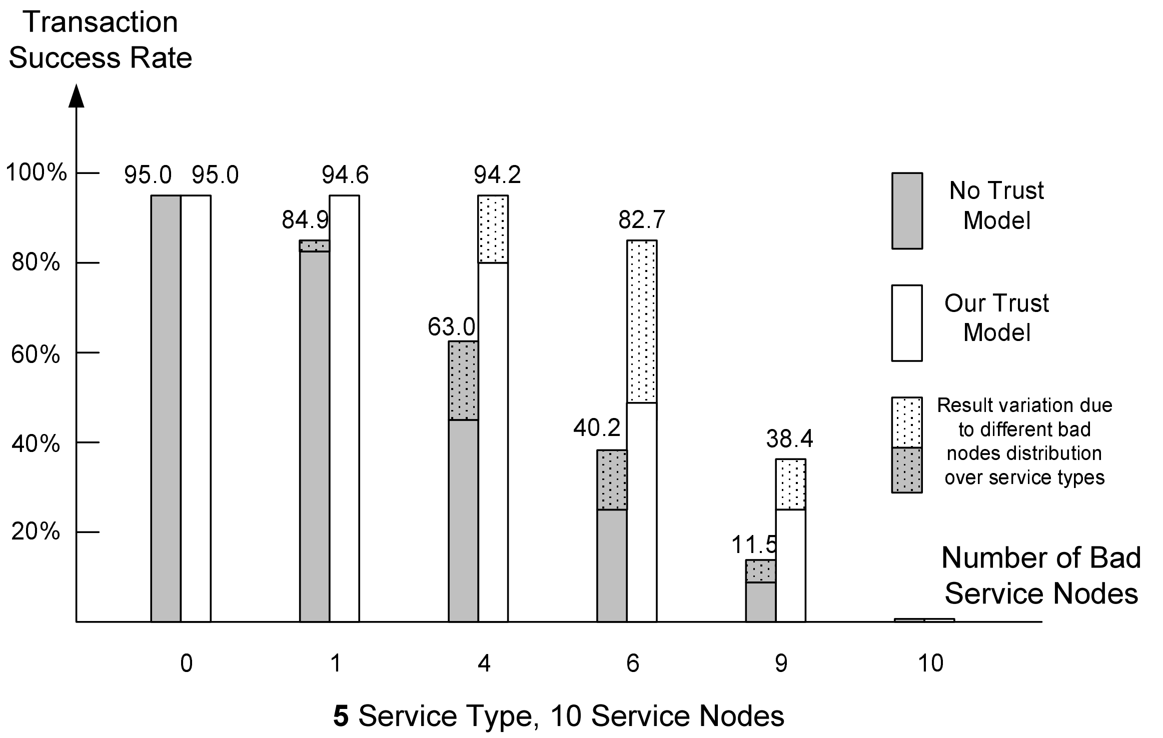


Figure 4.13: Simulation Results With Our Trust Model and Without a Trust Model

nodes are growing, the possibility that a client can receive a good service is going straight down if there is no protection from our trust model. But with our trust model, the services can maintain a very high success rate (around 94%) even though 9 out of 10 of the service nodes are compromised. The transaction success rate is close to 95% because the good nodes have a 5% possibility to make unsuccessful transactions. This suggests that our trust model is robust in tough environment: even when there is only 1 out of 10 service nodes is functioning correctly, our model can help most computing nodes to find that service node.

To further test our trust model, we simulated the system with 10 service nodes providing 5 different type of services, thus 5 different types of DTs are used. The simulation result for small number of bad nodes is similar to the result where all service nodes are providing one type of service. But as the number of bad nodes is growing, the simulation results show some variations from simulation to simulation, as shown in second part of Figure 4.13. The variation comes from the fact that the bad nodes' distribution over service type is different. For example, 4 malicious nodes could cover 100% of one type of service, which makes all transactions of this type of service fail, no matter whether or not a trust model is used. Still our trust model can clearly help the system to achieve a much better overall transaction success rate. For easier presentation and discussion purpose, in the following sections we will analyze the simulation results based on one type of service, unless specified otherwise.

4.4.2.2 Service Load Balance

Besides the transaction success rate, the load balance among service nodes is also an important factor that affects the system performance for most trust based systems. When a trust based system selects a service node among several service providers, if the node with highest trust value will be always selected, eventually one node will accumulate all the trust and gets all the service requests. This will cause a highly unbalanced load for service nodes. In an ideal situation, the service load should be distributed equally to all the good service nodes so that no one is exhausted due to a high load.

We introduced an balancing scheme into our trust model that sacrifices potentially a

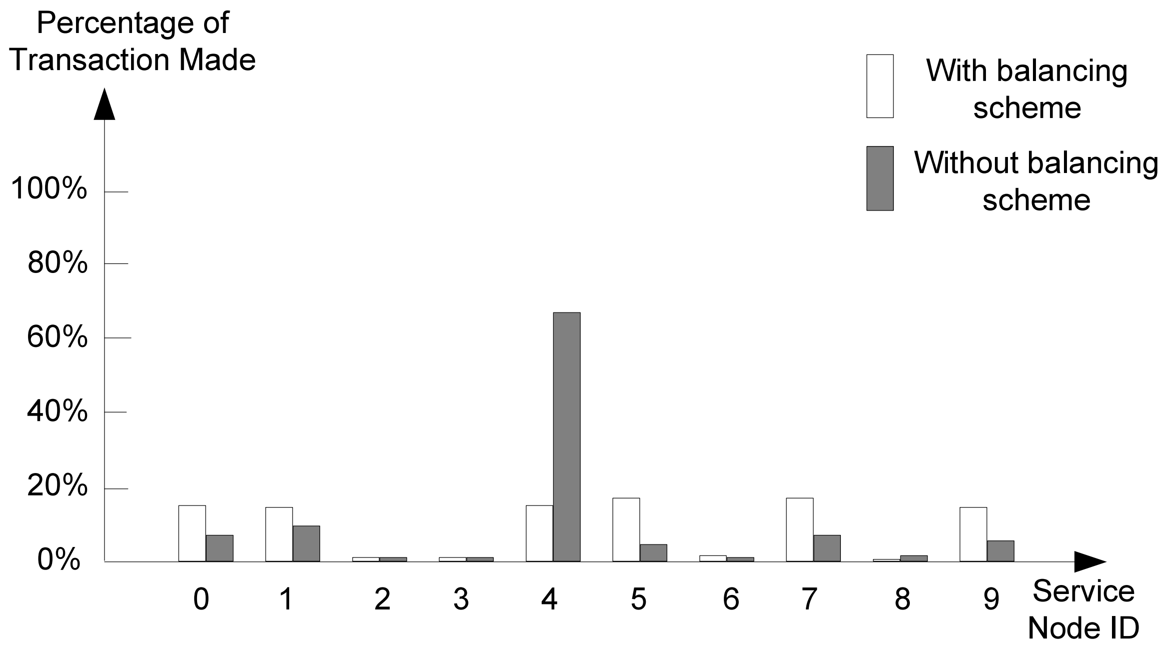


Figure 4.14: Transaction Balance among Service Nodes

small amount of transactions by not selecting the best available node all the time. This would avoid dominance of a few good service nodes, and give new good service nodes a chance to increase their trust values. In our balancing scheme, when several services are available for one transaction, we will first calculate the trust value of these service nodes. Then we look up those nodes whose trust value is higher than the predefined threshold and refer them as “trustable" nodes. Next, if there is any trustable node among the candidates, we have a 95% chance to select the service node from those trustable nodes, with a probability proportional to their trust value: the higher trust value, the bigger chance to be selected. For the other 5% chance, we will select one node from the nodes that are not trustable, also according to their trust values. But if there is no “trustable" node, we will select from all the nodes, again with a probability proportional to the trust value.

Figure 4.14 shows the simulation result of load balance of our balancing scheme: the white bar represents transaction load when our scheme is applied, while the dark bar depicts when there is no such scheme. In both cases the bad nodes (node 2 3 6 8) have low transaction load, which is consistent with the overall high transaction success rate. This

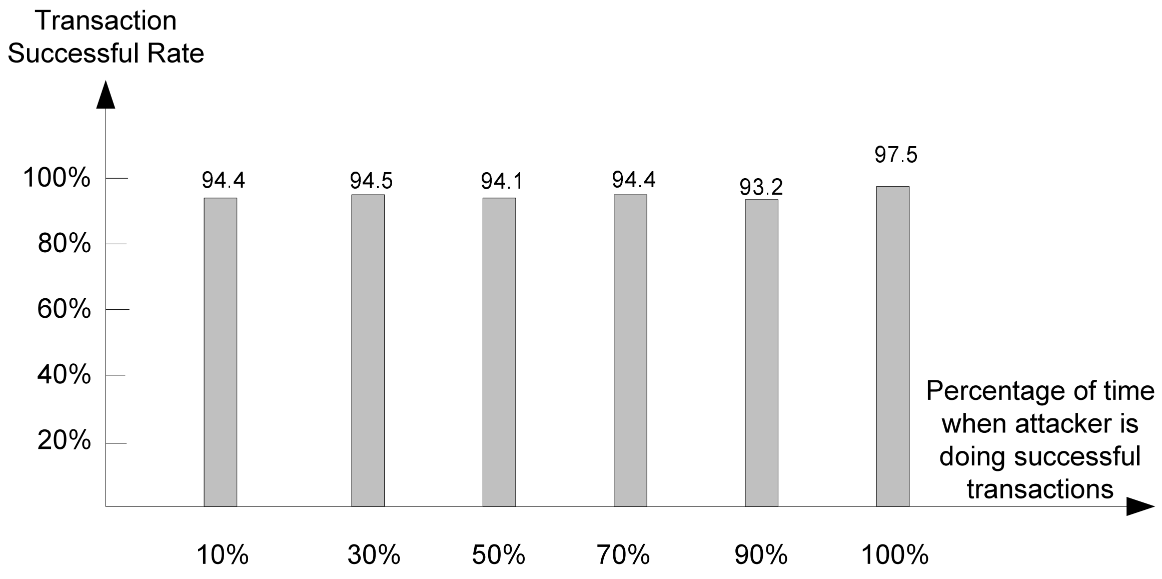


Figure 4.15: Simulation Results for On-Off Attack

figure also shows that our balancing scheme successfully helps the system distribute the transaction load almost equally among the good service nodes. We have performed similar balance simulation on the various attack models discussed later and the results show that our balancing scheme can help balance the service load in all the attack scenarios.

4.4.3 On-Off Attack

In the following sections we are going to discuss several attack models. In these attack models, the attackers are not simply uncooperative. In addition, they try to hide themselves among the good nodes to avoid detection. We divide their attack scheme into several categories, analyze the attack and present our simulation result.

The first attack scenario we will discuss is the on-off attack. In an on-off attack, a bad node will try to hide itself into the crowd and only do bad transactions occasionally, i.e. switching attack mode on and off. By doing this, the attacker is trying to confuse the trust model by making the node performing trust evaluation frustrated since our trust model is based on observation. In our simulation, we fix the number of malicious nodes (4 bad service and 6 bad clients) and tune the percentage of time that the bad nodes do good things in order to hide themselves.

The simulation results (Figure 4.15) show that the transaction success rate remains higher than 93% no matter how much percentage of time the attackers try to behave good to hide themselves, which suggests that our model is highly resistance against on-off attack. In Figure 4.15, the simulation result of attackers being 90% of the time good is slightly lower than the others. The reason is that our trust model could not effectively help the clients recognize the bad nodes when they hide deep enough. However, the overall transaction success rate still remains high since the attackers can not effectively damage the system by hiding too deep (i.e being good most of time). When the attacker performs success transactions with a rate higher than 95%, the observed transaction success rate is even better than where there is no attacking model, because the attackers are behaving even better than the supposed good nodes (with 95% success rate).

4.4.4 Independent Bad Mouthing Attack

The second attack scenario we will discuss is called independent bad mouthing attack. In an independent bad mouthing attack, the bad node (either client or service) will try to compromise the trust system by giving negative trust value to the good nodes. The attacker is expecting to compromise the trust propagation algorithm by adding negative CoRs and DTs into the trust graph. This attack scheme is by natural defeated by our trust model because these malicious nodes will have negative CoRs if they always give useless evaluation. The honest nodes will eventually ignore the opinion of malicious nodes' bad mouthing. Our simulation result shows a transaction success rate of 95% in the default parameter set-up. When we tune the number of malicious nodes higher than the default set-up, the observed transaction success rate remains stably 95%. As we discussed, this independent bad mouthing attack is naturally defeated by our model. But if the attackers can form an alliance and attack the system by gossiping the set of good nodes together, it can create a threat.

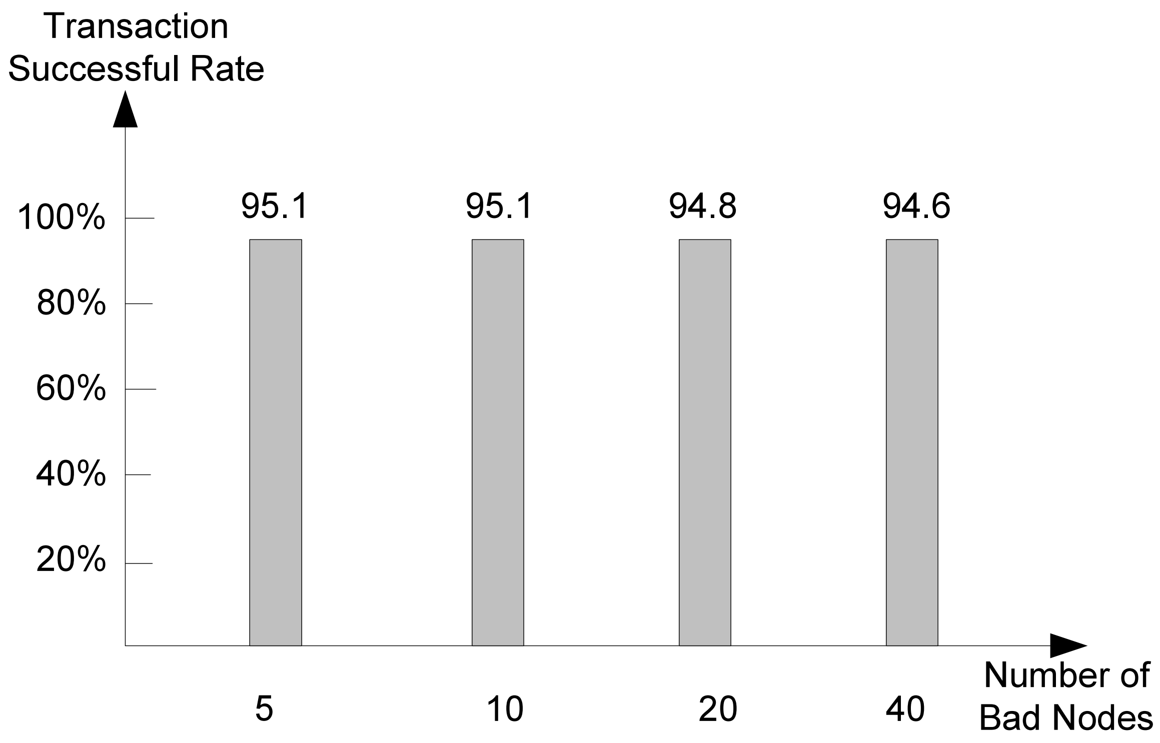


Figure 4.16: Simulation Results for Collaborative Bad Mouthing Attack

4.4.5 Collaborative Bad Mouthing Attack

In this attack scenario, we assume that the bad nodes can cooperate by giving positive reference to each other, while giving negative reference to the good ones. As expected, it is harder to defend than the independent badmouthing attack because once a node accidentally trusts a bad node, the collaborative bad nodes will eventually lead her to a bad service over their malicious references network.

In our simulation, we gradually increase the number of bad clients nodes to test the robustness of our trust model against multiple attackers. The simulation results show a strong resistance of our trust model against the attack. As shown in Figure 4.16, our model achieves 95% transaction success rate, a perfect resistance to this attack. Even when the malicious nodes are dominating the computing system (40 out of 50 client nodes are attackers), the trust model still generates a very high transaction success rate of 94.6%.

4.4.6 Conflict Behavior Attack

The last and most sophisticated attack scenario is the conflict behavior attack. In such attack scenario, the malicious service node will behave well (perform successful transaction, give correct recommendation) to a subset of the nodes while behaving badly to the others. By doing this, the attacker is trying to develop opposite opinions among these two sets of good nodes in order to confuse them and finally to compromise the trust system.

The simulation is done under our default parameter setup, with the number of malicious nodes tuned. In addition, we divide the good nodes equally into to two subsets: one will always receive good services from the malicious service nodes and one will receive unsuccessful service transactions. Our simulation result is displayed in Figure 4.17. The observed transaction success rate shows a strong resistance of our trust model against conflict behavior attack, especially when the number of bad nodes is low. When the bad nodes becomes dominating the system, the good ones will have harder times to find the right service, but still with a transaction success rate over 85% even when majority of the nodes are maliciously forwarding bad recommendations. The observed transaction success rate is higher than when there is no attack model because the malicious nodes are sometimes performing good transactions, which helped enhancing the overall success rate.

4.4.7 Simulation Results for Variations on Trust Calculation Scheme

In our simulations, when our trust system is stabilized, the trust graph becomes a saturated graph where every single client has one DT to every service node. According to our trust calculation scheme (where shortest path has priority), at this stage the trust value calculation are mostly relied on the DTs. This is because we ignore the longer trust path when there are shorter ones. Obviously when there is a DT existing between the trustor and trustee, any CoR will be ignored because it will only lengthen the trust path. So when a system is running for long enough time, the trust value will be stabilized and CoR will be seldom used to calculate the trust value.

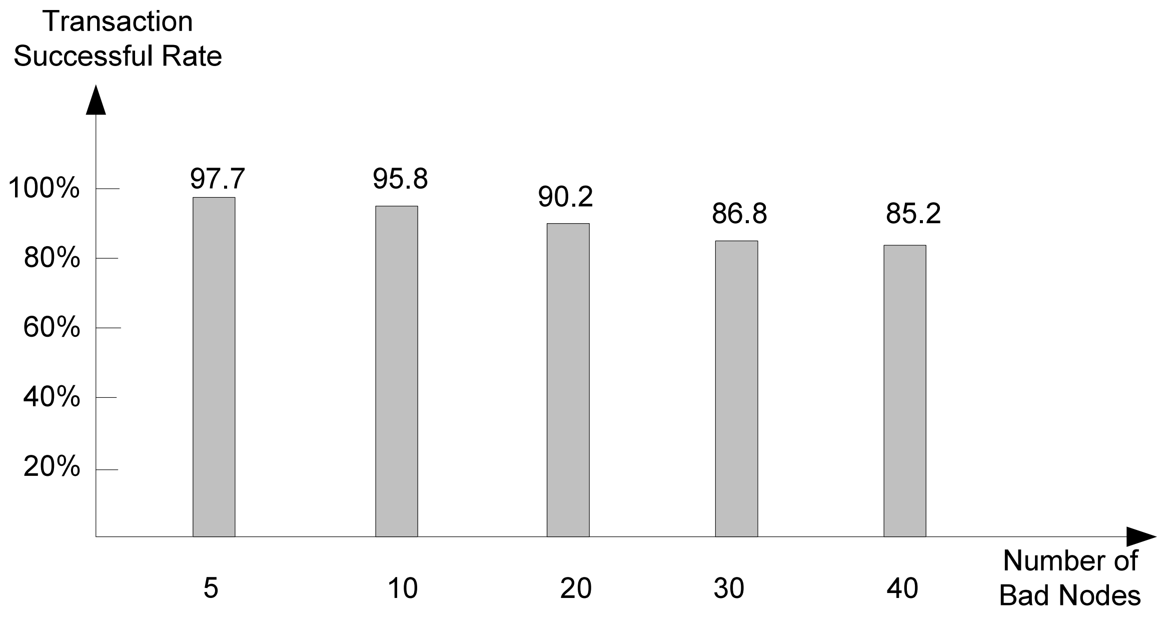


Figure 4.17: Simulation Results for Conflict Behavior Attack

Though CoR may seem to be useless in such scenario, it is not true when it comes to the real pervasive environment, where the computing nodes are mobile and population of the system is dynamic. The simulations conducted above assume a static enclosed environment where computing nodes can not enter or leave the system. We need to further investigate our trust model in the dynamic environment, where the system might not become stabilized before some computing nodes are joining or leaving the system. Our expectation was that CoR would play an important role in such a scenario due to the system's immature nature. To investigate the impact of CoR and make our trust model complete, we design several schemes for trust evaluations:

- Scheme 1: DT only. In this scheme, no CoR will be used. All trust evaluation is only based on the direct trust relation. That is, when a good transaction happens, the client will reward the service node with some DT increment, while penalty will be given to service node doing bad transactions. For consistence, the way they reward and penalize a DT is the same as the other schemes. No recommendation will be made in this scheme, so there is no CoR involved.

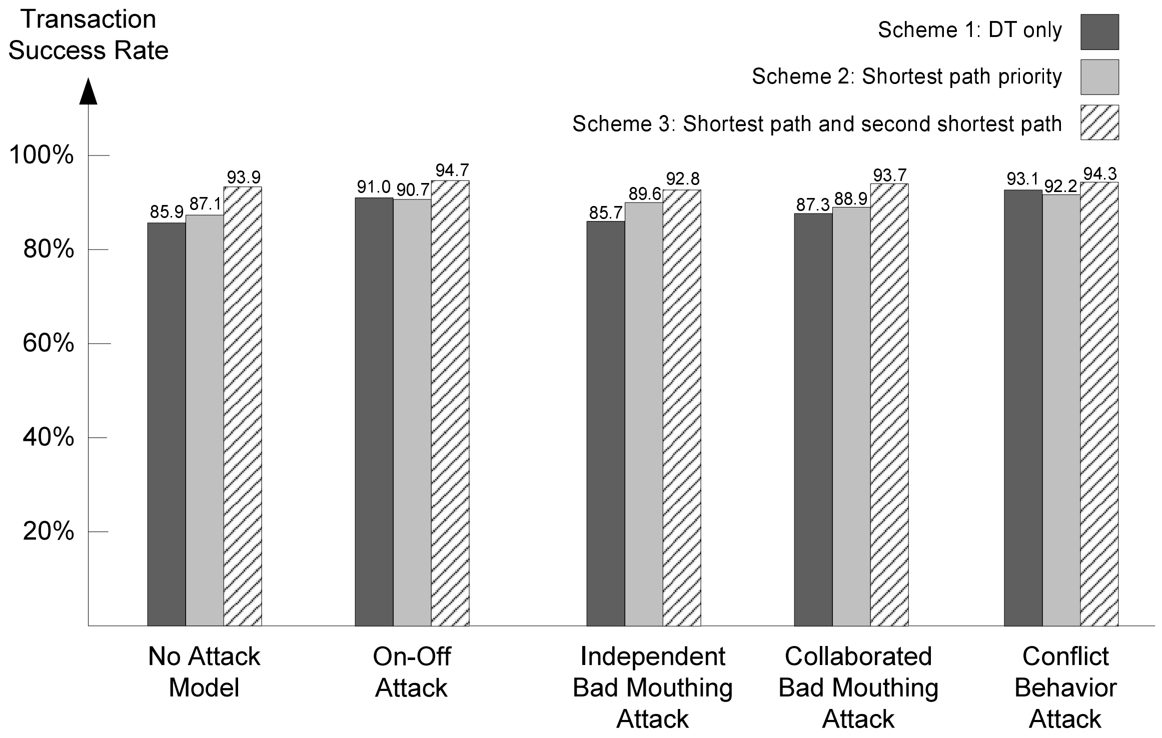


Figure 4.18: Comparison on Variations of Trust Calculation Scheme (Dynamic Environment)

- Scheme 2: Shortest path priority. In this scheme, CoR will be taken into consideration. However, only the trust path with shortest length from the client to service node will be considered. This implies that when the system is running for long enough time and become stabilized, the CoRs will be ignored since there will be DTs between almost every pair of client and service nodes. This is the scheme that we have described and used.

- Scheme 3: Shortest path and second shortest path. This scheme is similar to Scheme 2. The only difference is that, when we calculate the trust value of one node, we will consider the trust path that is shortest and the path that is only one hop longer than the shortest path. This will enforce the client to combine more trust relations, especially CoRs, even when the system is stabilized.

First of all, we simulated all the three schemes under all the attack models we discussed in the static environment, where no node is joining or leaving the environment. The simulation results show that all schemes can generate a very high transaction success rate over 90% under all the attack models. In other words, when the system is stabilized, most computing nodes can recognize the bad nodes no matter which scheme the trust model uses.

Then we simulation in the dynamic environment to study these above three schemes. In the dynamic environment certain percentage of nodes are constantly flowing in and out of the system. We tune the environment such that the nodes are not flowing too fast (otherwise most nodes do not have enough time to establish trust with the newcomers before next group of nodes flows in), or too slow (otherwise it will be the same as a static environment). In such a dynamic environment, it is harder for a client to find a good service because it will be more common that either the client or the service node is new to the system. Figure 4.18 shows the simulation result for such a dynamic computing environment. From this figure we can see that the schemes with CoR involved (Scheme 2 and 3) has higher transaction success rate than that with no CoR (Scheme 1). This demonstrates that CoR is useful to help the system to achieve stabilized state. Also result of Scheme 3 is slightly better result than that of Scheme 2, but the difference is marginal. This suggests that our scheme of choosing the shortest trust path over longer ones is acceptable, while it reduces the computation overhead at each node.

4.4.8 Simulation Summary

Our simulation results show that our trust model can greatly improve the total transaction success rate in a system with malicious nodes. Our model can achieve an almost perfect success rate even if the computing system is dominated by malicious nodes.

When the malicious nodes starts to take strategies to avoid being detected, the simulation shows that our trust model is very robust against these attack strategies. When the number of malicious nodes is not prevalent in the computing environment, our trust model can achieve a almost perfect transaction success rate. When the bad nodes are becoming dominate in the system, our model can achieve considerable mitigation and greatly help the computing nodes finding the best available service in such situations.

CHAPTER 5: RELATED WORK

This work is related to many research areas, including access control, authentication & identification, privacy protection and trust management in distributed systems. Much work has been done in each of these areas, and it is impossible to cover all of them thoroughly in this chapter. In the following, we describe the work most related to ours, especially in the field of authentication in RFID systems, user identification in decentralized systems, trust modeling and management.

5.1 Authentication in RFID Systems

Many cryptographic protocols [19, 29, 32, 36, 57, 61, 66, 77] have been proposed to address the privacy issue. They deploy relatively cheap implementation such as exclusive-OR and one way hash function on tags to achieve secure identity reports. Most of the proposed protocols address the threat of eavesdropping and physical attacks receive very few considerations.

A secret key based protocol is proposed in [57], which mainly deals with the threat of eavesdropping. In this protocol, every single tag pre-shares a secret with the reader system. When the tag receives a query, it will first generate a pseudo random mask with this secret key, apply the mask to its real ID and send it back to the reader. The reader will use this masked ID to locate a corresponding entry in its backend database and authenticate itself to the tag if necessary. This protocol (and many other following approaches [29, 32, 66, 77]) perfectly resists eavesdropping by using a shared secret to a pseudo random function to generate a response that can not be reversed without knowing the secret.

A token based approach named minimalist cryptography is proposed in [36], which is similar to our scheme. This approach follows a token reuse scheme that a token is not discarded after unsuccessful authentication but only recycled to reuse again. Token reuse scheme does not suffer from token exhaustion due to unexpected queries. The main potential threat of this scheme is that an attacker can send a few queries to obtain all the tokens on a tag and use these tokens to track this tag until the tag meets a legitimate reader again. In our proposed scheme, every token is used only once and we use several ways, as discussed in Section 3.1.3.2, to handle unintentional normal queries

The hash-chain protocol [61] is proposed to achieve Level II resistance (As we discussed in Chapter 3.1.2.1) to physical attacks. When the tag receives a query, it hashes a shared secret with two different hash functions and replaces the secret with its hashed result. The tag returns the current hashed value on its hash chain back to the reader as an authenticator. The backend database performs the same hash functions on its entries for each tag to find a match and recognize the tag. Though this protocol has scalability problems, a subsequent approach [19] called time-memory trade-off is proposed to improve the performance. The hash-chain scheme can resist eavesdropping, message hijacking and physical attacks at resistant Level II only.

Our research was motivated by the potential threat of physical attacks. We propose an approach to address this issue while resisting eavesdropping and message hijacking, which was achieved by most current schemes. We adopt a token based scheme to provide a means for the database to recognize the tag while minimizing the impact of tracking a tag by this necessarily shared information. Our proposed protocol resists physical attacks and other forms of attacks such as eavesdropping.

5.2 User Identification in Decentralized Systems

There are many research groups conducting studies on user identifications. Some of them are using authentication technologies [58, 82], in which users provide the system with pre-shared common secret to prove its identity. This research area has been thoroughly studied since cryptography is introduced. But authentication technology does not well fit in our user identification challenge because 1) a user's authentication key can be compromised and thus his/her identity can be stolen and 2) for convenience purposes, many pervasive systems do not even require mandatory authentication.

There are some identification studies based on biological and behavioral techniques. In [21, 64] researchers propose approaches to identify a user by recognizing its typing pattern via keystrokes. The smart floor project [62] use special techniques to identify users by their footstep force profiles. These studies are still progressing on better identification of users, but only biological and social behavior based identifications are insufficient for recognizing a user in pervasive computing environments.

The Honeynet Project [4] proposes a scheme of setting traps to detect and deflect attempts at unauthorized use of information systems for better countermeasures. A Honeypot sets up a virtual environment to trap malicious users to study and analyze their behaviors to better know the enemy. Instead, our system monitors the user behavior in a real computing environment. We analyze the their behaviors to establish malicious user profile to recognize suspicious users in the future.

Intrusion detection technologies [3, 5, 10, 11] are very widely adopted to monitor malicious behaviors. The difference between our approach and intrusion detection technologies is that IDS only collects *malicious* user signatures, while our approach also logs normal system events as user behavior signatures. Thus IDS can only detect an attack when it happens, but our approach can provide an early warning before the actual attacks starts. Moreover, on the perspective of trust management, an IDS treat trust as a binary relation: either trust or distrust. Our user behavior identification approach, instead, treat trust as a continuous variable. The detected *suspicious* behavior of a user is different than confirmed malicious actions detected by IDS, which may not deserve a remarkable trust level degradation.

5.3 Trust Modeling and Negotiations

Many researchers follow the real life intuition that trust is a probability [39, 73, 74] that one can expect the other to behave properly. The advantageous of treating probability as trust is obvious: trust has a physical and mathematical meaning, i.e. the expectation for the event to happen based on previous observations. But simply putting an equal mark between trust and probability makes it hard to fulfill some basic properties that a trust relation should have. For example, an event taking place recently should have more effects on the trust evaluation than that happened long ago. Since probability inherently has to treat every event equally, it has to compromise the physical meaning to make the model practical.

Besides the probability based models, there are many other approaches such as [35, 43] that try to find out the intrinsic nature of the trust relation in order to find a proper way to define and describe the relation. These approaches locate the desired properties that a trust relation should comply with and try to propose a way to model trust to fulfill these properties. Simulation results of these approaches suggests that they are largely advantageous than the system without a trust management. There is no systematical approach to address the desired trust properties and provide a framework to fulfill them.

For those approaches that try to catch the properties of trust by intuition and then model it in a mathematical way, there are many parameters that need to be tuned since there is no strict math foundation for these approaches. The best parameters may change with the application scenarios, such as the population level, malicious user proportion, and many other application factors. We believe it is a better practice to let trust management system to intelligently decide these parameters dynamically during run time. We will investigate this issue further in our future research.

Most approaches in the literature evaluate trust based on the best available information collected by the trustor, which is only a subset of the global trust information. The approach in [39] proposes to calculate the trust value in a centralized view, using the global trust information. Though in many application scenarios, it is too luxurious to have such global information, it still can be acquired in some applications with centralized control. On the other hand, users may be more interested for personalized, distributed trust which can be more useful to users' needs.

CHAPTER 6: CONCLUSIONS

Pervasive computing systems is becoming widespread as the next generation of computing systems as the rapid growth of distributed and open infrastructures, including the Internet, Grid, and wireless networks.

In such a computing environment, computing devices try to weave themselves into the fabric of everyday life until they are indistinguishable from it. Most computation will use resources on demand for communication, information retrieval, and computation, in a manner of taking advantages of pervasive resources, instead of depending on personal computing platforms. Physical and temporal boundaries will fade away and people can access information transparently and freely.

Due to the open infrastructure of pervasive computing spaces, to provide security and trusted coalition is different than traditional static infrastructures such as the Internet. In pervasive computing systems, supporting efficient and trusted dynamic coalitions without security and privacy violations is challenging due to the difficulties of limited resources, dynamic population, wireless communication and so on.

In this thesis, we present a trust supportive architecture to address these challenges and secure the pervasive computing systems. We elaborate our reconfigurable trust frameworks by concreting identity and trust management components. Specifically, we resolved the user authentication challenges in low-cost RFID systems against various attack models including physical attack. We propose a user identification model to help detect a user's true identity without the help of registration. We define a trust model in pervasive computing context and propose trust propagation methods and algorithms to achieve successful trust management.

As we mentioned before, the full deployment of trust support depends on the resolution of many challenging problems. In our future work in this area, we plan to explore the following directions.

1. In pervasive computing systems, different applications may have different understandings of and requirements for trust and trust management. Synchronization between components within the trust supportive framework might be a good start point of future research. We plan to investigate the information exchange between trust establishment module and user identification module. That is, when a malicious user is identified, the trust module can take some response to lower the trust level of that user in accordance with the user identification module's judgment. We are expecting that this cooperative system would generate a better result.

2. Our simulation for user identification is currently based on an attack-free dataset collected by 100 normal users. Though this simulation proves that our user identification model is functioning, further experiments need to be done in order to test how robust our system is against attacker's attempts to hide their identities. Theoretically an attacker can perfectly hide her true identity if she can mimic the behavior of any known good user with no doubt. But this can be very hard because human beings are unique in nature. We plan to apply our model onto a dataset with more aggressive attackers and investigate its performance.

Trust is a very critical component in pervasive computing systems. In this thesis, we present a trust-supportive framework to address the security challenges and privacy concerns to help unknown entities establish trust. With our on-demand-assembled security components such as user authentication & identification, trust negotiation and maintenance collaborating, we believe the pervasive computing system can become more secure and privacy-friendly.

BIBLIOGRAPHY

[1] Boycott against clothes with rfid tags.
http://www.boycottbenetton.com.

[2] Boycott rfid enabled products.
http://www.boycotttesco.com.

[3] Cisco intrusion prevention system.
http://www.cisco.com/en/US/products/sw/secursw/ps2113/index.html.

[4] The honeynet project.
http://www.honeynet.org/.

[5] Mcafee intrushield.
http://www.mcafee.com/us/smb/products/network_intrusion_prevention/index.html.

[6] Protest against rfid.
http://www.spychips.com.

[7] Reality mining project at mit.
http://reality.media.mit.edu/.

[8] Rf-dump.
http://www.rf-dump.org.

[9] Rfc 3164 the bsd syslog protocol.
http://tools.ietf.org/html/rfc3164.

[10] Snort intrusion detection project.
http://www.snort.org.

[11] Symantec intrusion protection.
http://www.symantec.com/Products/enterprise?c=prodcat&refId=1005.

[12] Tamper lab.
http://www.cl.cam.ac.uk/Research/Security/tamper.

[13] W3c document type definition sites.
http://www.w3.org/TR/html401/sgml/dtd.html.

[14] W3c extensible markup language sites.
http://www.w3.org/XML/.

[15] Security technology: Where's the smart money? *The Economist*, 2002.

[16] Y. Mass D. Naor A. Herzberg, J. Mihaeli and Y. Ravid. Access control meets public infrastructure, or: Assigning roles to strangers. *Proceedings of IEEE Symposium Security and Privacy*, 2000.

[17] R. Anderson and M. Kuhn. Tamper resistance- a cautionary note. *Proceedings of the Second Usenix Workshop on Electronic Commerce*, 1996.

[18] G. Avoine and P. Oechslin. Rfid traceability: A multilayer problem. *The 9th International Conference on Financial Cryptography*, 2005.

[19] G. Avoine and P. Oechslin. A scalable and provably secure hash-based rfid protocol. *IEEE International Workshop on Pervasive Computing and Communication Security*, 2005.

[20] C. Bodei, P. Degano, R. Focardi, and C. Priami. Authentication primitives for secure protocol specifications. *Future Gener. Comput. Syst.*, 21(5):645–653, 2005.

[21] M. Brown and S. J. Rogers. User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies*, 1993.

[22] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 1990.

[23] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[24] D. Denning. Information warfare and security. *Addison Wesley - ACM Press Books*, 1999.

[25] E. Ferrari E. Bertino and A.C. Squicciarinii. Trust-x: A peer-to-peer framework for trust establishment. *IEEE Transactions on Knowledge and Data Engingeering*, 2004.

[26] N. Eagle and A. Pentland. Eigenbehaviors: Identifying structure in routine. *MIT Media Lab Vision and Modeling Technical Report*, 2005.

[27] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, 2005.

[28] N. Eagle and A. Pentland. Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing*, 2006.

[29] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for rfid systems using the aes algorithm. *Workshop on Cryptographic Hardware and Embedded Systems*, 2004.

[30] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theor. Comput. Sci.*, 283(2):333–380, 2002.

[31] Alan Harbitter and Daniel A. Menascé. A methodology for analyzing the performance of authentication protocols. *ACM Trans. Inf. Syst. Secur.*, 5(4):458–491, 2002.

[32] D. Henrici and P. Muller. Hash-based enhancement of location privacy for radiofrequency identification devices using varying identifiers. *Workshop on Pervasive Computing and Communications Security*, 2004.

[33] K. Henricksen, J. Indulska, and A. Rakotonirainy. Modeling context information in pervasive computing systems. *Proceedings of First International Conference on Pervasive Computing*, 2002.

[34] G. Hulme. The threat from inside. *Information Week*, 2003.

[35] A. Josang and R. Ismail. The beta reputation system. *Proceedings of 15th Bled Electronics Commerce Conferece*, 2002.

[36] A. Juels. Minimalist cryptography for rfid tags. *Security of Communication Networks (SCN)*, 2004.

[37] A. Juels and R. Pappu. Squealing euros: Privacy protection in rfid-enabled banknotes. *Financial Cryptography*, 2003.

[38] A. Juels, RL Rivest, and M. Szydlo. The blocker tag: Selective blocking of rfid tags for consumer privacy. *Proceedings of ACM Conference on Computer and Communications Security*, 2003.

[39] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. *Proceedings of 12th International World Wide Web Conference*, 2003.

[40] J. Grand (Kingpin). Attacks on and countermeasures for usb hardware token devices. *Proceedings of the Fifth Nordic Workshop on Secure IT Systems*, 2000.

[41] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 1981.

[42] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: theory and practice. *SIGOPS Oper. Syst. Rev.*, 25(5):165–182, 1991.

[43] Z. Liang and W. Shi. Pet: A personalized trust model with reputation and risk evaluation for p2p resource sharing. *Proceedings of Hawaii International Conference on System Sciences*, 2005.

[44] Armin Liebl. Authentication in distributed systems: a bibliography. *SIGOPS Oper. Syst. Rev.*, 27(4):31–41, 1993.

[45] Hung-Yu Lin and Lein Harn. Authentication protocols for personal communication systems. *SIGCOMM Comput. Commun. Rev.*, 25(4):256–261, 1995.

[46] Z. Liu, R. H. Campbell, and M. D. Mickunas. Active security support for active networks. *IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews*, 33(4), 2003.

[47] Z. Liu, T. Joy, and R. Thomson. A dynamic trust model for mobile ad hoc networks. *Proceedings of 10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2004.

[48] Z. Liu and D. Peng. A secure rfid identity reporting protocol for physical attack resistance. *Journal of Communications*, 2006.

[49] Z. Liu and D. Peng. A security middleware architecture for pervasive computing systems. *Proceedings of 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 2006.

[50] Z. Liu and D. Peng. True random number generator in rfid systems against traceability. *IEEE Consumer Communications and Networking Conference (CCNC)*, 2006.

[51] Z. Liu and D. Peng. Towards mobile user identification for trust support. *Mobile Intelligence: When Computational Intelligence Meets Mobile Paradigm*, 2007.

[52] Z. Liu and D. Peng. User behavior identification for trust management in pervasive computing systems. *IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2007.

[53] Z. Liu, D. Xiu, and D. Peng. Agent based automated trust negotiation in pervasive computing environments. *Handbook on Mobile and Ubiquitous Computing: Innovations and Perspectives, American Scientific Publishers*, 2007.

[54] Z. Liu, S. S. Yau, D. Peng, and Y. Yin. A flexible trust model for distributed service infrastructures. *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2008.

[55] Ji Ma and Mehmet A. Orgun. Formalising theories of trust for authentication protocols. *Information Systems Frontiers*, 10(1):19–32, 2008.

[56] D. J. Malan, M. Welsh, and M. D. Smith. Implementing public-key infrastructure for sensor networks. *ACM Transactions on Sensor Networks*, 2008.

[57] D. Molnar and D. Wagner. Privacy and security in library rfid: Issues, practices, and architectures. *Conference on Computer and Communications Security CCS*, 2004.

[58] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 1978.

[59] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[60] B. C Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 1994.

[61] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to a privacy friendly tag. *RFID Privacy Workshop*, 2003.

[62] R. J. Orr and G. D. Abowd. The smart floor: a mechanism for natural user identification and tracking. *Conference on Human Factors in Computing Systems*, 2000.

[63] Dave Otway and Owen Rees. Efficient and timely mutual authentication. *SIGOPS Oper. Syst. Rev.*, 21(1):8–10, 1987.

[64] A. Peacock and M. Wilkerson X. Ke. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2004.

[65] Government report. Radio frequency identification technology in the federal government.
http://www.gao.gov/new.items/d05551.pdf, 2005.

[66] K. Rhee, J. Kwak, S. Kim, and D. Won. Challenge-response based rfid authentication protocol for distributed database environment. *Proceedings of International Conference on Security in Pervasive Computing*, 2005.

[67] M. Roesch. Snort: Lightweight intrusion detection for networks. *Proceedings of USENIX LISA*, 1999.

[68] M. Román, C. Hess, R. Cerqueira, A. Ranganat, R. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, October-December 2002.

[69] R. Roman and C. Alcaraz. Applicability of public key infrastructures in wireless sensor networks. *European PKI Workshop*, 2007.

[70] D. Samyde, S. Skorobogatov, R. Anderson, and J. Quisquater. On a new way to read data from memory. *First International IEEE Security in Storage Workshop*, 2002.

[71] E. Shaw, K.G. Ruby, and J.M. Post. The insider threat to information systems. *Security Awareness Bulletin*, 1998.

[72] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. *International Workshop on Security Protocols*, 1999.

[73] Y. Sun, Z. Han, W. Yu, , and K. J. Ray Liu. A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. *Proceedings of IEEE INFOCOM*, 2006.

[74] Y. Sun and Y. Yang. Trust establishment in distributed networks: Analysis and modeling. *IEEE International Conference on Communications*, 2007.

[75] Zhi-Jia Tzeng and Wen-Guey Tzeng. Authentication of mobile users in third generation mobile systems. *Wirel. Pers. Commun.*, 16(1):35–50, 2001.

[76] S. H. Weigart. Physical security devices for computer subsystems: A survey of attacks and defenses. *Workshop on Cryptographic Hardware and Embedded Systems*, 2000.

[77] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. *Security in Pervasive Computing*, 2003.

[78] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–104, 1991.

[79] Thomas Y. C. Woo and Simon S. Lam. Authentication for distributed systems. *Computer*, 25(1):39–52, 1992.

[80] Thomas Y. C. Woo and Simon S. Lam. A lesson on authentication protocol design. *SIGOPS Oper. Syst. Rev.*, 28(3):24–37, 1994.

[81] D. Xiu and Z. Liu. A formal definition for trust in distributed systems. *The 8th Information Security Conference (ISC)*, 2005.

[82] T. Ylonen. Ssh-secure login connections over the internet. *Proceedings of the 6th USENIX Security Symposium*, 1996.

[83] J. Yoshida. Euro banknotes to embed rfid chips by 2005. *EE Times*, 2001.

[84] Chang N. Zhang and Chunren Lai. A systematic approach for encryption and authentication with fault tolerance. *Comput. Netw.*, 45(2):143–154, 2004.

[85] X. Zhang and B. King. Integrity improvements to an rfid privacy protection protocol for anti-counterfeiting. *The 8th Information Security Conference (ISC)*, 2005.