UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

A RISK AND TRUST SECURITY FRAMEWORK FOR THE PERVASIVE MOBILE ENVIRONMENT

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

CARLOS A. SANCHEZ
Norman, Oklahoma
2013

A RISK AND TRUST SECURITY FRAMEWORK FOR THE PERVASIVE MOBILE ENVIRONMENT

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

_____
Dr. Amy McGovern, Chair

_____
Dr. Lee Williams

_____
Dr. Andrew Fagg

_____
Dr. Deborah Trytten

_____
Dr. Dean Hougen

ACKNOWLEDGMENTS

I want to gratefully thank every one who contributed to this project. In particular, I want to express my deep gratitude to my advisor Dr. Amy McGovern for her guidance, support, encouragement, and patience.

Also, I would like to thank Dr. Lee Williams, Dr. Deborah Trytten, Dr. Andrew Fagg and Dr. Dean Hougen for their support. My sincere appreciation is extended to them for taking the time to serve on my dissertation committee.

I am grateful to the University of Oklahoma and its resources as well as all the School of Computer Science staff

A great deal of thanks shall be extended to all people who collaborated in some technical details during the research leading to this dissertation. Especially I want to thank my former colleagues at RiskMetrics particularly Lee Wise, Gifford Webber and Peter Benson for their contribution, guidance and comments in the formulation of the security risk and trust framework. I also would like to offer my gratitude to my current colleagues at NextThought especially to Jason Madden, Chris Utz and Tom Stockdale for their patience and encouragement. I want to express my deep appreciation to not only my former committee chair Dr. Le Gruenwald who was instrumental in developing this dissertation, but also to Dr. Sudarshan Dhall, Dr. S. Lakshmivarahan, and Dr. Sridhar Radhakrishnan for their constant support.

And finally, I dedicate this work to my twin brother Mauricio, my parents Saulo and Amalfi, my son Santiago, my niece MariaJose and my nephew Samuel. My sincere gratitude is expressed to them for their support, sacrifice, help, and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

ABSTRACT

A pervasive mobile computing environment is typically composed of multiple fixed and mobile entities that interact autonomously with each other with very little central control. Many of these interactions may occur between entities that have not interacted with each other previously. Conventional security models are inadequate for regulating access to data and services, especially when the identities of a dynamic and growing community of entities are not known in advance. In order to cope with this drawback, entities may rely on context data to make security and trust decisions. However, risk is introduced in this process due to the variability and uncertainty of context information. Moreover, by the time the decisions are made, the context data may have already changed and, in which case, the security decisions could become invalid.

With this in mind, our goal is to develop mechanisms or models, to aid trust decision-making by an entity or agent (the truster), when the consequences of its decisions depend on context information from other agents (the trustees). To achieve this, in this dissertation, we have developed *ContextTrust* a framework to not only compute the risk associated with a context variable, but also to derive a trust measure for context data producing agents. To compute the context data risk, *ContextTrust* uses Monte Carlo based method to model the behavior of a context variable. Moreover, *ContextTrust* makes use of time series classifiers and other simple statistical measures to derive an entity trust value.

We conducted empirical analyses to evaluate the performance of *ContextTrust* using two real life data sets. The evaluation results show that *ContextTrust* can be effective in helping entities render security decisions.

Chapter 1. INTRODUCTION

1.1 Problem Statement

The distinguishing characteristic of pervasive mobile environments is the wireless communication medium. This communication vehicle is used to transport data amongst mobile users and static computer systems. In today's busy, technology dominating and communication intensive business environments, wireless computing offer numerous possibilities. For example, mobile users can not only browse and use information from the World Wide Web and access different services based on their current locations, but also obtain real time information with the advent of RFID tags [Want, 06] and sensor networks [Chong, 03]. In addition, users (or their system representation such as agents, programs or objects) can enter or leave the system and interact with each other with minimal efforts [Mostefaoui, 04]. In this new evolving pervasive scenario, the amount of data available along with context-aware services to access such data must be protected from malicious or accidental abuse in order to avoid harmful consequences.

In the last decades we have seen a transformation from an industrial society to an information society. We have seen tremendous growth in the areas of telecommunications and, especially, mobile communications. As such, information, which now has become one of the most important resources of our society, plays a vital role in the day-to-day life of many people. Moreover, as research and technology improve, it is now possible for mobile users to access information from multiple sources anywhere at any time. However, this information must be protected from malicious or accidental abuse in order to avoid harmful consequences. It is for this reason that the

provision of security services for the pervasive ad-hoc computing environments is crucial for the adoption and use of mobile technologies. In a pervasive mobile computing environment, context-aware services are required to be non-intrusive and easily adaptable to a changing environment or *context* [Hulsebosch, 05] , as they can be deployed at and referenced by multiple types of static and mobile units (PDAs, laptops, servers, etc…). A [security] *context* is defined as *"a set of information collected from an entity (user or application) and that is relevant to a task or to the security infrastructure of it"* [Mostefaoui, 04]. In addition, these services need to include and build the notion of *trust* (*risk*) since many interactions and collaborations may take place amongst entities that are alien or unknown to one another. This trust measure calculated by the services is then used to grant or deny access to a particular piece of data. Nowadays, the concept of *trust (risk)* is used in applications such as:

- *Commercial systems*: In these systems, buyers need to make sure (*trust*) that sellers will deliver the agreed items. Moreover, sellers need to trust that buyers will actually pay for services [Ryutev, 05]. Finally sellers and buyers use recommendations (assertion claims) made by others to improve the quality of the goods provided.
- *Location-based systems*: In these applications, service access policies are designed to restrict or allow access to data based on the current or future position of an accessing entity. For example, access to battlefield information by a platoon leader depends on whether or not he is line-of-sight of his company commander. As the platoon leader moves away from the company commander, access to information become more restricted as the

risk of battle and capture increases.

- *Coalition-based systems*. In these systems, *ad-hoc* groups of mobile and fixed stations can be established at any given moment in order to provide solutions to a series of problems. For instance, in a natural disaster, several governmental and non-profit organizations may converge to the trouble site in order to provide logistical and relief services. In this case, *trust* must be built and refined rapidly amongst these organizations as the situation on the ground develops.

However, traditional security implementations such as access control lists (ACL) [Swift, 02] and role base access controls (RBAC) ([Neumann, 01], [Park, 04]) are ill equipped to handle security operations that include the user or service context. This is due to fact that the characteristics of the mobile and pervasive environments affect the conventional functioning of a security implementation by introducing a new series of issues. For instance, in a conventional environment, there is a central infrastructure that not only keeps references and serves as storage of all permission information for every user and service in the system, but also makes all security related decisions. However, in a pervasive environment, it would be unfeasible to keep a reference to every entity since there could be a great number of them at any given time. Moreover, these entities could not only enter and leave the system, but also interact (collaborate) with one another at any time. In addition, traditional security implementations are ill equipped to handle security operations that include the user or service context because the definition of policies and enforcement of security policies do not incorporate the dynamic nature of context variables, such as location, network

status, and connection type. Furthermore, such dynamic nature introduces uncertainty, as it is not possible in all cases to capture all the necessary context knowledge to make a reliable security decision. Moreover, once the security decision has been made, current security models do not establish a time line procedure to revaluate the security decision to account for context data changes (i.e. the security decision would stand firm regardless of future changes in the context data). Also, by failing to predict future values in the context data over a time period (*horizon*), current security models are unable not only to run what-if scenarios (to aid them in the security decision making process), but also to anticipate situations where the security decision become invalid due to the change in context data. Therefore, in order to determine a potential negative impact that the dynamicity of context data may have on security decisions, it becomes necessary to introduce risk measures into the security implementation.

The above observations motivate us to revisit to revisit the problem of access controls, trust and risk in the pervasive ad-hoc environment. As such, we need to identify the issues that a security system, which depends on context information to render a security decision, must address.

Before describing the new security issues that arises in pervasive mobile environments, we adopt the following two concepts that will help us understand the issues and systems described in this paper:

Trust

According [Jøsang, 04] Trust is defined as "the extent to which one entity is willing to depend on another entity in a given situation with a feeling of relative security, even though negative consequences are possible."

Risk

The word “risk” refers to situations in which it is possible but not certain that some undesirable event will occur. In everyday usage, ‘risk’ is often used synonymously with ‘probability’ of an unwanted event that may or may not occur [Hansson, 07].

1.2 Security Implementation Issues

In the pervasive ad-hoc environment, the uncertainty and variability of context information raises new issues that must be cope with by a security implementation that relies on such data to render a security decision. In the following, those issues are discussed.

1.2.1 Invalid Context Data Claims

In the pervasive ad-hoc environment users can access services, which are dynamically customized according to their contextual data (i.e. location, identity, etc.). For example, a soldier could request battlefield information, which is personalized according to his current geographical position and rank. As he moves about in the field, this information is updated according to his new location. Furthermore, his rank (or identity) will restrict the type and amount of data he can see. However, a malicious entity could make bogus claims about some of its context information in order get access to a service. For instance, a traveler can override a GPRS/GSM receiver in order to produce invalid location claims with the aim of disrupting an emergency service (like 911) at a particular location. Furthermore, a commercial truck equipped with a GPS receiver (which is used for tracking) can be seized and its GPS receiver tampered with

in order to generate false location and identity claims allowing the theft of its cargo or generating disruptions or delays at a shipping or production point.

1.2.2 Inclusion of Context Data History

As users operate and interact in the pervasive environment, their context variables (e.g. location, velocity, network connectivity, etc.) may change with time. This change introduces uncertainty. For instance, in a location based service, a user reports his/her current location at different times (t_o, t_1, t_2…) in order to access information of interest; however, since there may be delays (due to, for instance, network connectivity or atmospheric noise) between the time the user sends the location information and the time the location service receives it, the user may have moved to a different position. The question that arises is whether this new location is still valid for the service access (permission) being requested (evaluated). In this case, the location service can use the user reported location history to not only predict future locations, but more importantly to measure the degree of change (*volatility*) of the context variable (location) in order to arrive at a more accurate security decision. For example, if the network connectivity of user changes sporadically, the security implementation of a system may decide to deny access or change the access type to a critical object since there may not be a guarantee that the object would be read or updated correctly due to connection issues.

1.2.3 Risk Evaluation Based on Context Data

Due to the dynamicity of the context *risk factors* (variables) in the pervasive ad-hoc environment, a model to measure the risk incurred when evaluating a security

policy must be developed. This means that, given a series of context variables (in a time series form), the model should produce a series of estimates to measure the risk and the likely loss incurred if access is granted erroneously.

1.2.4 Evaluation Horizon

Since user context variables may change in an uncertain way, context based security policies must establish time windows where permissions in the policy are valid (after being acquired) before doing a policy revaluation. For instance, a soldier accessing information about friendly units in the vicinity of his current location becomes a casualty and his equipment falls in enemy hands. If the system security implementation does not enforce a time window for security policy reevaluation (which requires the soldier's identity and password), enemy forces could either continue accessing the information or do some signal intelligence.

1.2.5 Stress & What-If Analysis

Due to the nature of the pervasive ad-hoc environment, it is not always possible to have up-to-date context information when evaluating a security policy. It then becomes necessary to use a simulation framework where context *risk factors* (variables) can be modified in order to measure the *risk* (degree of trust) of an access control policy under different situations. Moreover, what-if & stress analysis can be used not only to predict future scenarios but also to reproduce past situations. For instance, if the location and velocity history for a traveling user is stored, a security implementation could predict – using the *variability (or volatility)* of the stored data - the possible locations of the user and come up with a more suitable security decision.

1.3 Research Objective

A key part in the implementation of a security strategy in a pervasive ad-hoc environment is to understand the complexity, variation and integrity of the data to be protected, in addition, the dynamic environment under which users and applications operate. This dynamicity along with the inner nature of the pervasive environment introduces uncertainty as not only an entity may interact with other unknown entities, but also it could access services and data based on its contextual information, which may not be complete or fully reliable. This uncertainty demands that *risk or trust* measures for the context data be developed in order to arrive at a more accurate security decision. *The objective of this research is to develop a context-based trust security framework for the pervasive ad-hoc environment.*

In order to achieve this goal, the following subtasks need to be completed.

1. Design a system architecture that supports a context-based trust security in the pervasive ad-hoc environment.
2. Determine the following in order to facilitate the inclusion of context data history in the security model:
    a. Provide definitions for different *context risk factors* along with examples to measure them.
    b. Give context data time series definition and operators.
    c. Establish the number of items to keep in the history.
3. Develop a new access control grammar to either describe or expand a security policy using context risk factors.
4. Develop not only risk and trust measures but also algorithms for security

policy evaluation.

5. Define a method to mitigate the impact of invalid context data claims.
6. Define a framework that allows not only back-testing analysis, but also assessing the accuracy of the model.
7. Construct of a prototype for the system and conduct performance evaluation studies using the developed prototype.

1.3.1 Models for Trust and Risk

The question regarding whether adequate models of risk and trust can be designed is still open at the present time [Jøsang, 04]. However, in many expert risk evaluation schemes, risk is measured by describing the relationship between the expected losses that can be caused by a risky event and to the probability of this event. Moreover, to the best of our knowledge only one of the current research systems explicitly takes the risk factor into account [Jøsang, 04]. Furthermore, in most trust systems (i.e. [Patel, 05]) the user must explicitly handle the relationship between risk and trust. That is, risk and trust assessments depend only on user (entity) interactions and do not include the costs for such transactions. Furthermore, transaction-cost models do not take into account relevant context variables, which may greatly affect the risk of a transaction.

In the pervasive environment, the outcome of many transactions may depend on context factors and due to the uncertainty and dynamicity of the context data risk and cost may increase. Since risk and costs are involved, we could reach out to research and developments done in the finance industry to measure them. In this financial setting, risk is measured by "*degree of uncertainty of future net returns*" [RMG, 96]. Moreover,

risk is classified based on the source of the underlying uncertainty [RMG, 96]. For instance, *Credit Risk* measures the potential loss because of the inability of an entity to meet its obligations. Another classification is *Market Risk*, which takes in the uncertainty of future earnings resulting from changes in market conditions, (e.g., prices of assets, interest rates). Furthermore, over the last few years measures of *Market Risk* have become synonymous with the term Value-at-Risk (VaR) [RMG, 96]. By using a *Market Risk* model we could then try to measure the uncertainty and changes in context *risk factors* such as location, identity, velocity, network/power resources, time, suspicion level, user intention, usage history/patterns, etc.

1.3.2 Dissertation Overview

This dissertation address many issues associated with the design of a context-based trust and risk security framework for a pervasive ah-hoc environment. The approach taken in this work is to adapt and use the Value-at-Risk (VaR) methodology in order to give an appropriate measure of *risk* using contextual information with the aim of providing, supporting, enhancing and reasoning about access controls. In this approach, access control policies are expanded with given thresholds for different *risk* and trust measures and context risk factors. Moreover, this framework can be used to provide *feedback* to users when requests are granted or denied. In addition, this approach could allow us to *decrease the risk* posed by fraudulent generation of context data, and yield a way to *reproduce or simulate* the dynamics or volatility of such data [Hulsebosch, 05]. Finally, this framework relies heavily on history of context data in order to yield a more accurate risk measure.

The rest of this dissertation is organized as follows. Chapter 2 provides a

through description of current research work in the area trust and context based security. In addition, it also provides an introduction of database security concepts, risk measures and time series classification. Chapter 3 proposes a trust based security framework that includes context data for the pervasive ad-hoc environment. Chapter 4 provides a description of the risk model used for security decisions. Chapter 5 offers a full explanation of entity the trust model used in this system (*ContextTrust*). Chapter 6 provides some numerical and performance analysis for *ContextTrust*. Finally, Chapter 7 postulates some conclusions and future work.

Chapter 2. LITERATURE REVIEW

In this chapter, a review of relevant literature is presented. First, database security concepts are introduced and then followed by a description of relevant context and trust based security systems. It continues with a description of ways to model variables. Finally, an introduction to time series classification is offered.

2.1 Database Security Concepts

As mentioned in chapter 1, the goal of this research in the introduction of a context and trust based security framework for the pervasive ad-hoc environment. In order to achieve this goal we need to examine the different aspects of information security. This section introduces the concept of data security, access rights, authorization and the different security models found in the literature.

Information Security

Information security is concerned with ensuring privacy, secrecy, integrity and availability of the produced and stored data. This is often discussed in terms of the need to restrict access to information in data repositories and is called access control.

Access Rights

Access rights are premises used to either allow or prevent users (or processes running on behalf of them) from executing a particular operation over an object (file, table, method, etc.) [Dittrich, 94]. Access controls are defined based on a security policy, which is a set of rules used to manage, protect and distribute sensitive data [Pernul, 94]. In general, security policies are stated in terms of security subjects and

security objects. Security subjects can be classified into three categories: 1) users or processes running on behalf of users; 2) roles, which are abstract users that reflect organizational or functional positions in a particular environment; and 3) security domains or nested sets of subjects used to define general control policies [Dittrich, 95]. Security objects can be classified as objects (tables, attributes, services, etc…) and protection domains (set of objects) used to define more abstract protection objects.

Authorization

An authorization scheme describes the methodologies used to grant, deny and delegate access to data among security subjects. In addition, it stipulates the steps to follow to resolve access control conflicts when they arise [Essmayr, 95]. Basically, authorization allows us to know which requests are permitted and which ones are not. To better describe the authorization mechanism of a system, the following concepts are introduced ([Dittrich, 94a], [Dittrich, 95]):

a) Authorization Paradigm: Authorizations to an object can be set and/or revoked by either the object's owner or the system administrator. In the former case (ownership paradigm) access to an object are based on the owner's judgment. In the latter case (centralized administration), a central authority, which delegates access to objects, enforces securities policies.

b) Kinds of Authorization: Authorization schemes can be described as positive, negative or mixed systems. A positive scheme means that the system only gives permissions, whereas in a negative model, the system only specifies prohibitions. A mixed paradigm includes permissions and prohibitions and requires a set of established rules to resolve conflicts.

c) Transmitted Authorization: Systems implement different mechanisms that not only allow the assignment of default access rights to new objects but also to permit security administrators the propagation of permissions and prohibitions assigned to security subjects.

Discretionary Access Controls (DAC)

The Telecom Glossary[1] defines Discretionary Access Controls (DAC) as "*A means of restricting access to objects based on the identity and need-to-know of users and/or groups to which the object belongs. Controls are discretionary in the sense that a subject with certain access permission is capable of passing that permission (directly or indirectly) to any other subject.*" Formally a DAC is defined as follows [Pernul, 94]: Given a set of objects *O* (entities to be protected), a set of subjects *S*, a set of access operations *T* and a set of privileges *P* (content-based access rules), a tuple ***<o, s, t, p, an, g, op>*** is defined to decide if an operation *t* to a subject *s* is allowed *(an = +)* or denied *(an = -)* for the object *o* within the range defined by predicate *p*. The component *g* is the grantor of the authorization and *op* indicates if *s* could grant *t* to other users.

Most systems supporting *DAC* store access rules in an access control matrix. The rows of this matrix represent subjects; the columns denote the objects and the intersections specify the access types (permissions or prohibitions) that subjects have over the corresponding objects [Dittrich, 94a]. However, in many cases, the set of access rights is not complete, that is not all the intersections in the access control matrix are specified. Consequently, there may be some subject requests for which neither a prohibition nor permission was granted. Hence a closure assumption is necessary

[1] http://www.atis.org/

[Dittrich, 95]:

- Closed world assumption: a request is forbidden unless an appropriate permission exists.
- Open world assumption: a request is allowed unless an appropriate prohibition exists.

Mandatory Access Controls (MAC)

According to [Pernul, 94], Mandatory Access Controls are concerned with the flow of information within a system in addition to enforcing access control to information. They require security levels (labels) to be assigned to objects and subjects. A label for an object is called classification and a label for a subject is called clearance. A label consists of two components: a level from a hierarchical list of sensitivity levels or access classes (e.g. *Secret* > *Confidential*, *Unclassified*) and a member of a non-hierarchical set of categories representing classes of object types (e.g. Development, Research Production). Clearance and classification levels are totally ordered, thus, the resulting labels are partially ordered. That is, for label c_1 to dominate a label c_2, c_1*'s* level must be greater than or equal to that of c_2 and categories of c_1 must contain all those of c_2. For example, c_1 = *(Secret, [NATO, ARMY])* and c_2 = *(confidential, [ARMY])* then $c_1 > c_2$, since the sensitivity level of c_1 is greater than c_2's (*Secret* > *Confidential*) and c_1's categories encompasses the ones for c_2.

One of the most widely known mandatory access controls models is called multilevel security which is based on the work done by Bell and LaPadula [Bell, 73] and is formalized in two basic rules [Pernul, 94]:

- Subject *s* can read an object o if Clearance(*s*) ≥ Classification(*o*)

- Subject *s* can write on an object *o* if Clearance (*s*) ≥ Classification (*o*)

Among other designs we could mention is the Biba model [Biba, 77] where, besides protecting information from unauthorized flow, also protects the information from authorized change by imposing strict integrity constraints. In addition to the Biba model, the Dion model [Dion, 81] combines the principles of secrecy of the Bell and LaPadula model with the integrity constraint policy from the Biba model without providing discretionary access controls.

Role Based Access Controls (RBAC)

The main motivation for a Role Based Access Control (RBAC) system is the capability to define security policies tailored to any operation within an organization structure rather than to low level data objects as is done in traditional DAC and MAC systems [Zhang, 03]. The conventional RBAC model consists of four basic components [Ferraiolo, 01]:

1. A set of users (***Users***)*:* A user is a person or automated agent to which the system manages access to resources.
2. A set of roles (***Roles***): A role is job function or title that defines an authority level.
3. A set of permissions (***Permissions***): A permission refers to an access mode that can be exercised on a resource..
4. *A set of sessions* (***Sessions***): A session is a mapping that relates a user to one or multiple roles. In each session, a user can request activation of some of the roles he/she is authorized to assume. Depending on the RBAC implementation, the system may restrict the number of roles that a user can

activate in a given session.

5. *User Assignment:* It refers to the roles a user belongs to.
6. *Permission Assignment:* It refers to the permissions associated with a role.

In an RBAC system, a security request is decided according to the user and permission assignments [Zhang, 03]. Besides the six core components of an RBAC implementation, Figure 2-1 depicts the *Role Hierarchy function* (denoted by ≥) allows permission inheritance among roles. That is, for two roles r_1 and r_2, if $r_1 \geq r_2$, then all permissions in r_2 are also permissions of $r_1$and all users in r_1 are also users of r_2 [Ferraiolo, 01]. Finally, the constraint function allows separation of duty (static and dynamic), which place limits in the role user assignment to and role activation.

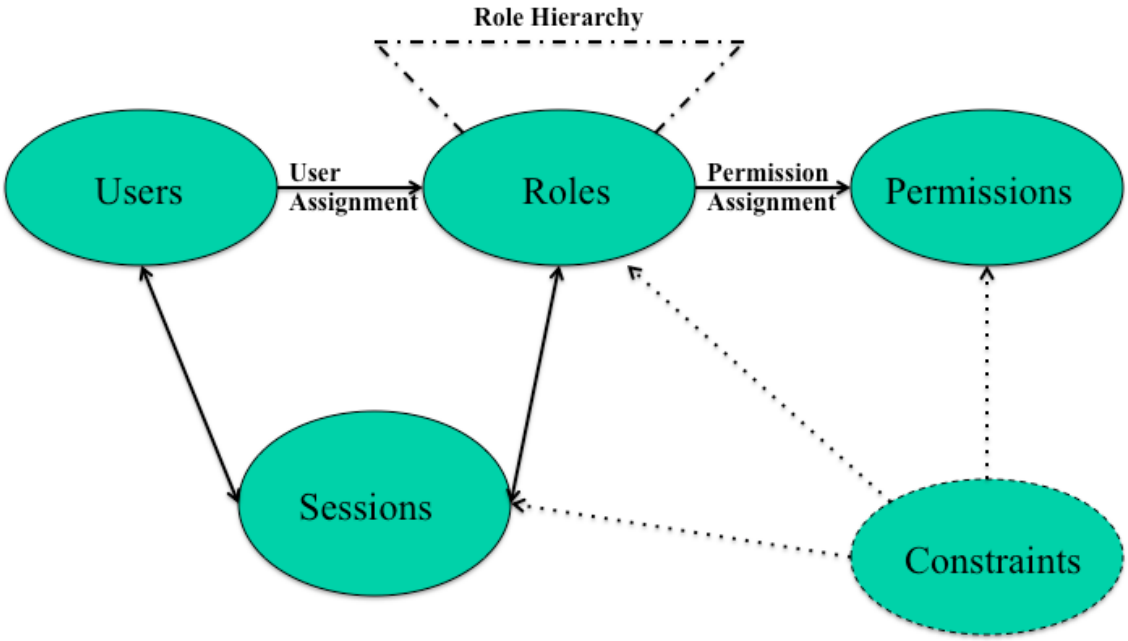


Figure 2-1 Role Based Access Control Components with Hierarchical Roles

Trust Management Systems

According to [Carbone, 03], a global computing environment is composed of entities that not only are autonomous, decentralized, mobile and dynamically configurable but also capable of operating under partial information. Moreover, traditional security frameworks are not suitable for this type of environment because they become rigid, weak and difficult to administer due to the following [Dimmock, 05]:

1. The great and diverse number of entities that can be present at any given moment,
2. An entity's possible new modes of operations. For instance an entity can be either disconnected from, partially or fully connected to a central network.
3. The nature of interactions where in many cases entities unknown to each other collaborate in order to complete a given task.

As such, *Trust Management Systems* have been proposed to address these challenges. These systems are designed to include the human notion of *trust* in order to promote interactions amongst entities while evaluating risks in situations where partial information is available [Cahill, 03]. In addition, a Trust Management System (TMS) may implement the concept of recommendation (i.e. a user's opinion about other) to construct and refine the trust in an entity. It must also be noted that trust in a TMS is not always symmetric. That is if *a* user Alice trusts *Bob*, it does not necessarily imply that *Bob* trusts *Alice* [Dimmock, 05]. Moreover, trust is *context* dependent, specifically if *Alice* trusts *Bob* as a doctor; it does not mean that *Alice* trusts *Bob* as a car dealer. Finally, A TMS build up the necessary structures to track and adjust trust data based on

time, as the trust of an entity may vary (increase or decrease) through multiple interactions.

Usually, an entity in a TMS has two components that are used to compute trust and render a security decision. The *"trust engine" component* updates trust information based on evidence (observations and recommendations), whereas the *"risk engine"* handles the security requests and provides feedback to the trust engine [Carbone, 03].

2.2 Context & Trust Security Systems

In this section we review the state-of-the art context and trust based security systems. They are classified into three major categories:

1. Context Based Security Systems: this category covers systems that use context information in order to render a security decision.
2. Expanded Role Based Access Control (RBAC) Based Security Systems. This category covers those systems that expand and enhance the RBAC model with context and trust predicates.
3. Trust and Risk Security Systems: this category covers those systems that implement trust and risk measures.

At the end of this section, performance parameters for the reviewed context and trust security techniques are introduced. Finally, we present guidelines to select an appropriate security system according to the types of application and environment.

2.2.1 Context Based Systems

This section reviews systems that concentrate on improving data security by relying on context data to render a security decision.

Managing Context Information in Mobile Devices

According to [Korpipää, 03], in order to manipulate and become fully aware of context information, mobile systems must generate reliable records in the presence of uncertain and changing data from multiple sources. The framework described in this work defines methods for acquiring and processing context information in order to provide applications with reliable data. In order to achieve such a goal, the framework defines four functional entities:

1. *Context Manager*: a context manager not only stores context information, but also delivers notifications to clients about changes in the stored information.
2. *Resource Server*: a resource server retrieves information from multiple context data sources and posts it to the context manager.
3. *Context Recognition Service*: the main function of a context recognition service is to create higher-level contextual information from multiple context data observations in order for it to be shared by multiple applications.
4. *Applications*: in this framework, applications are the users of context data which is obtained from the context manager by either a notifications or direct queries.

Furthermore, the framework describes a context by using the following properties.

1. *Context Type*: the [hierarchical] category for a context;
2. *Context Value*: the semantic value for the context type;
3. *Confidence*: the degree of uncertainty of context value.

4. *Source*: the description of the context provider;
5. *Timestamp*: the time at which the context occurred;
6. *Attributes*: additional details for the context that is not described in the previous properties.

Finally, higher-level contexts are constructed by linking a context type to a recognition service, which performs classification from an input context set type. The latter is an entity to be notified of changes of the variables to be monitored. As part of the implementation, this framework uses Bayesian reasoning to classify and generate higher-level contexts.

This model provides a programming interface to handle imprecise information from multiple sources while checking the trustworthiness of the input context data. However, no implementation for data authorization is given; this responsibility is to be fulfilled by the application level.

Context-Based Security Policies with Contextual Graphs

According to [Mostefaoui, 04], a security context is defined as "a set of information collected from the user's application environments that is relevant to the security infrastructure of both the user and the application." Many types of context information can be defined including user's identity, interaction history with a service, location, preferences, type of request service, request time and date, sensitivity of data exchanged, cryptographic protocols, state of network and unit resources (CPU, bandwidth etc.). In order to formalize a security policy, contextual graphs are proposed. A contextual graph (CxG for short) is based on the concept of decision trees in which there are no decision nodes but only "chance" nodes where a contextual element is

analyzed to select the corresponding path. A CxG is an acyclic graph with unique input and output that allows a representation of a given problem (or operational processes) by taking into account the working environment.. A node in the graph can be a security action, a contextual node or a recombination node. A security action is an executable method that aims at enforcing the policy at a given point of the CxG. Contextual and recombination nodes represent a security node. The former corresponds to the explicit instantiation of the contextual element, i.e., role of a requesting user. A contextual node represented by *Cm* where m is the number of exclusive (choice) branches corresponding to known practices. A recombination node *Rn* corresponds to the end of the instantiation of a contextual element once the action on the branch is accomplished, for example R3 in Figure 2-2.

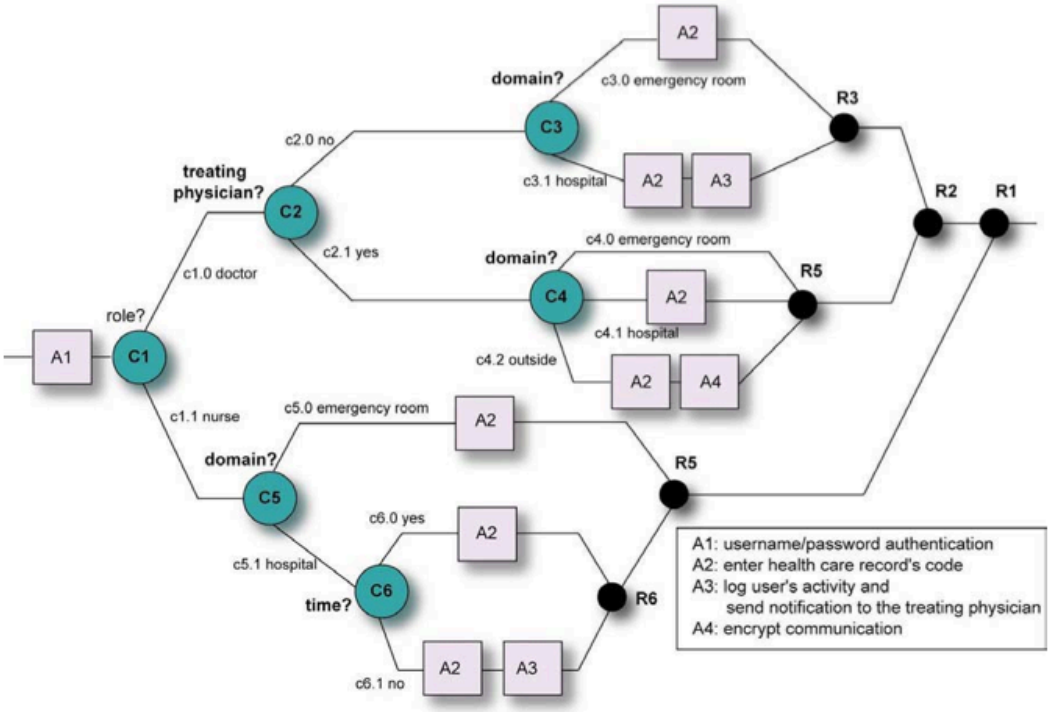


Figure 2-2 An example of a Contextual Graph-Based Security Policy [Brezillon, 04]

The model represents a good theoretical approach to modeling context-based

policies since it not only provides understandable representations of security mechanisms, but also allows a security administrator to add and remove security paths. However, the model does not define ways to include trust, especially since it assumes there are no invalid context data claims when evaluating a policy. Although not defined in the model, either by adding or removing security paths and determining the outcome as security breaches occur, we could build stress tests. Finally, there is no implementation of this model and no provision for the use of historical or past experience context data in applications of security.

Context Owners, Brokers, Sources and Service Providers

A model provides access control for services using context information while keeping the anonymity of the accessing users is presented in [Hulsebosch, 05]. The system architecture defines four entities: Context owner (CO), context provider (CP), Context broker (CB) and Context Aware Service Provider (CASP). The context owner is the mobile user who owns the context data. The context provider makes sure that the context data of a CO is distributed according to the CO's policy and also provides context management (indexing, querying, etc.). The context broker provides services publishing mechanisms to CPs and service discovery mechanisms to service providers. The context aware service provider provides services to a user. The authentication for this Context Aware Access Control (CSAC) is based on verifying whether or not the situational context claimed by the subject is valid. For privacy purposes, the identity of the CO is decoupled from its context information. As such two types of security tickets (tokens) for authentication are required: the Context Ticket and the Context Granting Ticket. The CO and CB know both the tickets. The Context Ticket is issued by the CB

to the CO and contains a pseudonym that the CB uses to link the CO to its CP. The Context Granting Ticket is issued by the CB and instantiates the association between the CO and its CP. Only the mediator CB has access to the Context Granting Ticket. The CB uses this ticket to create new Context Tickets when the CO decides to use other context-controlled services. The CASP only knows the Context Ticket and uses it for communication with the CB.

The model described above does not require complex static mechanisms like PKI (Public Key Infrastructure) or user/password information, while allowing setting complex control policies based on reference context data. However, several drawbacks can be identified. The system requires an infrastructure for the collection, management, interpretation, and controlled release of context information and no provision for it is given. Moreover, since security depends solely on context information, the level of security may not be as high as traditional solutions. Furthermore, no time rate is suggested to keep and refresh context data. The system relies on context providers to supply context data; however, the system does not present a mechanism to handle or detect invalid context claims by such providers, especially when the context owner plays the provider role. Finally, the system is in an early stage and has neither a method for what-if analysis nor a mechanism to include context data history in the evaluation of security.

Context-Enhanced Mobile Services for the Web

In [Debaty, 05] a framework to integrate the physical environment with the World Wide Web is presented. The approach establishes a network of *Web presences* for physical components. Any person, device or location can be described as a web

presence (or software object). This object contains the most up-to-date information about its assigned physical item and presents such data for use by other entities. In this approach, context is defined as "any information that can be used to characterize the situation of an entity, with an entity being a person, place or object that is relevant to the interaction between a user and an application" [Debaty, 05]. Moreover, the framework classifies the context information according to: *Where*: The location of an entity, *When*: Event time, *Who*: Identity of the user and of those around him/her, *What*: The objects near the user and the kind of actions the user can taken on demand *How*: How are those entities organized?

In addition, the physical world is considered to be a dynamic set of interrelated physical entities. Each entity can be described by its type, identity, and physical attributes. Moreover, each entity has relationships with other entities. The relationships are characterized by an extensible set of properties, such as "Contains", "isContainedIn", "isNextTo" and "isCarriedBy". Finally, relationships have inner meta-data such as creation time, expiry time and relative coordinates with respect to a subject.

A Web presence being a virtual representation of a physical entity is formed by a set of modules. Each module delivers a user interface and an API for local/remote programmatic access. XML is used as a standard to expose and exchange information about a Web presence. The system uses a directory model in order to not only store but also manage the Web presence relationships. In addition, an observer and autobiographer module is used to capture and save environment changes. This information allows users and objects to learn and reason about the physical environment. Finally an indexer module used to gather the meta-data about other

entities is included to support Web presence functions.

The system described above enables and facilitates the development and deployment of context-enhanced applications that contain World Wide Web resources in order to enhance a user's experience. However, the system does not implement any module to enforce security over objects (Web presence) and services.

2.2.2 Expanded RBAC Based Security Systems

In this section, systems that improve data security by expanded the RBAC model with context predicates are described.

Generalized Role Based Access Controls (GRBAC)

According to [Moyer, 00], the traditional RBAC model can be extended by applying the concept of roles to both objects (e.g. files, tables, resources, etc.) and environmental variables. Thus, in a GRBAC system, three types of role types can be described:

1. *Subject roles*: in GRBAC, a subject role is semantically equivalent to the traditional RBAC user role.
2. *Environment roles*: environment roles expand security policies by adding constraints or control on context variables.
3. *Object roles*: an object role allows the classification of objects according to their properties, thus allowing the specification of access control on the object role instead of each low-level object item.

In a traditional RBAC system, if a user wants to access a particular object, the user must activate a role that is authorized to access the requested object. However, in

GRBAC, the authorization algorithm is more complex. Not only the user must activate the proper role to access the object, but also there must be an environment role activated in the system that allows the operation the user wants to do with the object. Formally in a GRBAC implementation, for a subject *s* to access an object *o*, *s* must have some subject role *Rs*, such that:

1. There exists some object role *Ro* that contains *o*;
2. There exists some system environment role *Re*, that is currently active; and
3. Re allows the requested operation *t* on the object role *Ro*.

By allowing the definition of environment and object roles, the GRBAC framework can be a powerful model to be used in the specification of security policies in a computationally rich environment. However, since there can be a great number of environment roles, the system may become difficult to administer. Moreover, there must be always an environment role in order to arrive at a security decision. This implies that the system must collect enough context data to determine whether a given environment role is active. In addition, GRBAC does not provide ways to use context data history or to counter invalid context claims. The system depends on trusted mechanisms capable of generating events based on various system state changes. Finally, the model does not define any means to perform what-if analysis.

Role Based Access Control in Ambient Spaces

A description of a distributed, location-dependent and multi-layered RBAC grammar for Ambient and Remote Spaces (a.k.a. ubiquitous environments) is presented in [Wedde, 04]. Each access right is described as a pair *(<role>, <unit>)* where subjects (users) are assigned to a role in a unit or organizational entity (e.g. department,

group, etc.). Each unit is under the control of a special type of unit denoted as authorization sphere. A local authorization team sets the rules of roles for the authorization sphere using five access predicates defined as:

1. *Basic rules (cando):* used to enable or deny access rights;
2. *Derivation rules (dercando):* used to derive relationships between access rights;
3. *Decision rules (decide)*: used to confirm with the user if an access right is granted;
4. *Deduction rules (do):* used to deduce an access right based on the validity of other predicates;
5. *Grant rules (grant):* used to resolve conflicts arising from *dercando* and *do* rules.

Due to the assumed spatial context assumed, the model makes use of hierarchical trusts levels (top secret, secret, etc.), locations (Europe, France, Paris, etc.), and security levels (public key, RSA, RSA1024, etc.). As such, the five access control predicates mentioned above can be augmented with the following assertions:

1. Predicates that model the trust level of a host (login, storage, location, etc.);
2. Predicates that model the security of the communication channel (encryption, key exchanged, authentication, etc.); and
3. Predicates that model the application security (execution, application trusted, application level).

For instance, the rule

cando(OU, s, o, +r) ← *login (s, h) & location (h, OU) & storage (o, h_o) &*

channel_level (OU, h, h_o, secured)

states that in the *OU* authorization sphere a subject *s*, logged in a host *h* (located at OU) can access (read) an object *o* (stored at host h_o), so long as the communication channel between *h* and h_o is secured.

The above framework specifies a grammar to describe access based on context information and location. The grammar can be easily extended to include host power resource information as well as connection mode (weak, strong, etc.). With some additional complexity, the model could describe restrictions of movement for a given object. However, the grammar does not include predicates to take into account the variability of context data or any related data history. Finally, this work does not define predicates to define what-if scenarios.

Dynamic Role Based Access Controls (DRBAC)

According to [Zhang, 03] and [Zhang, 04], the proliferation of small and smart devices has enabled the transformations of regular physical spaces into intelligent spaces. The model extends the RBAC model while dynamically adjusting role and permission assignments based on context data. As such, the new dynamic RBAC (DRBAC) has the following components:

1. *USERS*: entities whose access is being controlled;
2. *ROLES*: job functions that define the authority and responsibility for a user;
3. *PERMS*: an approval or denial to access one or more resources;

4. *ENVS*: a set of context information in the system;
5. *SESSIONS*: a session is a set of interactions between entities;
6. *UA*: a mapping that assigns a role to a user and along with context data, it aids in determining the role to be activated, and
7. *PA*: the mapping between permissions and a role.

A central authority (*CA*) maintains the overall role hierarchy and, when a user enters the system, a base set of roles is assigned to him/her based on his/her capabilities. The *CA* assigns a context agent to the user, which monitors his/her environment and dynamically changes the active role. In addition, each resource maintains a set of permission hierarchies for each potential role that wants to access it. State machines maintain the role subset for each user and the permission subset for each role. Each state machine consists of state variables and events. The context agent (loaded at the requesting unit) collects context information and generates predefined events to trigger transitions in the state machine.

As in other research efforts, this model extends the current static role based access control with dynamic data in order to provide context aware security. However, the model does not provide a way to invalidate erroneous context claims; so it is possible that even though the active roles of a user change, he/she still can access resources with permissions already acquired. In addition, the model requires context agents to be loaded in the units, which may limit the type of units that can participate in the system as it will require those hosts to have enough resources to load and run the agents. Finally, there is no provision for the inclusion of past activity history in order to derive a more accurate security decision.

2.2.3 Trust Based Systems

In this section, we review the current work in trust based security systems. These trust models are adjusted using the user's context or preference data in order to arrive at a security decision.

Building Trust in Decentralized Peer-to-Peer Electronic Communities

The model in [Xiong, 02] describes a trust management system that relies on reputations in order to quantify and compare the degree of trust trustworthiness of peers in a decentralized peer-to-peer environment. In this system a satisfactory interaction is defined to have a value of 1 and a complaint to have a value of 0. The trust metric is defined as follows:

$$T(u,t) = \frac{\sum_{v \in P, v \neq u} S(u,v,t) \bullet Cr(v,t)}{\sum_{v \in P, v \neq u} I(u,v,t)}$$

where

- P is a set of peers in the P2P system;
- u and v are peers in the system, that is $u, v \in P$;
- $S(u,v,t)$ is the degree of satisfaction that u has with v until the t^{th} transaction;
- $T(u,t)$ is u's trust value evaluated by other peers until the t^{th} transaction;
- $Cr(v,t)$ is the balance factor for filtering feedback from v;
- $I(u,v,t)$ is the number of interactions that u has with v up to the t^{th} transaction.

$T(u,t)$ is the ratio of cumulative weighted satisfaction that u receives with respect to the total number of interactions that u has within the P2P system. Moreover, the term

$S(u,v,t) \bullet Cr(v,t)$ is approximated by $I(u,v,t) - C(v,t) \bullet T(v,t)$, where $C(v,t)$ denotes the level of complains that v files against u. Therefore, the trust metric $T(u,t)$ becomes

$$T(u,t) = 1 - \frac{\sum_{v \in P, v \neq u} C(u,v,t) \bullet T(v,t)}{\sum_{v \in P, v \neq u} I(u,v,t)}$$

It is readily seen that the higher the value of $T(u,t)$ is, the more trustworthy u is. Moreover it uses other users' trusts $T(v,t)$ as a balancing factor, that is, the higher $T(v,t)$, the more reliable the level of complaints. In addition, PeerTrust defines the trustworthiness decision criterion as follows:

If $I(u,t) > C_1$ and $T(u,t) > C_2$, then u is trustworthy. C_1 and C_2 are system defined thresholds, where C_1 defines the minimum number of interactions required and C_2 defines the minimum trust value that an entity must have in order for it to be considered trustworthy.

Due to the simplicity of the trust metric $T(u,t)$, this model can be easily applied to mobile systems while providing a good mechanism for measuring the trustworthiness of entities using either direct experiences or recommendations. However, the system requires a minimum number of interactions to derive an adequate trust value. This may present a disadvantage for new users and entities that reenter the system after a long time. Moreover, the system equally weights the feedback from other users when evaluating trust instead of giving the most recent feedback a higher weight. Finally, the system does not use any context data to calculate or refine the trust measure.

A Reputation-Based Trust Model for Peer-to-Peer EC-Communities

The model in [Xiong, 03] expands the system described in [Xiong, 02] in order to increase the trustworthiness of an entity by including the following factors in the trust calculation:

- *Credibility of Feedback*: the model uses credibility weighted peer feedback when calculating the trust metric in order to mitigate the impact of malicious (false) statements about a peer.
- *Transaction Context Factor*: when aggregating feedbacks for a peer's transactions the size of such transactions is an important context that needs to be incorporated in the trust metric to weight the feedback for each transaction. In this case, large (big) transactions weight more in the trust metric calculation than small transactions do.
- *Community Context Factor*: in a particular peer-to-peer community, the use of recent transaction history can be used to evaluate the consistent behavior of a peer.

As such the model in [Xiong, 02] is expanded as follows:

$$T(u) = \alpha * \frac{\sum_{i=1}^{I(u)} S(u,i) * Cr(p(u,i)) * TF(u,i)}{I(u)} + \beta * CF(u)$$

where

- *I(u)* denote the total number of transactions performed by peer *u* during a given period.

- *p(u,i) denotes the other participating peer in peer u's i^{th} transaction.*
- *Cr(p(u,i))* denotes the credibility of the feedback submitted by *p(u,i)*.
- *TF(u,i)* denotes the transaction context factor for peer *u*'s i^{th} transaction.
- *CF(u)* is the community context factor for peer during the given period.
- α and β are weight factors for the transaction and community context part of the equation.

According to the requirements of a peer community, the above parameters can be modified to model a particular requirement. For instance, by making α =1, β = 0 and *TF(u,i) = D(u,i*), the cost in dollars of a transaction, the size of a transaction becomes part of the trust metric calculation. In addition, a peer community could encourage more feedback by its peers by providing a small increase in the trust of a peer whenever he/she provides feedback to others. This may be achieved by making *CF(u)* the ratio of the total amount of feedback (or *F(u)*) over the total number of transactions (*I(u)*).

This model improves the system on [Xion, 02] by checking for the credibility of the feedback (recommendations) given by different peers. However, this system suffers from the same weaknesses of the previous model.

Engineering Trust Base Collaborations

The work done in [English, 04] does not present a true trust model. Rather, it describes the characteristics of what a trust/risk model should have in order to allow trust base collaborations. It starts by denoting that in the global computing environment, entities are required to make their own security decisions often enough with incomplete knowledge of the environment. These entities then develop concepts of trust and risk when interacting with other entities. Users use this trust as a way for managing risk and

learning from past interactions in order to reduce incorrect data or service exposure. As such, it is necessary for a model of trust to allow comparisons in order to express "less risk" or "more risk" in a security domain.

In a trust model, a *decision maker* decides the outcome of a request from an entity (*requester*) by reasoning about the trustworthiness of the requester (*trust evaluation*) and the risk of the interaction (*risk evaluation*). There are two alternatives to view the relationship between trust and risk. The first choice is to consider risk "driving" trust, that is, in a situation *s* and an action *a* which entail a level of risk *r*, how trustworthy an entity should be to be allowed to enter *s* and execute *a*. On the other hand, trust can "drive" risk. In this case, for a situation *s* and an action *a* along with an entity *p*, how much risk the system is willing to accept by allowing *p* to enter situation *s* and carry out *a*. Wherever the cost benefits are quantifiable, the latter alternative gives us a better measure of the risks involved. The *decision maker* is required to have the ability to associate an entity with a risk profile, which can be described, by the combination of the risks of individual outcomes of an interaction. As information becomes more available, the risk profiles for the entity should be reevaluated so that given a new risk value, a new profile may be selected for the entity.

In order to compare two risk values, a "≤" operator needs to be defined. If two trust values, t_1 and t_2, with risk profiles, r_1 and r_2, respectively, such that $t_1 \leq t_2$ then r_1 represents more risk than r_2. Since both profiles may have been evaluated with a different or incomplete set of information, a notion of uncertainty needs to be included in the model to better characterize the collaborating entities. Trust values can be mapped to a probability distribution of costs to represent the likely cost of each value.

Each trust value can be considered equally likely. Thus, trust calculating the maximum costs/benefits for each outcome trust evaluation can be performed. Finally, to introduce uncertainty, a trust value range can be mapped to a cost interval. The size of the interval represents the uncertainty value.

The model describes the key aspects of what risk model should have in order to support trust base cooperation in an environment with incomplete knowledge. However, the framework does not make any reference to what-if analysis when either predicting future scenarios or refining risk profiles.

Trust of Ubiquitous, Transparent Collaboration

A model based on user recommendations to control both the flow of information as well as defining its access is provided in [Shand, 04]. In addition, the model allows the definition of complex policies by ordering user recommendations according to the data content. Users and data may be associated with categories, each of which has a trust value, and each item (users or data) may have one or more associated recommendations. The model then combines the recommendations in order to determine the importance of an item with respect to the category. Categories are analogous to roles in the RBAC model and they are used to restrict the dissemination of information amongst users (principals). A category has a list of privileges associated with it; users use these privileges to distribute information. The trust assessment of an entity is a mapping from an action category pair (category id, list of privileges/actions) to a primitive trust value. In addition, a category can extend another one (hierarchical relationship) and extensions can be grouped into category-bands. Such bands facilitate the exchange of information and allow user's privileges to be intuitively assigned.

In this framework, recommendations associate one permission with another and participants compute their trust in the information by combining their own trust assumptions with others' recommendations [Shand, 04]. The set of permissions *P* linked by recommendations can be classified as follows:

- Actor permissions ***A*** are used to identify people;
- Category permissions ***C*** are used to represent a category memberships;
- Data Entry permissions ***D*** are used to refer to application data;
- Action permissions $\boldsymbol{P_A}$ are used to represent operations over data;
- **{Link}**, initially empty, is the set of all actor category associations;
- Link permissions $\boldsymbol{P_L = \{Link\} \times \{A \cup C\}}$ are used to recommend an actors associated with a category.

Recommendations are combined transitively to determine effective trust value discounted according to the permissions the recommender hold. In this framework, actors, categories and data entries are treated as permissions in order to have a homogeneous structure for privilege assignment and information flow restrictions. For instance, $\{Rec(Michael,trade,t_1)\}_{Max}$ denotes that "Max recommends that Michael be a member of category 'trade' with trust t1". If a user trusts Max to be a member of "trade," then Michael will be considered too. Moreover, if the same user recommends that "trade" be allowed to read data from category "commerce" then the trust will be transferred and Michael will be allowed to read data from commerce as well. A trust value consists of a *(belief, disbelief)* pair, with $(belief + disbelief \leq 1)$. Trust values can be ordered according to trustworthiness. That is, if *b1* and *b2* denote belief values and d1 and d2 disbelief values, a primary order can be established by defining $(b1, d1) \leq (b2,$

d2) iff (b1 ≤ b2) and (d2 ≤ d1). A second ordering is also given by *(b1, d1) ⋒ (b2, d2) iff (b1 ≤ b2) and (d1 ≤ d2)*.

Two recommendations are combined through the ⊗ operator. This operator is used to merge two trust estimates (b_1, d_1) and (b_2, d_2) by dividing the latter by a factor *k,* which represents the maximum value between the normalized b_2, d_2 values and 1. Formally,

$$(b_1,d_1)\otimes(b_2,d_2)=\left\langle \begin{matrix}(0,0)\ if\ (b_1\le d_1)\quad otherwise\\ \left(\frac{b_2}{k},\frac{d_2}{k}\right)\ and\ k=\max\left(\frac{b_2}{(b_1-d_1)},\frac{d_2}{(b_1-d_1)},1\right)\end{matrix}\right\rangle$$

Transitive recommendations are united through the use of a policy function. A policy function $Pol_x(T,y,z)$ is the degree to which user *x* believes *y* should hold permission *z* if everyone else's trust assignments are given in *T*. Let $d_x(y, z)$ summarize *x's* recommendations, with $d_x(y, z) = t$ if there is a recommendation $\{Rec(y,z,t)\}_x$, and *0* otherwise (newer recommendations are supposed to supersede older ones), that is, two recommendations are transitively merged as follows:

- For those recommendations where *x* associates *y* with *p*, and *p* with *z*; that is, $\bigcup_{p\in P} T(x,y,p)\otimes T(x,p,z)$ and
- For those recommendations where *x* gives *p* permission *z* and *p* recommends *y* for *z*; that is, $\bigcup_{p\in P} T(x,p,z)\otimes T(p,y,z)$

Therefore, the policy $Pol_x(T,y,z)$ *is defined as*

$$Pol_x(T,y,z)=\oplus\left\langle d_x(y,z)\cup\bigcup_{p\in P}T(x,y,p)\otimes T(x,p,z)\cup\bigcup_{p\in P}T(x,p,z)\otimes T(p,y,z)\right\rangle$$

The ⊕ operator is defined as a way to combine a series of recommendations by averaging their belief and disbelief components [Shand, 04]. That is, the risk of an operation is the sum of all risks of all the possible outcomes of that operation, where the risk of an outcome is a function of the likelihood and impact of that outcome.

This model organizes principals (users) based on categories much like role-based access control does. Belief-disbelief pairs specify uncertainty of trust values, and thus, policies can be constructed with the use of the ⊗ operator. However, since the new recommendations supersede the older ones, no history is used when constructing trust values. Policies are stored locally in the mobile devices; so they must be refreshed periodically in order to minimize a potential security breach since permissions may be granted because of stale data. Moreover, the model does not define trust in terms of context data, but on interactions, and it does not offer mechanisms to validate invalid or outdated recommendations.

TrustBAC

The implementation of the traditional RBAC model in an open decentralized system (e.g. a pervasive environment) is not an acceptable solution. The implementation is not satisfactory because the environment is dynamic and because, in most cases, the user population is unknown. In [Chakraborty, 06], the researchers propose TrustBAC as an extension of the traditional RBAC where a multi-level trust scheme is defined to allow the creation of trust levels that are mapped to roles (unlike the conventional RBAC model where users are mapped to roles). In TrustBAC the trust level of a user can be computed not only by using their credentials, but also from the results of past interactions and recommendations. As the trust level of a user changes, the roles they

can have also change.

Formally, TrustBAC defines a trust relation between two entities, *A* and *B,* as follows:

$$\left(A \xrightarrow{c} B\right)_t = \left[{}_A E_B^c, {}_A K_B^c, {}_\phi R_B^c\right]$$

where ${}_A E_B^c$ represents the experience of *A* with regard to *B* in context (session) *c*; ${}_A K_B^c$ denotes *A*'s knowledge about *B* and ${}_\varphi R_B^c$ represents the cumulative effect of all recommendations that *A* receives about *B*. All these three factors are expressed as numerical values in the range [-1, 1] and {Ɏ}. A negative value denotes a *trust-negative* type for the entity, whereas a positive value indicates a *trust-positive* type. A zero value indicates a *trust-neutral* type and *Ɏ* indicates that there is insufficient information to calculate a trust value. Table 2-1 shows the formulae used to calculate the user experience, understanding and recommendations for a trust relation. To calculate the experience with regard to another entity, user *A* keeps a record of all events occurring during a time interval. Each event *k* has an associated a value v_k between -10 and 10 to express negative, neutral (0 value) and positive experiences. For each interval *j* there are *n* events. Finally, for each of the *n*, there is a non-negative weight w_i which is used to make the final calculation of ${}_A E_B^c$.

A's understanding about *B*, ${}_A K_B^c$ is defined by two parts: a direct knowledge (*d*) and an indirect knowledge (*r*), where *d* and *r* take values between [-1, 1] or *Ɏ* expressing *A's* direct and recommended knowledge about *B*. w_d and w_r are the corresponding no-negative weight factors.

The recommendation ${}_\varphi R_B^c$ is computed by normalizing the recommendation

scores. A recommendation score is simply the multiplication of the recommender's trust value by the actual recommendation values, where, for *n* recommenders, ${}_AT_{j,t}^N$ and V_j are the trust and the recommendation values for the j[th] recommender.

Since two trusters may come up with different trust values for a given trustee, TrustBAC defines the operator ⊗ to normalize the trust relationship, that is,

Table 2-1 TrustBAC Relation Definition

User Experience	$I_j = \begin{Bmatrix} \psi, if\ no\ event\ is\ found \\ \dfrac{\sum_{k=1}^{n} v_k^j}{\sum_{k=1}^{n} \lvert v_k^j \rvert}, otherwise \end{Bmatrix}$ $${}_AE_B^c = \sum_{i=1}^{m} w_i I_i$$
User Understanding	${}_AK_B^c = \begin{Bmatrix} d, if\ r = \psi \\ r, if\ d = \psi \\ \chi, if\ r \neq \psi, d \neq \psi, \\ \psi\ otherwise \end{Bmatrix}$ $$\chi = w_d \cdot d + w_r \cdot r$$
Recommendations	${}_\phi R_B^c = \dfrac{\sum_{j=1}^{n} \left({}_AT_{j,t}^N \right) \cdot V_j}{\sum_{j=1}^{n} \left({}_AT_{j,t}^N \right)}$ $${}_AT_{j,t}^N = v\left(A \xrightarrow{rec} j \right)_t^N$$

$\left(A \xrightarrow{c} B\right)_t^N = W \otimes \left(A \xrightarrow{c} B\right)_t$ where W is a weight vector of the form [W_E, W_K, W_R] - *experience, knowledge, recommendation* - such that $W_E+W_K+W_R=1$ and W_E, W_K, $W_R \in [0,1]$. Thus, the trust value of an entity B from A perspective and denoted as $v\left(A \xrightarrow{c} B\right)_t^N$ is formulated as follows: $v\left(A \xrightarrow{c} B\right)_t^N = W_E \cdot_A E_B^c + W_K \cdot_A K_B^c + W_R \cdot_\varphi R_B^c$

When the value of $v\left(A \xrightarrow{c} B\right)_t^N$ falls in the range [-1, 0], it implies the entity is distrusted. If the value is 0 there is neither trust nor distrust, whereas when it falls in the range (0, 1], the entity is trusted. A value of *¥* is to indicate that the trust is undefined.

This framework allows the activation (deactivation) of roles by users according to their trust values. Most importantly, each event is weighted to calculate the final experience value during a time interval. The system does not specify how to capture those events and it is up to a particular system implementation to specify not only the event weights, but also the recommendation and knowledge (direct, indirect) values. The system does not use context information to expand the RBAC model to refine the entities trust measures.

The SECURE Project

According to [Cahill, 03] and [Carbone, 03] the Secure Environments for Collaboration among Ubiquitous Roaming Entities (SECURE) seeks to create a framework to reason about trust and risk as well as to deploy verifiable security policies. In SECURE, risk analysis is done by decomposing a trust-mediated action (or interaction) into possible outcomes. Each outcome is given a cost-PDF (probability density function) which represents the range of possible benefits and costs that may be incurred should each outcome comes about. Therefore, when an interaction from an

entity *p* takes place, the SECURE risk evaluator reviews the possible cost-PDFs for the different outcomes while at the same time the SECURE trust calculator provides information (denoted as *t*) that determines the risk's likelihood based on the identity of *p*. Furthermore, the risk evaluator uses *t* to select the appropriate cost-PDFs. Finally, the SECURE request analyzer combines all the cost-PDFs and decides which action should be taken.

Each trust decision is based on evidence gathered from personal observations and recommendations from third parties. Personal observations are recordings of outcomes of interactions, which are evaluated against the expected (correct) behavior to produce experiences, the values of which are given in terms of gain or loss. The trust model in [Carbone, 03] focuses on the set Γ of trust values representing the degree of trusts. Γ has two orderings $\preceq$ and ς, such that $(\Gamma, \preceq)$ is a complete lattice and (Γ, ς) is a complete partial order. The ordering $\preceq$ tells us which one of the two given trust values is of higher level, whereas ς represents that a particular trust value might contain more information than another. The model assumes that not all entities have information about each other. Therefore two elements $\perp_{\preceq}$ and $\perp_{\varsigma}$ are defined to represent complete distrust and unknown trust value (no evidence of trust or distrust), respectively.

[Carbone, 03] further defines a technique to build the triple $(\Gamma,\preceq,\varsigma)$ starting with a complete lattice, $(D, \leq)$, while taking into account the set of intervals of type $[d_0,d_1]$ over *D*. The ordering ς is considered the "width," which represents the degree of uncertainty. The pair $([0,100], \preceq,)$, the intervals of which are subintervals of [0,100], is an example of a lattice over percentage numbers. The interval [40, 60] could represent the trust that an entity *e* has in entity *z*, with uncertainty 60 – 40 = 20. Formally, given a

set of entities *P* and the set *Γ*, the trust information function can be defined as: $m:P\rightarrow P\rightarrow\Gamma$, where the function *m* applied to *e* applied to *z* is the trust value $m(e)(z)\in\Gamma$ expressing *e*'s trust in *z*; that is, entities' trust in each other can be modeled as a function that associates to each pair of entities a trust value $t\in\Gamma$ [Nielsen, 03]. Furthermore, every subject (user, principal) has a local policy that expresses how it computes trust information. The model implements delegation, which means that an entity can refer to another entity's trust information.

When an entity *e* makes a request for interaction, the request is passed to the request analyzer, which gathers the information about *e* from three sources: the entity recognition component, the trust calculator, and the risk evaluator. As such, the entity recognition verifies that entity *e* is recognized; then, the request analyzer requests a trust calculation from the trust calculator. The risk evaluator considers the possible cost-PDF and uses the trust calculation to select the appropriate one. Finally, the request analyzer combines all the information gathered and renders a decision about *e*'s request.

SECURE defines a complete trust cycle from gathering evidence about entities to computing trust in order to render a decision when two entities interact. In SECURE, an explicit cost-benefit analysis is used to determine how much trust is required to offset the risk [Dimmock, 04]. Moreover, the system allows delegation; so one entity can compute the trust on behalf of another entity. In addition, trust calculations and evidence can be stored to be used by other entities (as recommendations) and as part of the entity recognition. However, the model only takes into account one context risk factor (identity) to render a security decision. Moreover, the model does not present a way to do a what-if analysis on this context factor. Finally, the model does not include

mechanisms to counter invalid claims (identity and recommendations).

The Extended OASIS Project

The OASIS model [Bacon, 02] is a role-based access control implementation that allows the definition of conditions for roles activation and the permission such roles can acquire in a security policy. In this model, an authorization rule specifies the roles a principal must hold and environmental constraints that must be satisfied for a privilege to be exercised. A role activation rule denotes the prerequisite conditions and constraints that a principal must satisfy in order to enter a role.

To allow trust and cost/benefit computations to be included within policy rules, OASIS environmental predicates are extended to include SECURE [Cahill, 03] rules and thus, allow a policy author to reason about the trust and cost data per-outcome basis, instead of relying on a simple risk metrics value. To allow trust and cost benefit computations in OASIS, three main predicates are defined as follows:

- *Trust retrieval*: - *trust (principal, context, value?*)* – used to get trust values for a particular principal under a particular context (or action).
- *Cost retrieval*: - *cost (action, outcome, cost?*)* - used to retrieve the cost for a particular outcome.
- *Risk thresholding*: defined BNF predicates to determine if a trust value is adequate for a particular outcome cost.

This model allows the specification of trust values on context risk factors (trust contexts) in order to expand access control definitions. In addition, the system allows the inclusion of discounted inputs from multiple parties in order to build new trust

* Denotes an output parameter

values. The inclusion of SECURE in OASIS represents an attempt to unify trust reasoning into access control. However, the system has not been fully validated for the pervasive computing environment. In addition, the model does not present a mechanism to perform what-if analysis on context factors in order to predict future scenarios. At the time the work proposed in [Dimmock, 04] was published, the use of context data history in the evolution of both trust and risk measures, was in an early phase.

Trust Enhanced Security for Mobile Agents

According to [Lin, 05] a model that takes into account the trust dynamics of past experiences including opinions (intentions or beliefs) is presented. As such, a trust model TM = (E, R, OP) is defined where E represents the entities in the system, R trust relations amongst entities, and OP operations for managements of trust relations. There are several types of trust relations, for instance:

- *Authentication Trust*: belief on the legitimacy of the keys held by an entity in the system;
- *Execution Trust*: risk of the incorrect execution of the agent in a mobile host;
- *Mobile Code Trust*: belief in the maliciousness of the mobile code that has been deployed in a host;
- *Direct Trust*: trust of one entity that holds in another entity with respect to a given context;
- *Recommended Trust*: capacity of an entity to decide whether another entity is reliable in the given trust class and in its honesty for recommending other entities;

- *Derived Trust*: trust relationship derived from other atomic trust relationships.

Furthermore, a trust relationship of the form (*P, Q, C, T, D, t, v, p, n*) can be defined to denote that entity *P* trusts entity *Q* with regard to trust class *C* (the model defines authentication, execution, code as trust classes) with trust type *T* (direct, recommended, derived) in time duration *t. D* denotes the set of domains (a domain is a group of entities that obey the same security policies), *v* defines the trust value (expressed as an opinion), *p* denotes the number of positive experiences and *n* the negative experiences.

An opinion consists of belief (*b*), disbelief (*d*), and uncertainty (*u*) values, where

$$b = \frac{p}{p+n+k}, d = \frac{n}{p+n+k}, u = \frac{k}{p+n+k}, for\ u \neq 0, k \geq 1, \left(b+d+u\right) \leq 1$$

In this model, trust operations include trust evaluation, evidence mapping, trust combination, and trust comparison. An entity collects opinions about other entities either via a recommendation algorithm or through internal trust analysis. The system requires that each entity collect its own positive and negative evidence about others with the use of counters. With the collected evidence, the truster can obtain an opinion value. A trust-security manager defines trust management services in order to provide trust decisions and dynamically update the trust relationships.

This model focuses on building trust (relationships) amongst interacting agents by keeping minimal and historical information based on past experiences (positive and negatives). The relationships are based on only the three trust classes; however they do not include any user context information. Moreover, the model does not offer

mechanisms to check the updates with respect to the three trust classes after an interaction since it assumes that each entity has the capability to do such a check.

Autonomic Trust Prediction

According to [Capra, 06], the realm of pervasive computing calls for new autonomic and lightweight trust in order to minimize not only user intervention, but also the overhead imposed on the mobile devices themselves. As such, a system grounded on the Kalman filter theory [Kalman, 60] is proposed to derive a trust model that uses a set of observations (i.e., direct experiences) to predict the state of the system.

The Kalman filter is a set of recursive equations that provide a way to estimate the current state of a dynamic system, starting from observations that contain random errors. For instance, a Kalman filter of one dimension is represented by a vector $x \in R^n$ (n = 1) and modeled by the equation:

$$x_{t+1} = x_t + V_t, t = 1,2,...$$

In this case, the state of the system at time *t+1* depends on the state of the system at time *t* and a random process noise V_t. In addition, the *current system observation* y_t is expressed in terms of the system's *current state* and a random measurement noise W_t, that is: $y_t = x_t + W_t, t = 1,2,...$ Assuming that not only process and measurement noises (V_t, W_t) are uncorrelated and normally distributed, the Kalman filter provides a prediction algorithm of the following form:

$$x_{t+1} = x_t + \frac{\Omega_t}{\Omega_t + R_t}(y_t - x_t) \ , \ \Omega_{t+1} = \Omega_t + Q_t - \frac{\Omega_t^2}{\Omega_t + R_t}$$

with $\Omega_0 = E[(y_0 - x_0)^2]$ (the variance of the distribution) and Q_t, *and* R_t the time

dependent covariances of V_t, *and* W_t, respectively.

Given the system observation y_t at time t and the prediction x_t computed after the $(t-1)^{th}$ observation, the next best estimate of the state of the system is equal to the previous state plus a term that is proportional to the distance between the last observation and the prediction. Intuitively, the higher the value of R_t, the lower the impact of the observation in computing the next estimate. The higher the value of the process noise Q_t, the higher the importance of the latest observation.

To apply the Kalman filter in a trust model for the pervasive system, the following is assumed: for any given service *s* offered in the system, there are many different providers that a client may contact regardless of time and location. Each provider advertises service *s* promising a certain quality of service, which is represented in terms of attribute/value pairs (a_i, vi). In order for a client to trust a provider, it can use the Kalman filter to predict the discrepancy between the provider's advertised attribute values and the client's measurements. Therefore, it is possible to model a client's trust in terms of these discrepancies so that trust decreases as a result of high discrepancies, and vice versa. Formally, $a_i \; i \in [1, n]$, denotes the attributes that constitute the specification of a certain category of services. The term $p[a_i, B]_t$ denotes the value of the attribute a_i promised by server *B* for interaction *t*, and $m[a_i, A]_t$ expresses the value of the attribute a_i measured by client A for interaction t; the state of the system for service *s* at time *t* can be described by $x_t^s \in \Re^n, x_t^s = [x_{1t}, x_{2t}, ..., x_{nt}]$, with

$$x_{it} = \begin{cases} \dfrac{|\, m[a_i, A]_t - p[a_i, B]_t \,|}{\max\{m[a_i, A]_t, p[a_i, B]_t\}} \; if\,(m[a_i, A]_t < p[a_i, B]_t \wedge a_i \; is\, IU)\, or \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (m[a_i, A]_t > p[a_i, B]_t \wedge a_i \; is\, DU) \\ 0 \qquad otherwise \end{cases}$$

where from the client's perspective, each attribute a_i is assigned to one of these two categories: (a) Increasing Utility (the higher the measured value of the attribute, the higher the utility experienced by the client, i.e. bandwidth); and (b) Decreasing Utility (the higher the measured value of the attribute, the lower the utility, i.e. service time).

The server trustworthiness can then be described by state variable $l_t^s \in \Re^n, l_t^s = [l_{1t}, l_{2t}, ..., l_{nt}]^T, with\; l_{it} = 1 - x_{it}, l_{it} \in [0,1]$. The higher the discrepancy the lower the experienced trust and vice versa. The basic Kalman filter can now be used to predict how much the user's experience of a service will deviate from what the service provider has promised (x_{t+1}^s) and, consequently, how much the server can be considered trustworthy (l_{t+1}^s). It should be noted that at the beginning (time t = 0) a client chooses a service provider based on his/her natural trust mindset.

The Kalman filter is particularly appealing to pervasive systems as it is very lightweight, both in terms of memory requirements (only vector X_t) and computational load (the recursive Kalman equations can be efficiently computed). Moreover, it makes predictions based on an arbitrary long history of interactions (the discrepancies between the predictions and the real values are used to train a Kalman filter to assess the trustworthiness of an entity). That is, the more frequently two entities interact, the faster the filter stabilizes and thus reducing the distance between prediction and actual state of

the system. Finally, the model enables what-if analysis as the white Gaussian noises R_t and Q_t can be used to model the degree of importance to the latest observation. However, the system does not define trust in term of user context data, but as the measured quality of specific attributes of a service provider. Since it uses the latest observation, the filter may take some time to stabilize when a client does not interact enough times with a service provider.

TRAVOS

According to [Patel, 05] and [Teacy, 06], TRAVOS provides an entity (truster) with two methods for assessing the trustworthiness of another entity (trustee) in a given context. First, the truster makes an assessment based on its direct interactions with the trustee. Second, the truster may assess reliability based on the reputation of the trustee.

When two entities *r* (*truster*) and *e* (*trustee*) interact, the outcome of their interaction is considered successful by the truster if the trustee fulfills its obligations. The (binary) outcome *O* is said to be 1 when *e* fulfills its contract and 0 otherwise. The set of outcomes from time *1* to *t* is denoted as O^t where {t: t ∈ Z, a set of integers t>0}. In addition, each entity in TRAVOS keeps a history of the outcome of its interactions. Formally, the interaction history between *r* and *e* can be denoted by the tuple $R^t = (m^t, n^t)$, where m^t denotes the number of successful interactions and n^t the number of unsuccessful interactions. Furthermore, if the tendency of a trustee to fulfill its obligations is governed by its behavior, then such behavior can be specified by the probability *P* that this trustee will fulfill its commitments. Therefore if $B \in [0,1]$ denotes the trustee behavior with respect to the truster, then $B = P(O = 1)$, provided that *r* has complete information about *e.* However, complete information cannot be assumed

at all times, so the best it can be done is to get the *expected value* (*E*) of *B* [Patel, 05]. As such, TRAVOS adopts a probabilistic approach to model the trust that a truster has over a trustee as the expected value of *B*. The trust level *T* at a time *t* is defined as follows: $T = E[B \mid O^t]$. In order to determine the expected value, TRAVOS uses the beta family of *probability density functions* (PDFs) to model the distribution of *B* as follows:

$$f(B \mid \alpha, \beta) = \frac{(B)^{\alpha-1}(1-B)^{\beta-1}}{\int_0^1 U^{\alpha-1}(1-U)^{\beta-1}dU}, where\, \alpha, \beta > 0$$

In order to adapt the equation above to calculate the value of *T* based on the interaction outcomes recorded by truster *r*, the values of α and β must be determined. TRAVOS assumes that prior to observing any outcome, all values for *B* have the same expectation, that is $\alpha = \beta = 1$; therefore at the time *t* of assessment: $\alpha = m^t + 1$ and $\beta = n^t + 1$. Thus $T = E[B \mid \alpha, \beta] = \alpha / (\alpha + \beta)$.

However, *T* does not differentiate between cases when a truster does not have adequate information about a trustee. Therefore, a measure of *confidence Y* is defined to be the probability that the actual value of *B* lies within an acceptable margin of error ε about *T*. In other words, *B* should lie between $(T - \varepsilon)$ and $(T + \varepsilon)$; therefore *Y* is defined as:

$$Y = \frac{\int_{T-\varepsilon}^{T+\varepsilon} (X)^{\alpha-1}(1-X)^{\beta-1}\,dX}{\int_0^1 U^{\alpha-1}(1-U)^{\beta-1}dU}$$

In certain circumstances an entity can ask opinions from others in order to improve the information about another entity. Formally, the true opinion of an entity *p*

about *e* at time *t* is denoted as the tuple $R_p^t = (m_p^t, n_p^t)$. The reported opinion *p* about *e* is denoted as $R'_p^t = (m'_p^t, n'_p^t)$. The distinction is necessary because *p* may not report accurately for self-interest reasons. Assuming that the opinions reported by any entity *p* about other entities are independent, a truster *r* can form an opinion about *e*, given *k* recommenders as follows:

$$N = \sum_{i=0}^{k} n'_i, M = \sum_{i=0}^{k} m'_i \ where \alpha = M + 1 \ and \beta = N + 1$$

However, the equations above hold under two conditions:

1. *Common Behavior*: the behavior of the trustee must be independent of the identity of the truster and
2. *Truth Telling*: the reputation provider must report its observations accurately and truthfully.

Unfortunately, these conditions do not always hold since the entity providing the opinion could act maliciously towards several other entities. Therefore, in order to assess the reliability of an opinion provider, TRAVOS calculates the probability that an entity will provide an accurate opinion based on its past opinions and later observed interactions (made by the truster after it has acquired an opinion) with the trustees for which opinions were given.

In order to estimate the probability of accuracy, given the opinion $R'_p^t = (m'_p^t, n'_p^t)$. and denoting T_p' as the trust calculated using R'_p^t, TRAVOS calculates the probability that such trust reflects the trustee behavior towards the truster, that is, $T_p' = B$. This probability is denoted as $r_{r,p}$- the accuracy of *p* according to *r*. Since all possible values of T_p' are infinite, TRAVOS splits all possible values for R'_p^t into bins; for each bin the number of successes ($C_{success}$) and failures (C_{fail}) are recorded ($\alpha^o = C_{success} + 1$ and

$\beta^o = C_{fail} + 1$). For each bin bin^o, the probability that the true value of B lies within the range of expected values belonging to that bin is calculated. The probability $\rho_{r,p}$ is calculated as follows:

$$\rho_{r,p} = \frac{\int_{\max(bin^o)}^{\max(bin^o)} (X)^{\alpha^o - 1}(1-X)^{\beta^o - 1}\, dX}{\int_0^1 U^{\alpha^o - 1}(1-U)^{\beta^o - 1} dU}$$

By computing this probability an entity tries to guarantee that opinions from inaccurate entities are not given equal weighting to those from accurate entities [Patel, 05].

TRAVOS provides a mechanism for assessing the trustworthiness of entities using either direct experiences or recommendations. Moreover, TRAVOS not only calculates the probability that there will not be unsafe effects from an interaction, but also can be used to handle inaccurate opinions expressed by reputation entities. However, at the moment of publication, TRAVOS does not remove old experiences when calculating trust, which may be an unsafe assumption, since entities may change their behavior over time. The model does not take into account any user context data to compute a more accurate trust value.

Sociotechnical Architecture for Online Privacy

According to [Dawn, 05], users are able to change their privacy preferences according to context. As users have more control over situations and data, the more trusting they become when engaging in on-line activity. However, missing from this model of trust is the strong effect of user control and contributions from multiple stakeholders. This model extends the trust model in [McKnight, 02] with new constructs consisting of stakeholder interventions to foster trust. This architecture defines a series

of cooperating agents and a Web privacy ontology architecture, which incorporates legal, ethical and standard regulations, in order to increase the user's trust in the platform. The Web privacy ontology not only facilitates the integration of information exchange services, but also allows users to save their preferences as ontologies, which aids in the expression of intentions. Furthermore, external agents are defined to allow interactions with services and to allow interventions by foreign entities (trusted third party agents). Agents executed in a user system, such as the personal context agent, maintain contextual information to provide contextualized contract negotiation and transactional interaction with sites. This agent has independent user-preference induction and context-sensitive text-mining modules to analyze feedback from other agents and changes in user context. A user context contains a set of beliefs about the current situation, including privacy preference beliefs and trusting beliefs about an entity. Finally, the arbitrator agent uses service regulations and user preferences to determine whether or not a privacy contract can be established with the accessed service.

Due to the use of ontologies, the research described above illustrates the need for historical context data in order to give a better picture of trust in a security system. In addition, ontologies allow reasoning about the permissions and preferences for a user. However, the system does penalize the user for new contexts (preferences) because the system must learn and develop higher-quality semantics and reasoning with human intervention. In addition, the system is tailored to Web environments and requires powerful hosts in order to run the internal agents and collect the ontological information. The current implementation focuses on a user interface to support privacy

tasks such as those during contract negotiation, user preference population, filling out forms, and surfing. This makes the system difficult to implement in pervasive mobile environments, as user intervention may not be feasible for all cases. The trust model does not include contextual risk factors present in a mobile environment such as location, network status and power resources. Finally, this work mentions neither methods to handle unfounded claims nor ways to create what-if scenarios.

B-trust

According to [Quercia, 06] a trust framework for a pervasive environment should specify the following characteristics:

- Be distributed
- Be lightweight
- Support entity anonymity
- Use minimal bandwidth for data transmission
- Robust to common attacks
- Support trust evolution
- Use both positive and negative recommendations
- Base trust calculation in context, subjectiveness, and time
- Integrate the trust metric with a decision making process
- Be expressive and tractable

B-trust supports anonymity by using false pseudonym for each entity (using distributed blind threshold signature [Juang, 03]). In B-trust, an n-level metric measure is used to allow the definition of trust levels, which are employed as a way to define the trustworthiness of an entity. For example, with four levels (n = 4), a possible

classification is can given as follows: level-1 means ‘very untrustworthy’, level-2 means ‘untrustworthy’, level-3 means ‘trustworthy’, and level-4 means ‘very trustworthy’.

The authors in [Quercia, 06] argue that a model whose trust measures are purely continuous (i.e. from -1 to 1) introduce unnecessary overhead when computing confidence values for a particular trust value. In B-trust a counter is incremented at the corresponding interval where the trust value has happened to fall. Since there is a counter for each level, a confidence level can be computed for each interval. Initially, the system assigns low confidence values to all entities uniformly.

In B-trust, trust calculation is done using two evaluations. The first evaluation is based on direct experiences (past interactions). The second is based on recommendations. These to evaluations are then combined to make a decision and determine to which trust level in the metric falls into, thereby incrementing the corresponding the counter. Direct trust is evaluated by creating a tuple that describes the distribution of the trust the system has for the requesting entity. The trust is based on the context in which the request is made as well as the previous interactions. A confidence level and variance are also calculated over this tuple to determine the reliability of the direct trust calculation. The recommended trust is done in a similar way. A tuple is created based not only on the request context, but also on the probability that a recommender has a trust level of one of the discrete levels outlined in the n-level metric. The tuple for recommenders then also becomes distributed over the different levels and a confidence level can be calculated on their recommendations. These two trusts measures are then combined as part of a weighted average. Again, a confidence level can be calculated for the overall trust value.

Each trust value also evolves over time. As time progresses, a formula is used to push the confidence values towards the initial values (bootstrapping levels). When the confidence values decayed at the point of initial levels, the system deletes clears the counters and start the computation all over again.

Trust Networks on the Semantic Web

The work in [Golbeck, 03] describes the applicability of social network ("Web of Trust") analysis to the semantic Web in order to compute and infer trust relationships in the network. This analysis is based on studies in social networks that state that any two people in the world are separated only by a small number of acquaintances (a "small word" phenomenon or network).

The nature of the semantic Web is of a large graph where resources (or objects) are connected by predicates (properties). Thus, by treating a "Person" as a node and the "knows" or "trust" relationship as an edge, a graph can emerge when two "Persons" know or trust each other. Moreover, using a graph model, transitive relationships can be established. For instance, if person A does not know another person B, but some friends of A know B, then A is "close" to knowing B in some sense (or context). By weighting the edges of the graph with some measure of trust, it is possible to infer how much A should trust B based on A's friends. The system defines nine (9) levels (or ontology properties) to specify trust between two people. These levels are defined as follows: 1. Distrusts absolutely, 2. distrusts highly, 3. distrust moderately, 4. distrusts slightly, 5. trusts neutrally, 6. trusts slightly, 7. trusts moderately, 8. trusts highly, and 9. trusts absolutely. These levels of trust can be applied in general or limited to a specific factor or topic.

Using the trust levels (weights) assigned to the graph edges, several trust calculations can be made over the network as follows:

1. Maximum and minimum capacity paths: in this case trust follows the standard rules of network capacity paths, where the amount of trust a source can give a sink is limited by the smallest edge weight along a given path.
2. Maximum and minimum length paths: since there may be multiple trust paths between to two nodes in a graph, it is useful to know lengths of those paths with the aim of doing a better trust path selection.
3. Weighted average: the system defines the formula to represent the recommended trust for between to nodes i and s as follows:

$$t_{is} = \frac{\sum_{j=0}^{n} \begin{cases} (t_{js} * t_{ij}) \; if \; t_{ij} \geq t_{js} \\ (t_{ij})^2 \; if \; t_{ij} < t_{js} \end{cases}}{\sum_{j=0}^{n} t_{ij}}$$

where i has n neighbors with paths to s. A recursive algorithm implements the above metric in order to calculate trust between any two nodes in the network. However, when there is an edge between *i* and *s*, the system does not compute the trust average, since direct trust is always better measure than transitive trust.

This system illustrates a method for creating a trust network as well as a metric for finding trust between to nodes in the network. At the moment of publication of this research, the trust average metric was considered a proof-of-concept. However, the system assumes that each person sets the trust measures about the persons he/she knows in a subjective manner. Finally, the trust metric suggested does not take into account

past trust measures to compute or further refine new trust values.

Relationship between Trust and Risk

According to [Jøsang, 04], risk and trust are tools for making decisions in uncertain an environment. Risk is calculated using numerical information from the transaction (or collaboration) context. Using classical gambling theory, the *expected monetary value EV* of an investment *I* can be expressed as $EV = IpG$, where *p* is the probability of an expected outcome and *G* is the gain factor on *I*. However, in many situations utility is not the same as monetary value. That is, to express the personal preference of a user, the *expected utility EU* denoted as $EU= pu(IG)$ a is introduced. The function u is a non-linear function of the monetary value, that according to traditional EU theory, the shape of which determines the risk attitude. As such, when $u(IG)<IG$ there is a risk aversion behavior (i.e. the user is no willing to invest or enter in a transaction because his/her monetary gain is less that than the expected monetary value *EV*); when $u(IG)>IG$ there is a risk seeking behavior and when $u(IG)=IG$ there is risk neutrality.

Knowing that in a transaction there are two possible outcomes, a gain factor (for a successful transaction) $G_s \in [0,\infty]$ and a loss factor (for a failed transaction) $G_f \in [-1,0]$ are defined. That is, a gain on an investment can be arbitrarily large and the loss can be at most equal to the investment. Therefore, *the expected gain EG* of an investment can be denoted as $EG=p(G_s) + (1-p)G_f$. In addition to the expected gain, a user also takes into account how much he/she stands to lose when deciding to enter a transaction. Specifically, the higher the probability of success, the higher the fraction of the capital the user is willing to put at risk. If *C* represents the capital, and $F_c \in [0,1]$ is

the fraction of capital C then the amount to invest $I = FcC$. Normally, F_c moves in the same direction as G_s for a fixed p. Similarly F_c varies like p for a fixed G_s. Mathematically it can be described by the function: $F_c(p,G_s)=p^{\lambda/Gs}$ where $\lambda \in [1,\infty]$ is factor that influences the transaction gain G_s on the fraction of the total capital that a user is willing to risk.

Apart from the utility function, λ denotes the contextual component of a user's risk attitude. That is a low λ value indicates a risk taking behavior, whereas a high λ denotes a risk aversion. Since risk attitudes are relative to each user, there can be multiple surfaces generated by the function $F_c(p,G_s)=p^{\lambda/Gs}$, therefore a particular transaction will be represented by a point in a given 3D surface with coordinates (G_s, p, F_c). That is for all combinations of G_s, p and F_c, points underneath the surface D can be trusted, whereas points above the surface are distrusted (*reliability trust*).

Moreover, for a same values of G_s and F_c, and a given cutoff probability p_D over D, the *decision trust* $T \in [-1,1]$ is defined as follows:

$$\left\{ \begin{array}{l} For\, p < p_D : T = \dfrac{p - p_D}{p_D} \\ For\, p = p_D : T = 0 \\ For\, p > p_D : T = \dfrac{p - p_D}{1 - p_D} \end{array} \right\}$$

The decision trust if defined by its three extreme points (0,-1), (p_D,0), and (1,1). In addition, the decision trust is must depend on a distance δ between the current probability and the cutoff probability, that is, $\delta = p - p_D$. A positive decision trust says that the user x originating the transaction can be relied upon. A zero value means that the relaying party is undecided as to whether she trusts x. A negative decision correspond to x cannot be relied upon in the given transaction.

Evaluating Computational Trust Systems

According to [Bryce, 05], modern trust systems tend to be evidence based, in which an entity's decision to trust another is related to facts available using observation of action outcomes and recommendations. Moreover, due to their complexity, there is a need to develop methodologies to fully analyze a trust system, from the computation of an entity's trust value to establishing confidence in the credentials of requesting entities and calculating the risk of permitting or refusing a requested action. Each methodology should answer two questions:

1) *The Question of Security*: How good the framework is at automating the decision making process and how likely is to take the right decision.
2) *The Question of Trust*: How does the computational treatment of trust improve the security question

To answer the first question, a trust security system should have four major components. 1) An entity recognition element in order to ascertain the identity of a requesting entity. 2) A risk assessment component to attribute a cost to deny or refuse a request. 3) A trust attribution module to define a level of risk for an entity (or principal) *P*. 4) Decision element to decide whether a request from *P* should be allowed. Moreover, the strength of the system can be quantified by the triple $<\varepsilon, \rho, \eta>$ representing the total number of events to be controlled; the number of false positives and the number of false negatives respectively. The number of correct decisions is given by $\varepsilon - (\rho + \eta)$. It must also be noted that the notion of right decision varies from principal to principal. As such, there are two profiles that are used to when determining the strength of a decision making process. 1) Principal *P*'s profile whose decisions

define the correct outcome of an action. 2) The community profile made up of messages for P from other entities, representing requests, recommendations and evidence. Since values of <ε, ρ, η> are only comparable for the same profiles, the metric can be redefined as follows: $\varepsilon_P \rightarrow < \rho, \eta >$ where ε is the number of events for profile ε_P.

Finally, the cost of the decision making process can be measure with respect to the *network* (number of messages exchanged for evidence distribution), *memory* (number of items of evidence – observations and recommendations -) and CPU (number of instructions to verify an access right). Therefore, if *c* represents the aggregations of the three costs, the security metrics can be extended to $\varepsilon_P \rightarrow < \rho, \eta, c >$.

To answer the question of trust, trust systems whose trust values encode history assume that the best available indicator of a principal expected behavior is its previous behavior. These systems achieve stability when every principal has an accurate account of the past behavior of all other principals. That is for all ε_P profiles under evaluation the following triple holds <**min**(ρ), **min**(η), **min**(c)>. The authors point out that trust stability is hard to achieve since not only there is a lot of information to store and exchanged for principals, but also some principals may be faulty and unable or unwilling to exchange correct trust values.

2.2.4 Feature Comparison and Performance Model

In this section, we not only compare the previously reviewed context and trust security techniques using on a set of common features, but also introduce a set of metrics to measure the performance and accuracy of the aforementioned security mechanisms.

Feature Comparison

To compare the techniques in section 2.2.1, 2.2.2 and 2.2.3, we use a common framework consisting of the following critical features:

a) *Security Implementation Classification:* what security enforcement implementation does the system used? As reviewed in Section 3, three categories of security implementation considered are as follows:

a.1) Context Based: the system relies solely on context information to enforce security. Several features are highlighted for these systems:

- *Data Collection*: what entity in the system collects context information:
  - *User*: the user requesting access or implementing a security policy collects the relevant context information
  - *Third Party*: a third party unit (e.g. Context Provider) collects the relevant context information
- *Refresh Policy*: does the system implements a policy (e.g. time interval) to refresh the collected context information?
- *Access Control Specification*: does the system define a set of predicates or a grammar to specify access control policies?

a.2) Expanded RBAC Based: the system expands the RBAC framework with context variables and predicates. All the systems reviewed in this category expand the RBAC system by defining environmental roles (conditions) to allow the activation of traditional roles. However, these

systems differ in the order in which traditional roles are activated with respect to environmental roles:

- *TREV*: A traditional role is activated first, which then checks for an activated (by either the system or a trigger) environmental role in order to access the requested data or service.
- *EVRL:* An activated environmental role (by either the system or a trigger) determines the traditional roles that can be activated.

a.3) Trust and Risk Based: in these systems access to a service or data by a user is regulated according to his/her current trust. Several distinguishing properties can be stated as follows:

- *Trust Derivation Data*: the type of data the system uses to calculate trust. Three types of data are considered:
    - *Experience Data:* the system uses interaction results in order to arrive at or refine a trust measure.
    - *Context Data*: the system uses either environment data or user-defined categories to arrive at a trust measure.
    - *Ontology Constructs*: the system uses or defines ontologies to refine a trust measure.
- *Trust Modeling*: does the system use a Probabilistic, Stochastic, or Mathematical (system defined) framework to model trust?
- *Risk Modeling*: does the system calculate risk? There are two ways risk can be calculated:

- *Unspecified:* the system specifies the concept of risk but it does not define a methodology for calculation.
- *Context Data Based:* the system uses context data to calculate or model risk.
- *Risk Tolerance*: the system calculates how much risk it is willing to accept to allow a user to enter into a particular interaction.

▪ *Trust Evolution*: does the system allows trust measures to decay with time to account for lack of interactions or the effect of forgetfulness of the human mind?

b) *Context Data Validation:* is context data validated or is it used without validation?

c) *Use of Data History:* the technique may use historical data in order to arrive at a more accurate security decision. A system can use historical data in one of two modes:

c.1) Full mode: more than one item is used to refine a trust measure or security decision.

c.2) Limited mode: only the most current measure (trust or context data) is used to refine a trust measure or security decision.

d) *What-if:* the system defines a framework to measure the risk/trust of a security policy under different potential situations (e.g. past or future scenarios).

e) *Recommendations*: systems that implement trust recommendations can be classified using the following features:

e.1) Validated: the system validates the recommendations or weights recommendations with respect to the recommendation source.

e.2) Normalized: Two entities can come up with different trust values for a recommendation. Normalized recommendations define an operator to compare two standardized recommendations.

Based on the above features, Tables 2-2, 2-3 and 2-4 compare the reviewed context and trust based security systems.

Table 2-2 Features of Existing Context Based Techniques

Techniques	Common Features					Specific Features		
	Context Data Validation	Use of Data History	What-If	Recommendations		Data Collection	Refresh Policy	Access Control Specification
				Validated	Normalized			
[Korpipää, 03]	Yes	No	No	No	No	No	No	No
[Mostefaoui, 04]	No	No	No	No	No	User	No	Yes
[Hulsebosch, 05]	No	No	No	No	No	Third Party	No	Yes

Table 2-3 Features of Existing Extended RBAC Techniques

Techniques	Common Features					Specific Features
	Context Data Validation	Use of Data History	What-If	Recommendations		Role Activation Method
				Validated	Normalized	
[Moyer, 00]	No	No	No	No	No	TREV
[Wedde, 04]	No	No	No	No	No	EVRL
[Zhang, 04]]	No	No	No	No	No	EVRL

Table 2-4 Features of Existing Trust and Risk Security Techniques

Techniques	Common Features					Specific Features				
	Context Data Validation	Use of Data History	What-If	Recommendations		Derivation Data	Trust Modeling	Risk Modeling	RBAC Trust	Trust Evolution
				Validated	Normalized					
[Xiong, 02]	No	Full Mode	Yes	No	No	Experience Data	System Defined	No	No	No
[Cahill, 03]	No	Limited Mode	Yes	No	No	Experience Data	System Defined	Context Data	No	No
[Xiong, 03]	No	Full Mode	Yes	Yes	No	Experience Data	System Defined	No	No	No
[English, 04]	No	No	No	No	No	Experience Data	System Defined	Unspecified	No	No
[Dimmock, 04]	No	Limited Mode	No	Yes	No	Experience Data	System Defined	Context Data	Yes	No
[Shand, 04]	No	No	Yes	No	Yes	Experience Data	System Defined	No	Yes	No
[Dawn, 05]	No	Full Mode	Yes	No	No	Ontology Based	System Defined	No	No	No
[Lin, 05]	No	No	No	No	Yes	Experience Data	System Defined	Context Data	No	No
[Capra, 06]	No	Limited Mode	Yes	Yes	Yes	Experience Data	Stochastic & System Defined	No	No	No
[Chakraborty, 06]	No	Limited Mode	Yes	Yes	Yes	Experience Data	System Defined	No	Yes	Yes
[Quercia, 06]	No	No	Yes	Yes	No	Experience Data	Probabilistic & System Defined	No	No	Yes
[Teacy, 06]	No	Limited Mode	Yes	Yes	Yes	Experience Data	Probabilistic	No	No	No

Performance Comparison Model

Each article reviewed in Sections 2.2.1, 2.2.3 and 2.2.3 provides its own evaluation criteria. For example, [Xion, 02] represents the trust evolution accuracy by measuring how well the trust model helps users in making trust decision. A trust evaluation is considered accurate when a trustworthy user is evaluated as trustworthy or an untrustworthy user is evaluated as untrustworthy. In [Lin, 05], performance is measured according to the number of successful transactions in the presence of a trust management system with different number of good and malicious users (agents). In [Chakraborty, 06] the performance of the system depends on the number of recommendations used to calculate trust as well as the number of roles available that can be activated according to a user's trust. Finally, the experiments presented in [Quercia, 06] are geared towards gaining understanding on how the distribution of the confidence levels varies under simple and attach scenarios. According to the authors in [Quercia, 06], there are two kinds of [collusion] attacks for a trust system that are most likely to occur. The first is the bad mouthing collusion attack. In this attack, a collection of attackers colludes in that each of them spreads negative recommendations about the same benevolent entity. The second type of attack is the ballot stuffing collusion attack. Here we have a collection of colluding attackers: some offer services and others increase the remaining attackers' reputations as recommenders. The last subset of attackers (the good recommenders) sends positive recommendations about those in the subset of service providers. Based on the positive opinions, a victim selects the providers. They then offer a low quality of service. The victim lowers its trust level in the abusing service providers only, whereas it still deems trustworthy the remaining

attackers.

Since most of the evaluation metrics and parameters used in the reviewed articles are specific to each one of the reviewed systems, *there is no context and trust security consensus* that would enable users to compare the performance and accuracy of different security techniques. This is a serious problem because, as it stands today users evaluation a system need to become aware of the technique's specific features in order to judge the system's performance and accuracy.

To enable users to compare the performance and accuracy of different context and trust security techniques, since there is no benchmark available, experiments should be conducted by varying a standard set of dynamic parameters to study their effects based on a set of *performance* and *accuracy* metrics, while tuning different system *parameters*.

Accuracy Metrics

- Number (percentage) of accurate interactions
- Number (percentage) of inaccurate interactions
- Percentage of correct security decisions with valid context and trust information
- Percentage of correct security decisions with invalid context and trust information
- Percentage of incorrect security decisions with valid context and trust information
- Percentage of incorrect security decisions with invalid context and trust information

Performance Metrics

- Average trust/risk computation time
- Average transaction (response) or interaction time
- Average system throughput
- Average storage space used by the system to compute trust and risk
- Average context and trust computation data transmission size
- Average context and trust computation data transmission time
- Average time used for trust computation and collection of context data for bootstrapping
- Average entity trust decay time

Dynamic Parameters

- Number of entities – This is number of entities that can interact in the system at any given time. This will give users an idea how the system performs as the more entities in the system, the more interactions and [possibly] the more number of recommendations, which are used to compute the different trust and risk measures.
- Number of malicious entities – This is the number of entities that distribute incorrect (invalid) context information or give false recommendations. This will allow user to judge how resilient is the techniques against collusion attacks
- Frequency of entity interactions – This will allow users to not only have an idea of the system throughput and performance when calculating trust and risk measures but also it will aid users in determining the accuracy of the

model (risk, trust and probability measures) as the number of entities grow.

- Number of entities needed to compute trust – This will allow users to determine the accuracy of the trust measure when the number of entities needed to compute trust (i.e. number of recommenders) varies.
- Number of context data variables to collect for entity interactions – Besides aiding in measuring the system performance and storage requirements, this will aid users to determine the complexity of the system implementation, and the accuracy of the risk & trust values.
- Trust and Risk weight factors – This refers to the weight factors used to compute a trust measure whose evaluation is done in parts (i.e. Direct trust, recommended trust, etc.). This will aid users in finding the best suitable weights values to be used when computing trust
- Amount of context / interaction data to collect for trust computation. This will aid users in determining the optimal amount of data history that not only for trust-computation but also to minimize resource usage.

2.3 Modeling of Context Risk Factors Behavior

In the pervasive ad-hoc environment, context variables of a user may change in no predetermine way. For instance, a student moving about a college campus, not only go to different class rooms to attend classes based on his/her regular schedule, but also may stop at or go to unplanned places (cafeteria, library, restroom). Moreover, he/she may decide based on particular circumstances to connect his/her laptop to a power supply in order to recharge it. That is, *context risk factors* follow a *stochastic process* since their values may change over time in an uncertain way.

A stochastic process model serves as a risk measurement tool in order to attempt to characterize the future change in a process's value. Essentially, the stochastic process uses a statistical distribution to represents the evolution of its values. Moreover, stochastic processes are classified as discrete time or continue time [Hull, 99]. The former is a process where the value of the variable changes at certain fixed points in time. In the latter process, variables not only may change at any time, but also can take any value within a given range. In addition, a stochastic process (denoted as $\{Y_t: t \geq 0\}$) can be further classified as follows [Finch, 04]:

- Stationary if, for all times ($t_1 < t_2 < \ldots < t_n$) and scalar $h > 0$, the random n-vectors $(Y_{t_1}, Y_{t_2}, ..., Y_{t_n})$ and $(Y_{t_1+h}, Y_{t_2+h}, ..., Y_{t_n+h})$ are identically distributed; that is, time shifts leave joint probabilities unchanged.
- Gaussian if, for all times ($t_1 < t_2 < \ldots < t_n$), the n-vector $(Y_{t_1}, Y_{t_2}, ..., Y_{t_n})$ is multivariate normally distributed [Rose, 02].
- Markovian (see section 2.3.1)

This section introduces several key concepts for continuous variable/time stochastic process for context risk factors.

2.3.1 Markov Property

A Markov process is a stochastic process where the future value (state) of the process depends only on the present state and not on all past states [Papoulis, 84]. The Markov property entails that the probability distribution of variable at any future time does not depend on the particular 'path' followed by the variable in the past [Hull, 99]. Formally the Markov property is stated as follows [Finch, 04]:

Given an stochastic process {Y_t:t≥0} and for all (t_1<t_2<...< t_n), $P(Y_{t_n} \le y \mid Y_{t_1}, Y_{t_2}, ..., Y_{t_{n-1}}) = P(Y_{t_n} \le y \mid Y_{t_{n-1}})$*; that is, the future is determined only by the present and not the past.*

2.3.2 Continuous Time Stochastic Processes

Loosely speaking, a continuous stochastic process is a phenomenon that can be thought of as evolving in time in a random manner (i.e. location of a particle in a physical system, the price of stock in a financial market, mobile phone networks, internet traffic, etc.) [Zanten, 07]. Since, context variables may change in a random manner with time, we need a continuous time stochastic process to model their behavior. As such, in this section we introduce the main characteristics of several continuous-time stochastic processes that could help us model the behavior of a context variable.

Wiener Processes

A Wiener process is a continuous-time stochastic process denoted W_t for t≥0 such that the increment $(W_t - W_s)$ is Gaussian with mean 0 and variance *t-s* for any 0≤s<t, and increments for non-overlapping time intervals are independent [Weisstein, 06]. That is

1. If (0≤s<t) then $(W_t - W_s) \sim N(0, t-s)$, where N(μ, σ^2) denotes the Normal distribution with expected value μ and variance σ^2.

2. If (0≤s<t≤u<v) (i.e., the two intervals [s, t) and [u, v) do not overlap) then $(W_t - W_s)$ and $(W_v - W_u)$ are independent random variables.

It follows from the first property that if a variable W_t follows a Wiener process, the change ΔW_t during a small period of time Δt is $\Delta W_t = \varepsilon\sqrt{\Delta t}$, *where* $\varepsilon \sim N(0,1)$ and the variance of ΔW_t is Δt [Hull, 99].

Moreover, if we consider an increase in the value of W_t during a long period of time T (denoted as $(W_T - W_o)$), it can be regarded as the sum of the increases in W_t in N small Δt time intervals where

$$N = \frac{T}{\Delta t}, \textit{thus } (W_t - W_0) = \sum_{i=1}^{N} \varepsilon_i \sqrt{\Delta t}$$

and ε_i (i=1,2…N) ~ N(0,1) and are independent of each other. Moreover the variance of $[(W_T - W_o)]$ is $N\Delta t = T$ [Hull, 99][2].

Table 2-5 shows a simulation of a Wiener process W_t whose initial value was 20 and with a Δt = 0.001 (time measured in years). Different random samples would lead to different value movements. The final value for Wt is 20.023, which can be regarded as a random sample from the distribution of values at the en of 10 time intervals; that is at the end of one-hundredth of a year.

The Wiener process plays an important role both in pure and applied mathematics. In pure mathematics, the Wiener process has helped mathematicians to deeply understand stochastic calculus and diffusion processes (stochastic differential equations).

[2] If X ~ $N(\mu x, \sigma x^2)$ and Y ~ $N(\mu y, \sigma y^2)$ are independent normal random variables, then their sum is normally distributed with U = X + Y ~ $N(\mu x + \mu y, \sigma x^2 + \sigma y^2)$ [Grinstead, 97].

Table 2-5 Simulation of a Wiener Process with W_t =20 and Δt = 0.001

W_t	N(0,1)	ΔW_t
20.000	0.797	0.025
20.025	0.190	0.006
20.031	0.323	0.010
20.041	-0.123	-0.004
20.038	-0.396	-0.013
20.025	1.649	0.052
20.077	0.844	0.027
20.104	-1.550	-0.049
20.055	0.302	0.010
20.064	-1.305	-0.041
20.023	-0.079	-0.002

In applied mathematics, the Wiener process has been used to model random variables, such as volatility in mathematical finance, noise in electronics engineering, instruments errors in filtering theory, unknown forces in control theory and particle motion under molecular shocks in physics ([Hull, 99], [Hagen, 04]). The advantage of the Wiener process is that the Wiener measure is likely the simplest and good measure to model infinite-dimensional space (the space of continuous functions), and still it is a good model for many natural phenomena [Hagen, 04].

Generalized Wiener Process

As seen in the previous section, the normal distribution for the basic Wiener process has a drift rate (mean) of zero and a variance rate of 1.0. The zero mean indicates that the expected value of W_t at any future time is equal to its current value [Hull, 99]. Moreover, the variance of 1 denotes that a change of W_t in a T length time interval is equal to T.

A generalized Wiener process for a variable z can be defined in terms of dW_t (a

stochastic differential equation) as follows [Hull, 99]:

$$dz = adt + bdW_t$$
where a and b are constants.

The term adt implies that z has an expected drift rate per unit of time. Moreover, if we consider the only the term $dz = adt$ then it implies that

$$\frac{dz}{dt} = a \quad therefore\ z = z_o + at$$

where z_o is the value of z at time zero. The term bdW_t is thought of as adding noise to the path followed by x. The amount of noise added is b times a Wiener process [Hull, 99]. It follows, since a Wiener process has a standard deviation of 1.0, then a b times Wiener process has a standard deviation of b. Moreover, in a time interval Δt, the change of value of z, Δz would be given by [Hull, 99]

$$\Delta z = a\Delta t + b\varepsilon\sqrt{\Delta t}, where\ \varepsilon \sim N(0,1)$$

As such, Δz, denoting the discrete-time version of the model, has a normal distribution whose mean is $a\Delta t$ and a standard deviation (volatility) of $b\sqrt{\Delta t}$ (i.e. a variance of $b^2\Delta t$).

Table 2-6 shows a simulation of a generalized Wiener process variable z whose initial value was 20, Δt = 0.001, mean (drift) of 0.17 and standard deviation of 0.5.

Table 2-6 Simulation of a Generalized Wiener Process with z=20, α=0.17, b=0.5 and Δt=0.001

z	N(0,1)	Δz
20.000	-0.869	-0.01357
19.986	0.460	0.00744
19.994	0.100	0.00175
19.996	0.925	0.01480
20.010	-0.814	-0.01270
19.998	-0.461	-0.00712
19.991	1.148	0.01832
20.009	-0.166	-0.00246
20.006	-0.459	-0.00709
19.999	1.806	0.02872
20.028	-0.573	-0.00889

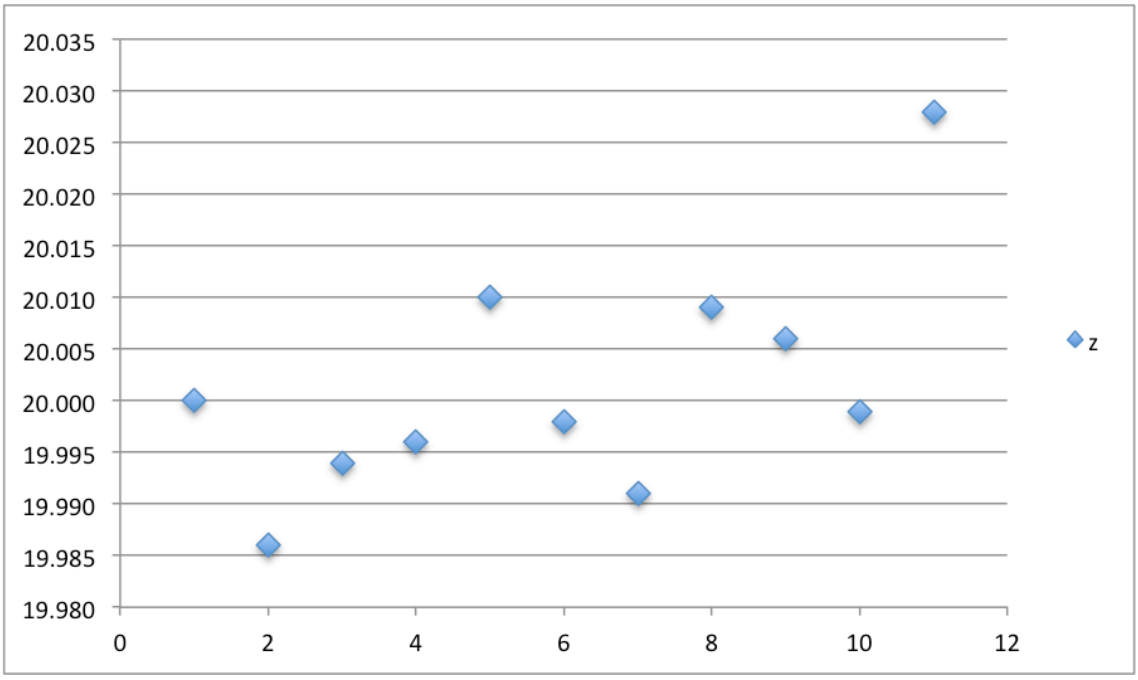


Figure 2-3 Z Values from the Generalized Weiner Process in Table 2-6

Ito Process

An Ito process is a generalized Wiener process where the drift and variance parameters are functions that depend directly on the value of *z* and time *t* [Hull, 99]. Formally, it can be described by the following stochastic differential equation:

$$dz = a(z,t)dt + b(z,t)dW_t$$

In a small time interval between t and t + Δt, the variable changes from z to z + Δz, where [Hull, 99]:

$$\Delta z = a(z,t)\Delta t + b(z,t)\varepsilon\sqrt{\Delta t}$$

The equation above assumes that the drift and variance of z remain constant and equal to $a(z,t)$ and $b(z,t)^2$ respectively, during the interval [t, t + Δt].

Geometric Brownian-Motion

A geometric Brownian motion (or exponential Brownian motion) is a continuous-time stochastic process in which the logarithm of the randomly varying variable follows a Wiener process [Ross, 03]. That is, a stochastic process S_t follows a geometric Brownian motion when it satisfies the following stochastic differential equation:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

or

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t$$

where W_t is a Wiener process and μ (the percentage drift) and σ (the percentage volatility) are constants.

The equation has an analytic solution [Ross, 03]:

$$S_t = S_o \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$$

for an arbitrary initial value S_0. Moreover, it can be shown that the random variable $\log\left(\frac{S_t}{S_0}\right)$ is normally distributed with mean $\left(\mu - \frac{\sigma^2}{2}\right)t$ and variance σ^2t, which

implies that increments of a geometric Brownian motion variable are normal relative to the current value [Ross, 03].

Based in the equations above, the time discrete version of the model is

$$\Delta S_t = \mu S_t \Delta t + \sigma S_t \varepsilon \sqrt{\Delta_t}$$

where ΔS_t is the change in the value of S_t, in a small time interval a Δt and ε is a random drawing from a standard normal distribution (i.e. $\varepsilon \sim N(0,1)$).

Table 2-7 shows a simulation of a geometric Brownian motion variable S_t, whose initial value was 20, Δt = 0.001, mean (drift) of 0.14 and standard deviation of 0.20.

Table 2-7 Simulation of a Geometric Brownian Motion with S_t=20, μ=0.15, σ=0.2 and Δt=0.001

S_t	N(0,1)	ΔS_t
20.0000	-0.1848	-0.0206
19.9794	0.7532	0.0980
20.0774	-0.6654	-0.0817
19.9957	-0.0594	-0.0047
19.9910	-0.7562	-0.0928
19.8982	1.4405	0.1841
20.0823	2.7820	0.3562
20.4384	0.8978	0.1189
20.5573	-0.9626	-0.1223
20.4351	-1.3457	-0.1711
20.2640	0.6043	0.0803

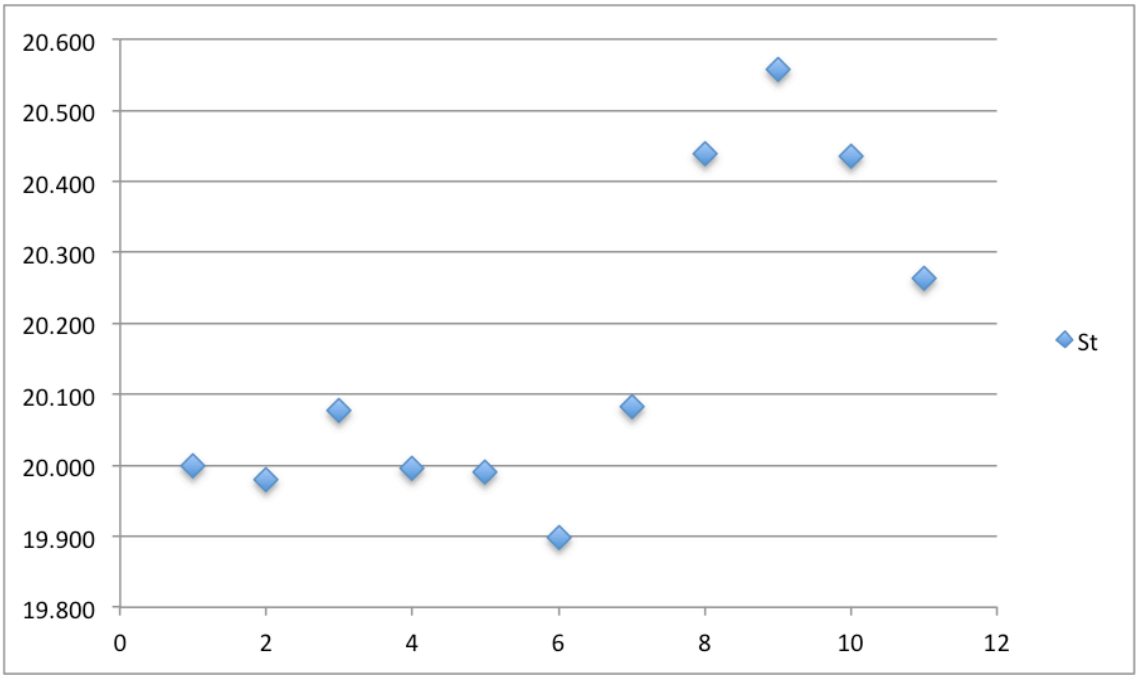


Figure 2-4 St values from the Geometric Brownian Motion in Table 2-7

Ornstein-Uhlenbeck Process

An Ornstein-Uhlenbeck process is a process that not only is stationary (in contrast to the Wiener process), Gaussian and Markovian, but also continuous in probability [Finch, 04]. A stochastic process $\{Y_t: t \geq 0\}$ is continuous in probability if, for all $u \in \Re^+$ and $\varepsilon > 0$,

$$P(|Y_v - Y_v|) \geq \varepsilon \rightarrow 0 \; as \; v \rightarrow 0$$

An Ornstein-Uhlenbeck process satisfies the following linear stochastic differential equation [Finch, 04]:

$$dY_t = -\rho(Y_t - \mu)dt + \sigma dW_t$$

where W_t is a Wiener process and ρ, μ and σ are constants.

A sample Ornstein-Uhlenbeck process (a.k.a. colored noise) over the time interval [0,T] can be generated as follows:

1. Let N be a large integer
2. Generate $z_0, z_1, \ldots, z_N$ independent random numbers taken from the standard normal distribution.
3. Define Y_0 = c where c could either be the initial condition or ($\mu + z_0$) is no constant is present.
4. Define recursively for $1 \leq n \leq N$

$$Y_n = \mu + K_N(Y_{n-1} - \mu) + \sqrt{1 - K_N^2}\, z_n$$

$$\text{where } K_N = e^{\frac{-\rho T}{N}}$$

Table 2-8 shows a simulation of an Ornstein-Uhlenbeck process with Y_0=20, T = 0.001, N = 10, μ = 0.14, σ = 0.2 and ρ =1.

Table 2-8 Simulation of an Ornstein-Uhlenbeck Process with Yo=20, μ=0.14, σ=0.2, T=0.001, ρ=1 and N=10

Y	z
20.0000	2.3168
20.0308	0.8957
20.0415	-0.1079
20.0379	-0.2286
20.0327	0.3251
20.0353	1.4966
20.0545	-0.1716
20.0501	0.7216
20.0583	0.0708
20.0573	0.4358
20.0615	-1.7044

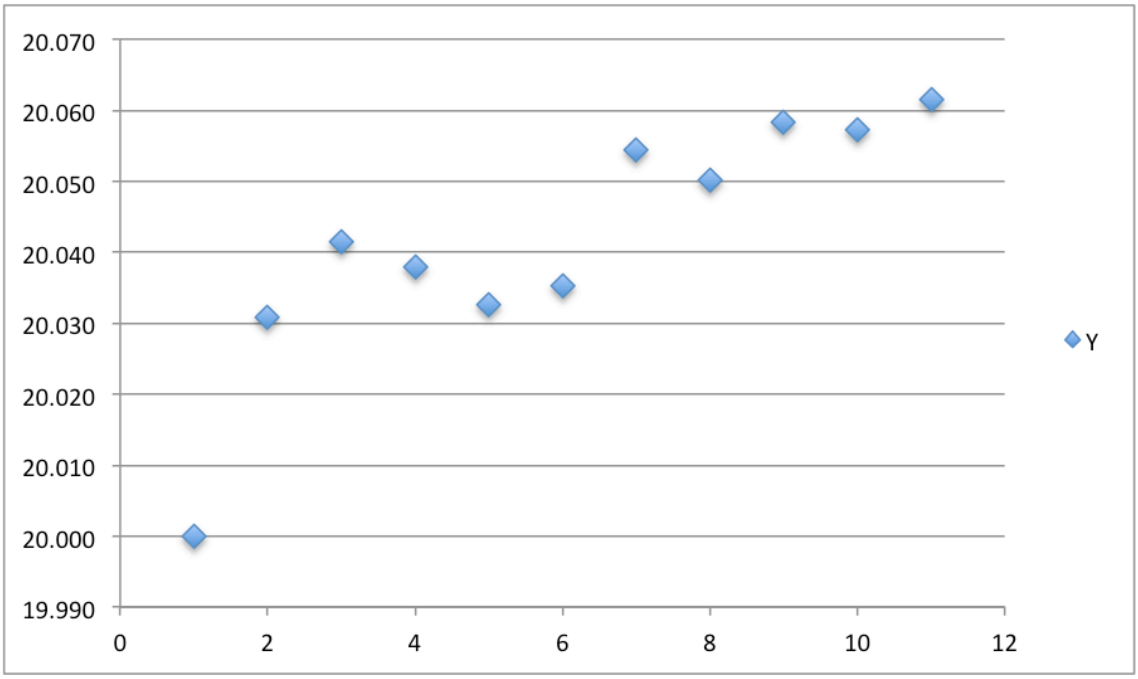


Figure 2-5 Y values from the Ornstein-Uhlenbeck Process in Table 2-8

CIR Process

The CIR process (a.k.a. squared Bessel process) is a Markov process defined by the following stochastic differential equation [Cox, 95]:

$$dr_t = -\theta(r_t - \mu)dt + \sigma\sqrt{r_t}\,dW_t$$

where θ, σ and μ are parameters. This process can also be defined as a sum of squared Ornstein-Uhlenbeck processes. The process value at time t, i.e. r_t follows a non-central Chi distribution [Stuart, 99].

Logarithmic Random Walk Model

The changes in the value of a variable can be expressed in a variety of forms i.e. absolute value change, relative value change, and log value change [RMG, 96]. When a value change is defined relative to some initial value, it is known as a return [RMG, 96]. Following the RiskMetrics standard, the change in the value of a group of variables

(e.g. a set of financial instruments such as equities, commodities, etc.) can be measured in terms of continuously compounded returns (log value changes).

A return random walk model constitutes a risk measurement where forecasts for each of the variable's future value changes - using only their past variations - are constructed. This methodology not only models the evolution of returns over time, but also the distribution of returns at any point in time.

Before defining the random walk model we need to introduce the concept of a variable return.

- **One-time (single period) value return horizon**

Denoting P_t the value of a variable at a time t, the absolute value-change on a variable between time t and $t\text{-}1$ (e.g. 1 day) is defined as

$$D_t = P_t - P_{t-1}$$

The relative value-change or percent return, R_t for the same period is

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

If the gross return on a variable is $1 + R_t$, the continuously-compounded return (log value change), r_t, of such variable is defined as:

$$r_t = \ln(1 + R_t) = \ln\left(\frac{P_t}{P_{t-1}}\right) = (p_t - p_{t-1})$$

where $p_t = \ln(P_t)$ or the natural logarithm of P_t.

Table 2-9 shows an example of proximity values (percentages) between two objects (*a* & *b*). It is readily seen that all the value changes have the same sign for any given time and the similarity of the relative and the log value changes.

Table 2-9 Proximity Values Between Two Objects a and b

Time	Proximity (a/b),Pt	Absolute change	Relative Change	Log Change
8:09	67.654	-0.078	-0.001152	-0.001152
8:08	67.732	0.310	0.004598	0.004587
8:07	67.422	-0.063	-0.000934	-0.000934
8:06	67.485	-0.119	-0.001760	-0.001762
8:05	67.604	0.059	0.000873	0.000873
8:04	67.545	0.096	0.001423	0.001422
8:03	67.449	-0.219	-0.003236	-0.003242
8:02	67.668	0.635	0.009473	0.009428
8:01	67.033	0.350	0.005249	0.005235
8:00	66.683			

- **Multiple-time (multi-period) value return horizon**

The multiple-time percent returns over the last k time periods can be defined as follows:

In terms of one-time returns, the multiple-time gross return $1 + R_t(k)$ is given by the product of 1-time gross returns.

$$1 + R_t(k) = (1 + R_t)(1 + R_{t-1})...(1 + R_{t-k-1}) = \left(\frac{P_t}{P_{t-1}}\right)\left(\frac{P_{t-1}}{P_{t-2}}\right)...\left(\frac{P_{t-k-1}}{P_{t-k}}\right) = \frac{P_t}{P_{t-k}}$$

For continuously compounded returns, the multiple-time $r_t(k)$ is defines as:

$$r_t(k) = \ln\left(\frac{P_t}{P_{t-k}}\right) = \ln\left[(1 + R_t)(1 + R_{t-1})...(1 + R_{t-k-1})\right] = r_t + r_{t-1} + ... + r_{t-k+1}$$

As seen in the equation above, the multiple-time returns based on continuous compounding are simple sums of one-time returns. Moreover, According to [RMG, 96], RiskMetrics assumes that the return of a group of N variables is a weighted average of continuously compounded returns. That is:

$$r_{gt} \cong \sum_{i=1}^{N} w_i r_{it}$$

where w_i represents the variable weights.

- **Random Walk Model for Single Value Risk Factor**

RiskMetrics models the dynamics of single-value variables using a random walk paradigm [RMG, 96].

A random walk[3] is a formalization of the idea of taking successive steps, each in a random direction; that is, it may be thought as being a model for an individual walking on a straight line who at each point of time either takes one step to the right with probability p or one step to the left with probability $1 - p$ [Hughes, 96].

The random walk model can be stated as follows:

$$P_t = \mu + P_{t-1} + \sigma\varepsilon_t$$
$$P_t - P_{t-1} = \mu + \sigma\varepsilon_t$$

where $\varepsilon_t \sim N(0,1)$. The equation above states that at any point in time the current value P_t depends on two fixed parameters μ (mean) and σ (standard deviation), the last period's value P_{t-1}, and a normally distributed random variable ε_t.

In order to avoid negative values, it is better to model the **log value** p_t as a random walk with normally distributed changes [RMG, 96].

$$p_t = p_{t-1} + \mu + \sigma\varepsilon_t \quad \varepsilon_t \sim N(0,1)$$

therefore

[3] Random walk models have been applied in several fields. For instance, in economics, a random walk is used to model shares prices and other factors. In physics, they are used to represent the random movement of molecules in liquids and gases. In psychology, random walks explain the relation between the time needed to make a decision and the probability that a certain decision will be made. Finally, in wireless networking, they are used to model node movement.

$$P_t = P_{t-1} e^{\mu + \sigma \varepsilon_t} \quad \varepsilon_t \sim N(0,1)$$

Given that ε_t is normally distributed then P_t then follows a log-normal distribution [Limpert, 01]. Moreover, contrary to the geometric Brownian motion, the assumption that the variance (σ^2) of the value changes is a constant can not only be relaxed in order to make it vary with time, but also modeled as a function of past variances. Thus:

$$p_t = p_{t-1} + \mu + \sigma_t \varepsilon_t \quad \varepsilon_t \sim N(0,1)$$

RiskMetrics further relaxes the model in equation above by assuming that log values have a mean (drift) μ set to zero[4] (see [RMG, 96] section 4.3). As such,

$$p_t = p_{t-1} + \sigma_t \varepsilon_t \quad \varepsilon_t \sim N(0,1)$$

in term of returns

$$r_t = \sigma_t \varepsilon_t \quad \varepsilon_t \sim N(0,1)$$

In order to not only compute the standard deviation (volatility) σ_t, but also to react faster to sudden changes in data (thus, allowing the latest observations carry the highest weight in a variance estimate), an exponential moving average of historical observations should be used [RMG, 96]. Therefore, for a given set of T returns, the variance can be defined as follows [RMG, 96]:

[4] Since RiskMetrics assumes the standard deviation of returns is usually much higher than the mean, the latter is neglected in the model [Pafka, 01]. Moreover, [RMG, 99] shows that return horizon predictions (value and sign) for periods of less than three months are not accurate. This occurs because for short horizon periods, the volatility is much larger than the expected return; thus, the forecast of the future return distribution is dominated by the volatility estimate. In other words, when dealing with short horizons, using a zero expected return assumption is as good as any mean estimate that could provide.

$$\sigma_t^2 = \frac{\sum_{i=1}^{T} \lambda^{i-1} \left(r_{t-i} - \mu_r\right)^2}{\sum_{i=1}^{T} \lambda^{i-1}}$$

where the parameter λ *(0 < λ <=1)* (***decay factor***) determines the relative weights that are applied to the observations (returns) [RMG, 96]. In addition, when $T \to \infty$ setting the mean (drift) μ_t and using the geometrics series $\sum_{i=1}^{\infty} \lambda^{i-1} = \frac{1}{1-\lambda}$ the above equation can be rewritten as follows:

$$\sigma_t^2 = (1-\lambda) \sum_{i=1}^{\infty} \lambda^{i-1} r_{t-i}^2$$

in a recursive manner, the equation becomes

$$\sigma_{1,t}^2 = \lambda \sigma_{1,t-1}^2 + (1-\lambda) r_{1,t}^2$$

Table 2-10 shows the calculation of the volatility for the proximity data in Table 2.9. The decay factor λ used was 0.95. The standard deviation (volatility) is equal to 0.394% (variance is 0.00155%)

Finally, assuming not only that no-autocorrelation for the data (returns in different periods are independent), but also the volatility does not change; then the variance of a sum (T time units) is the sum of the variances for independent variables, that is [RMG, 96]:

$$\sigma_{T-time.unit}^2 = T \sigma_{1-time.unit}^2$$

Table 2-10 Estimating Standard Deviation (Volatility) for Proximity Values with λ=0.95

Time	Proximity (a/b),Pt	log change	σ^2_t
8:09	67.654	-0.00115	0.00115
8:08	67.732	0.00459	0.00331
8:07	67.422	-0.00093	0.00278
8:06	67.485	-0.00176	0.00258
8:05	67.604	0.00087	0.00237
8:04	67.545	0.00142	0.00226
8:03	67.449	-0.00324	0.00204
8:02	67.668	0.00943	0.00379
8:01	67.033	0.00523	0.00394
8:00	66.683		

The algorithm to simulate future values for one factor is stated as follows:

1. Determine the number of scenario trials S and decay factor λ
2. Compute the 1 time unit volatility from the observed data
3. Compute the H time horizon volatility as $\sigma_H = \sigma_t \sqrt{H}$
4. For each trial i $(1 \le i \le T)$
    1. Simulate Z = φ-1 (X) ~ N(0,1)
    2. Compute the simulated value as $P_t = P_{t-1} \exp(\sigma_H Z)$ where exp(x) = e^x

Table 2-11 shows 5 simulation trials for a proximity values with P_t= 68 and σ_t = 0.394%.

Table 2-11 Simulation Trials for a Proximity Value P_t=68, δ=0.394% (τ=10)

Trial	Z	P_s
1	0.41405	68.11102
2	-1.53471	67.59006
3	-1.23825	67.66906
4	-0.76927	67.79421
5	-0.37158	67.90052
6	1.58510	68.42601
7	0.30701	68.08230
8	1.22780	68.32975
9	1.54738	68.41584
10	1.14262	68.30682

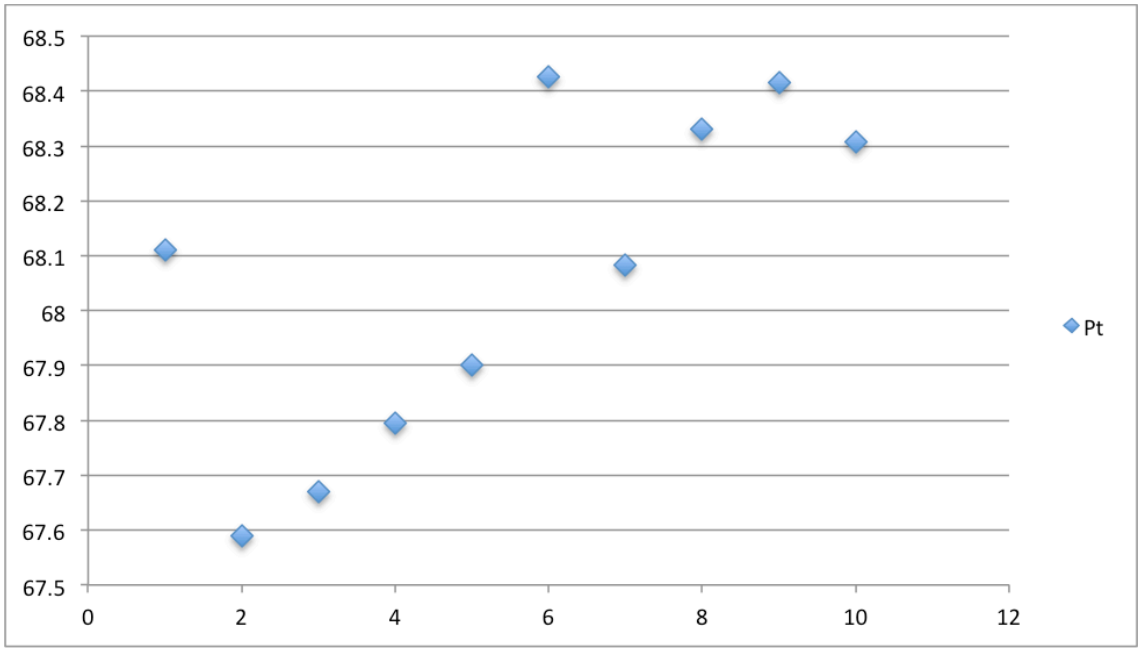


Figure 2-6 Simulated Proximity Values from Table 2-11

2.4 Time Series Classification

Time series data can be found in many avenues of human endeavor including medicine, aerospace, finance, entertainment, and industry [Xi, 06]. The task of time series classification is to map a time series to one of specific class. Several algorithms have been proposed as found in [Rodriguez, 00], [Geurts, 01], [Nanopoulos, 01],

[Ratanamahatana, 04], [Wu , 04], [Wei, 06], [Xi, 06], [Ding, 08], [Ye, 09], [Ye, 11] and [Rakthanmanon, 12]. Most time series data mining algorithms use similarity and or distance comparisons at their core, and there is increasing evidence that the classic Dynamic Time Warping (DTW) [Müller, 07] measure is the best measure in most domains [Ding 08] [Rakthanmanon, 12]. However, space and time requirements are a disadvantage as well as it does not yield useful information about why a particular time series was assigned to a particular class [Ye, 11]. Conversely, time series shapelets are time series subsequences that are accurate representation of a class [Ye, 09].

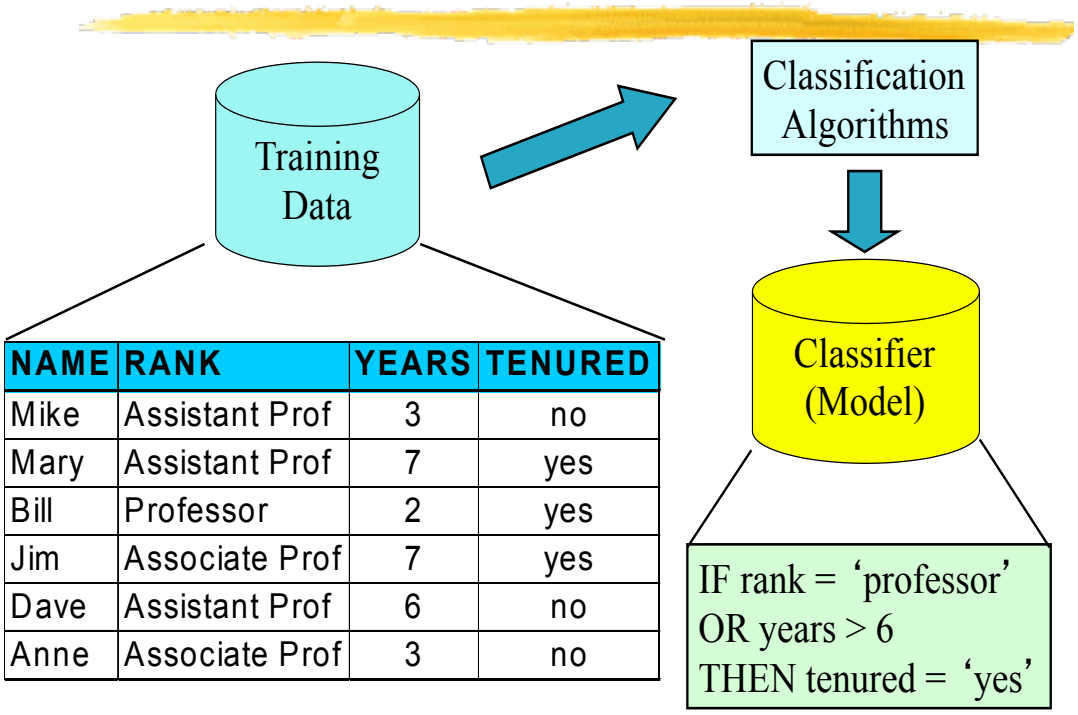
In this section we would give a brief introduction to time series shapelets as well as some concepts related to data classification.

2.4.1 Data Classification

Given a collection of records (called the training set or training data) where each record has a set of attributes, one of which is the class, classification refers to the finding of a model that predicts the class as a function of the values of other attributes, with the objective that previously unseen records should be assigned a class as accurately as possible [Tan, 05].

Several models for data classification are used currently ([Tan, 05], [Witten, 11]).

- Decision Tree based Methods
- Rule-based Methods
- Neural Networks and Memory based reasoning
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

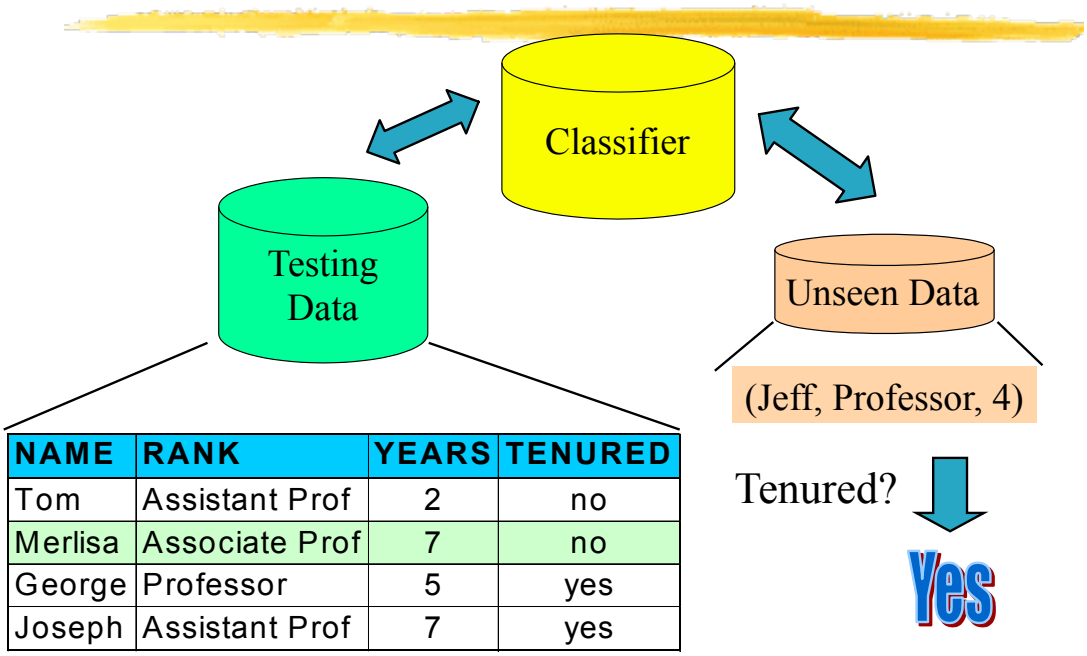


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Figure 2-7 A Sample Training Data [Saleeb, 08]

Figure 2-7 shows an example set of training records that shows the years of experience, the rank and whether is a professor is tenured. In this case, the class attribute is called *Tenured.* A classifier is used to build a model for the training data.

Figure 2-8 shows a how the model is used for classifying unknown objects. Using training data, the classifier attempts to find the best class label for a given unknown (unseen) data. The accuracy rate of the classifier is the percentage of test set samples that are correctly classified by the model.



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Figure 2-8 Classifying Unknown Records [Saleeb, 08]

In this dissertation, we will make use of decision trees, which we will briefly introduce in the next section

2.4.2 Decision Trees

According to [Quinlan, 86], [Murthy, 98], [Rokach, 02] a decision tree is a classifier expressed as a recursive partition of the universe of records. The Decision tree consists of nodes that form a rooted tree making it a directed tree with a node called root that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called internal node or test nodes. All other nodes are called leaves (also known as terminal nodes or decision nodes). Figure 2-9 shows a tree to decide to play or not play based on climate conditions. In this case, outlook is in the

position of the root node. The degrees of the node are attribute values. In this example, the child nodes are tests of humidity and windy, leading to the leaf nodes which are the actual classifications. This example also includes the corresponding data, also referred to as instances. In our example, there are 9 "play" days and 5 "no play" days. In the decision tree each internal node splits the set of records into two or more subsets according to a certain function of the record attributes values. In the simplest case, each test considers a single attribute, such that the set of records is partitioned according to the attribute's value [Rokach, 02].

The task of constructing a tree from the training set has been called tree induction, tree building and tree growing. Typically the goal is to find the optimal decision tree by minimizing the generalization error. Most existing tree induction systems proceed in a greedy top-down fashion, starting with an empty tree and the entire training set, a similar algorithm to the following is applied until there are no more splits are possible [Murthy, 98], [Quinlan, 86].

1. If all the training examples at the current node *t* belong to class category *C*, create a leaf node with the class *C*.
2. Otherwise, score each one of the sets of possible splits *S*, using a *goodness measure*.
3. Choose the best split *s**, as the test at the current node.
4. Create, as many child nodes as there are distinct outcomes of *s**. Label edges between the parent and child nodes with outcomes of s*, and partition the training records using s* into the child nodes.

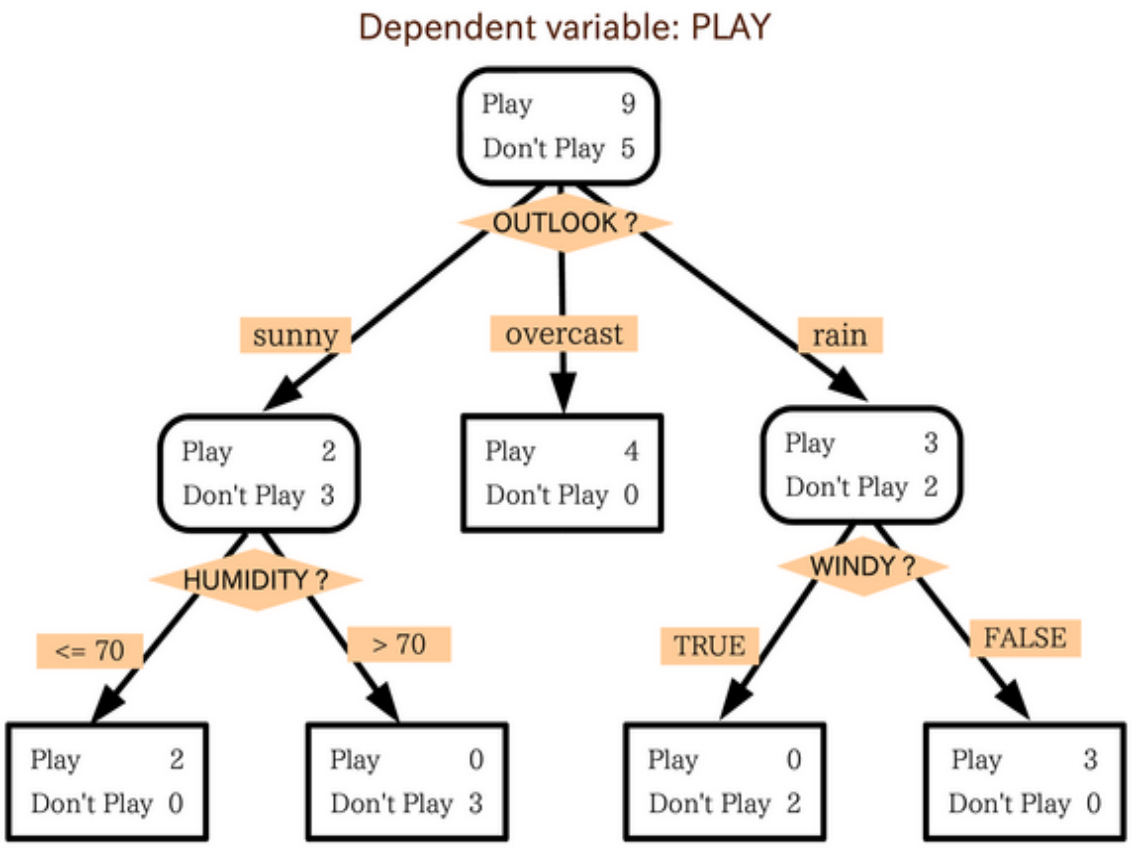


Figure 2-9 A Decision Tree [Monk, 13]

5. A child node *t* is said to be pure if all the training samples at *t* belong to the same class. Repeat the previous steps on all impure child nodes or a *stopping criteria* has been triggered. For instance,
    a. The maximum tree depth has been reached
    b. The number of cases in the terminal node is less than the minimum number of cases for parent nodes.
    c. If the node were the split occur, the number of cases in one or more child nodes would be less than the minimum number of cases for child nodes.
    d. The best splitting criteria - information gain - is not greater than a certain threshold [Rokach, 02].)

To build a decision tree, the training data is split into smaller and smaller subsets

in a recursive top-down fashion. Starting with all the training records at the root node, at each node a split function is evaluated in order to divide the training data into subsets accordingly. In most of the cases, the splitting functions are univariate, that is, an internal node is split according to the value of a single attribute. Consequently the inducer searches for the best attribute upon which to split [Rokach, 02]. When there are multiple attributes, the splitting criterion is usually based on linear combination of the record attributes [Murthy, 98]. Once the tree has been constructed, tree builder algorithms usually try to find a simplified tree in order to optimize for tree height and balance as well as increasing the accuracy and error estimation [Frank, 00].

In many applications, it is not simply enough to predict the class for a test sample. What is needed instead is to give a ranking of the probabilities. That is, a correctly calibrated-measure for the true probability that a test record is a member of the class of interest [Zadrozny, 01]. For a given class C, a simple way to calculate this probability is simply to compute the training frequency. On a leaf node, the probability score is measured to be $p=k/n$, where k is the number of training records of class C that fell into the node, and n is total training examples on that node. Finally, classifiers make use of Laplacian correction (or Laplace estimator) in order to handle zero probability values [Provost, 00].

2.4.3 Time Series Shapelets

Informally, shapelets are time series subsequences, which are in some sense maximally representative of a class [Ye, 09]. To formally define a shapelet, we need to introduce the following concepts [Ye, 09]. [Ye, 11]:

Subsequence. Given a time series T = $t_1,..t_i,.,t_m$ of real values of length m, a subsequence S of T is a sampling of length $l \le m$ of contiguous positions from T. That is, $S = t_p,...,t_{p+m-l+1}\, for\, 1 \le p \le m-l+1$.

Sliding Window: Given a time series T of length m, and a user-defined subsequence length of *l*, the set of all possible subsequences of size *l* across T is defined as: $S_T^l = \{S_p^l\, of\, T, for\, 1 \le p \le m-l+1\}$

Distance: The distance between two time series (T, R) with the same length - denoted as *Dist(T, R)* - is simply the Euclidian distance defined as

$$Dist(T,R) = d = \sqrt{\sum_{i=1}^{n}(t_i - r_i)^2}$$

The distance from a time series T to a subsequence S, denoted as *SubsequenceDist(T, S)* is defined as follows:

$$SubsequenceDist(T, S) = min(Dist(S, S')), for\, S' \in S_T^{|S|}.$$

That is, *SubsequenceDist(T, S)* is simply the distance between S and its best matching location somewhere in T, as shown in Figure 2-10.

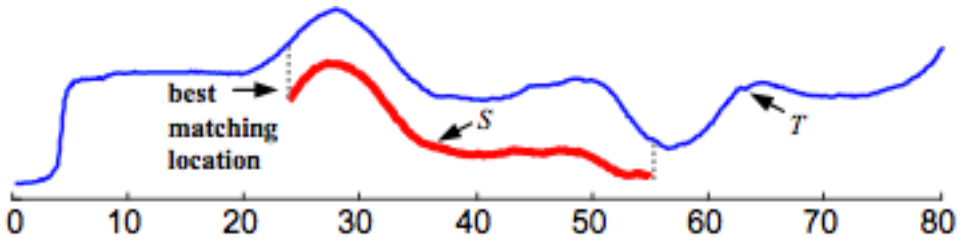


Figure 2-10 An Illustration of best matching location in a time series T for subsequence S [Ye, 09]

Entropy: A time series dataset D consists of two classes, A and B. Given that the proportion of objects in class A is p(A) and the proportion of objects in class B is p(B), the entropy of D is: $I(D) = -p(A)\log(p(A)) - p(B)\log(p(B))$.

Information Gain: Given a certain split strategy *sp* which divides D into two subsets D1 and D2, the entropy before and after splitting is $I(D)$ and $\hat{I}(D)$. So the information gain (*Gain*) for this splitting rule is

$$Gain(sp) = I(D) - \hat{I}(D),$$

$$Gain(sp) = I(D) - (f(D1)I(D1) + f(D2)I(D2))$$

where $f(D1), f(D2)$ are the fraction of objects in $D1$ and $D2$ respectively

Optimal Split Point (OSP): A time series dataset D consists of two classes, A and B. For a shapelet candidate S, we choose some distance threshold d_{th} and split D into $D1$ and $D2$, such that for every time series object $T_{1,i}$ in $D1$, $SubsequenceDist(T1,i, S) < d_{th}$ and for every time series object $T_{2,i}$ in $D2$, $SubsequenceDist(T2,i, S) \geq d_{th}$. An Optimal Split Point is a distance threshold such that

$$Gain(S, d_{OSP(D, S)}) \geq Gain(S, d'_{th})$$

for any other distance threshold d'_{th}.

Separation Gap of a Split (SGS): The separation gap of a split S is defined as:

$$\frac{1}{|D1|}\sum_{x \in D1} SubsequenceDist(x,S) - \frac{1}{|D2|}\sum_{y \in D2} SubsequenceDist(y,S)$$

Shapelet: Given a time series dataset D which consists of two classes, A and B, $shapelet(D)$ is a subsequence that, with its corresponding optimal split point,

$$Gain(shapelet(D), d_{OSP(D,\ shapelet(D))}) \geq Gain(S, d_{OSP(D,\ S)})$$

for any other subsequence S.

Table 2-12 Algorithm to Find a Shapelet

```
candidates ←generateCandidates(D, MAXLEN, MINLEN)
bsf_gain ← 0
for each S in candidates
      gain ←CheckCandidate(D, S)
      if gain > bsf_gain
         bsf_gain ← gain
         bsf_shapelet ← S
      endIf
end for
return bsf_shapelet
```

Table 2-12 shows the template that algorithms that extract a shapelet from a time series data set usually follow. In this case, *MINLEN* and *MAXLEN* represent the minimum and maximum length of the shapelet candidates that can be generated from a dataset D. The function *GenerateCandidates* computes the entire set of possible shapelet candidates; that is, all subsequences of all possible lengths. Finally, the function *CheckCandidate* finds the optimal split point (or split distance) thus dividing the time series data set into two subsets which allow the computation of the information gain for a given shapelet candidate. In [Muen, 11], for a given candidate shapelet S, the split procedure divides the data set D into two disjoint sets D1 and D2 such that $D1 = \{x : x \in D, sdist(S,x) \leq \tau\}$, $D2 = \{x : x \in D, sdist(S,x) > \tau\}$ where *sdist* can either be:

1) The Euclidian distance, as stated before.
2) The normalized Euclidian distance

$$NED(T,R) = d = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - r_i)^2}$$

3) The Dynamic Time Warping distance ([Keogh, 01], [Keogh, 04])

Finally, [Ye, 09], [Ye, 11], [Muen, 11] and [Zakaria, 12] discuss several optimizations to the finding of time series shapelets algorithm.

Figure 2-11 depicts the shapelet extracted from a time series. S can be thought of as particularly distinguishing subsection of T.

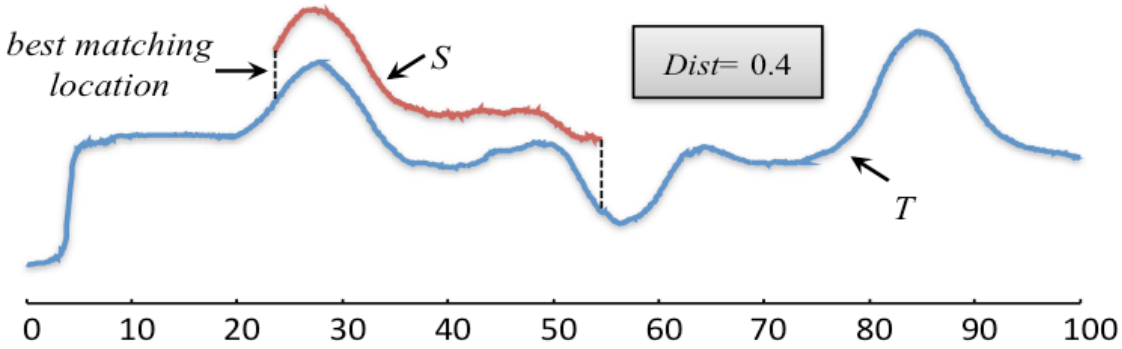


Figure 2-11 Sample Shapelet [Li, 13]

2.4.4 Shapelets for Classification

Classifying with a shapelet and its corresponding split point produces a binary decision as to whether a time series belongs to a certain class or not, since the algorithm can track from what time series the a shapelet came from [Ye, 9]. In [Ye, 11] and [Muen, 11], two simple decision tree classifiers based on time series shapelet are introduced. In each case, starting from a root node, the decision tree is constructed by adding two nodes to the tree after the algorithm finds a best split so far (*bsf_gain,*

bsf_shapelet). Moreover, not only shapelet classifiers are usually more compact than other classification approaches (hence classifying new instances is faster), but also, shapelets allow for the detection of shape-based similarity of subsequences [Lines, 12].

Given that MINLEN and MAXLEN denotes the minimum and maximum shapelet size where MINLEN $\geq$ 3 and MAXLEN $\leq$ m, the number of candidate shapelets is $O(m^3)$; therefore, it is best to find the best *k* shapelets (where *k* is estimated using a simple cross-validation approach) and finally, use these shapelets to transform data by calculating the distances from a series to each shapelet [Lines, 12]. Table 2-13 shows pseudo code to find the best k shapelets from a set of time series

Similarly to the algorithm in Table 2-12, the algorithm to extract the best k shapelets also starts by generating all possible candidates of a time series. For each candidate, the associated measures (information gain, distances, etc.) are kept. Once all candidates of a series have been examined, the candidates are sorted and 'similar' shapelets are removed. According to [Lines, 12], two shapelets are 'similar' if they are extracted from the same time series and have any overlapping indices. Once the 'similar' shapelets have been removed, the algorithm only retains the top k shapelets (see merge function) and continues to iterate through until all series have been processed.

[Bagnall, 12] and [Lines, 12] show that shapelets can be used with not only decision trees, but also with a different set of classification algorithms. That is achieved by using the *k* shapelets found in the algorithm described previously to transform instances of training data into a number of features that can then be treated as a generic classification problem. The transformation is described in Table 2-14.

Table 2-13 Algorithm to Extract the k Best Shapelets

```
kShapelets = ∅
for all time series Ti in T do
        shapelets = ∅
        for l = MIXLEN to MAXLEN do
                Wi,l =generateCandidates(Ti, MIXLEN, MAXLEN)
                for all subsequence S in Wi,l do
                        DS = findDistances(S,Wi,l)
                        quality = assessCandidate(S, DS )
                        shapelets.add(S, quality)
                end for
        end for
        sortByQuality(shapelets)
        removeSelfSimilar(shapelets)
        kShapelets = merge(k, kShapelets, shapelets)
end for
return kShapelets;
```

Table 2-14 Transformation Process Using Shapelets

```
output = ∅
for all time series ts in D do
        transformed = ∅
        for each shaplet s in kShapelets do
                dist = SubsequenceDist (ts, s)
                transf ormed.add(dist)
         end for
        output.add(transformed)
end for
return output
```

Once the k best shapelets have been found, the algorithm loops through each instance of time series Ti in the training data, computing the subsequence distance between Ti and each of the k best shapelets. The resulting k distances form a new record in the transformed data, where each attribute corresponds to the distance from each shapelet to the original time series [Lines, 12].

Finally, in order to classify an unknown time series U, we compute the distance between U and the k best shapelets extracted from the training set. Then using the

transformed training data obtained from algorithm in presented in Table 2-14 as main input, we could use a well-known classification tree algorithm (i.e. J48 [Hall, 09]) to classify the unknown time series using the k-computed distances (J48 can provide also with class ranking probabilities)

Chapter 3. SECURITY FRAMEWORK

To address the issues and shortcomings of the systems exposed in the previous two chapters, we propose a trust based security framework that includes context data for the pervasive ad-hoc environment. In this chapter, we introduced the system architecture assumed in this work followed by the definition for some mobile context risk factors.

3.1 System Architecture

The pervasive ad-hoc environment assumed in this work (for context-based security) is a collection of integrated and cooperating mobile and static components interconnected by a series of fixed and wireless networks. In this environment, there are not only the typical components of a mobile ad-hoc network (e.g. cellular phones, PDAs, mobile computers, etc.), but also a new emerging set of entities such as smart rooms, information sensors and smart wearable devices. Each component can be viewed as an independent entity in the framework with its own security implementation and policies. In addition, an entity can display the following types of autonomy [Essmayr, 95]:

- Design autonomy: a component chooses the design, representation and semantic of data.
- Communication autonomy: a component chooses when to communicate with another entity and how to respond to a request.
- Execution autonomy: a component decides when to execute any request by any other requesting entity.

- Association autonomy: a component decides not only how much data is exposed when collaborating with others, but also when to join or leave any collaboration group.

In this pervasive ad-hoc environment, a component providing a service is not only capable of querying but also using sensing and computational nodes in order to deliver such service to a community of users [Nixon, 04]. Furthermore, the pervasive ad-hoc environment exhibits the following characteristics ([Nixon, 04], [Langeheinrich, 01]):

1. Ubiquity: The essence of the pervasive and ad-hoc environment is the quality of seeming to be everywhere at once.
2. Invisibility and Sensing: Many components in the infrastructure will be imperceptible to the user and will have the ability to measure certain aspects of the environment (e.g. temperature, noise, light, position, etc.).
3. Memory and computing amplification: With not only the potential of a massive number of components (computer, sensors, etc.), but also with the advent of new software and hardware technologies, it has become easier for users to share and use memory and computing (processor) resources in order to achieve multiple complex tasks. Many of these devices can potentially store, observe and give opinions about many user interactions in multiple situations.
4. Mobility: The pervasive ad-hoc environment can be viewed as a complex space made of multiple mobile entities. These entities can range from mobile components such as PDAs, laptops and mobile phones to RFID tags. Mobile

entities can not only communicate with stationary objects, but also establish collaboration groups amongst themselves and are free to join or leave them at any time.

5. Trust: With many entities (users and data/service items) in the environment, an entity more often than not will interact with others alien to it. Therefore, as two any given entities interact; a measure of tolerance (acceptance) is developed in order to aid in the delivery and/or use of a service.

The model shown in Figure 3-1 depicts the system architecture for the pervasive ad-hoc environment. Four different entities or main components are found in this model: mobile and fixed hosts, sensors and services. Some of the fixed hosts, called base stations or Mobile Support Stations (MSS), are augmented with a wireless interface to communicate with mobile hosts. Each mobile unit communicates with one base station. The area covered by a base station is called a cell, and the process during which a mobile host enters a new cell is called hand-off. Within the boundaries of a cell broadcast is physically supported [Pitoura, 93]. A mobile host can also be part of an ad-hoc group. Ad-hoc groups can be established at any moment to enable collaboration and information exchange amongst a set of mobile users. In addition, mobile and fixed hosts can be part and make use of the services offered by a smart room. A smart room is an emerging intelligent environment that brings together computer vision, speech recognition, and sensors. Sensors are used to provide units with contextual or environmental information (i.e. temperature, noise level, etc.), which may aid the units in accessing and using services.

In the pervasive ad-hoc environment, mobile hosts are portable computers that

vary in size, processing power, memory, etc. Typically mobile hosts will have limited resources compared to theirs desktop counterparts. These limitations may include battery power, processing power, volatile memory, disk space, network bandwidth, etc. Due to the unreliability of the communication medium as well as limited resources available, the mobile host may operate in one of many modes ranging from highly connected to disconnected. Moreover, other nodes (fixed and mobile) will be able to predict that another is going out of range by monitoring the strength of the signal.

Amongst the functions that all components play, we distinguish four roles that are necessary in this context-based trust security framework: a *Context Owner* (CO) [Hulsebosch, 05], a *Context Provider* (CP)], a *Context Security Analyzer* (CSA) and *Context Accessible Object* (CAO). The first two roles are adopted from [Hulsebosch, 05].

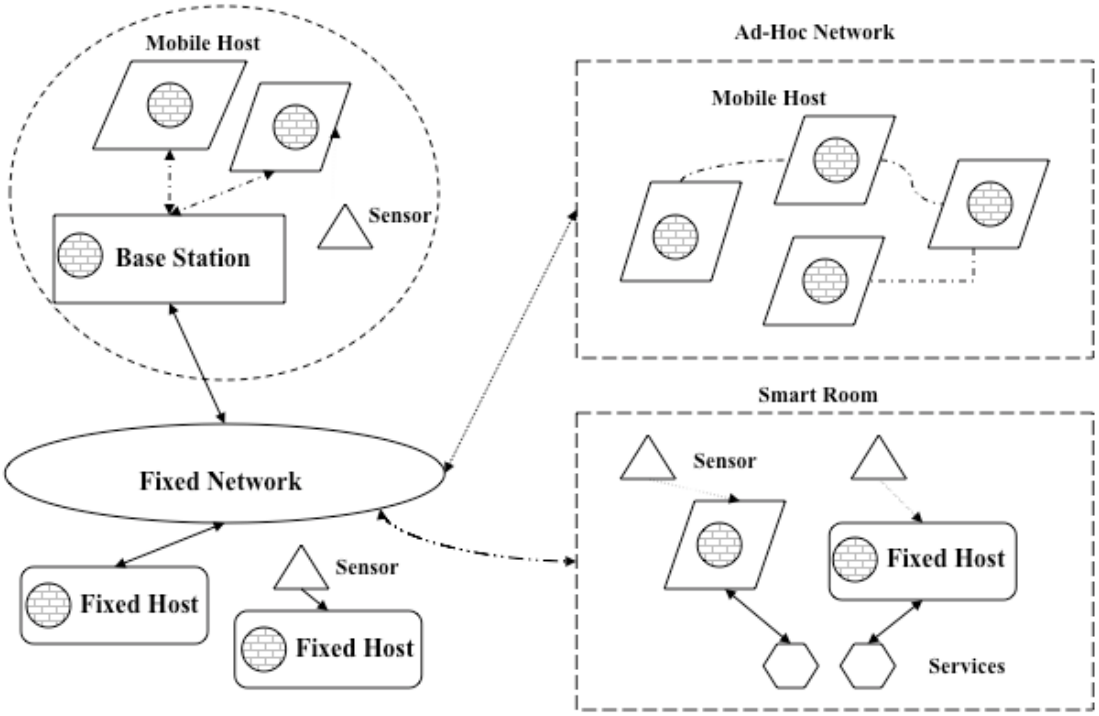


Figure 3-1 System Architecture

3.1.1 Context Owner (CO)

A *Context Owner* role is played by any entity that owns and possibly collects context data and decides which, when, where, how and by whom the context data may be stored, distributed and processed [Hulsebosch, 05]. For instance, a *CO* could be an agent that visits multiple web sites and collects data such as server response, number of links, etc. This agent decides autonomously when to store and release the collected information. Another example of a *Context Owner* can be a fired missile that stores air temperature, velocity, GPS location and terrain surveying data when traveling to its target. In this architecture, the context owner function can be assigned to any host (fixed/mobile) or sensor.

3.1.2 Context Provider (CP)

Following [Hulsebosch, 05], an entity playing the *Context Provider* role not only controls the access to the context data according to the *CO's* policies and restrictions, but also provides means to index, publish and create usage and history patterns about the context information. A *Context Provider* may use ontologies in order to reason about any other context related data. For instance, if a user is located at United Nations Headquarter, the user's *Context Provider* can deduce that he/she is in the city of New York.

The role of *Context Provider* can be assigned to any *Context Owner* (user and sensors). That is, each *CO* is responsible for storing and responding to context data queries made by other entities. However, in a typical wireless network setting, the most powerful nodes (by power availability and internal resources) can also play the role of context providers. In such a case, a mobile user will periodically deliver relevant

context data to a given set *CPs*. *CPs* can broadcast the context information or reply directly to queries for context data made by other entities. Moreover, they can deduce and store related context data such as signal strength and relative location. Context data can be forwarded from one *CP* to another as users move through the network.

In an ad-hoc group, there may be one or more units that play the *Context Provider* role. These mobile units should be powerful enough to store and release context data on behalf of other users. Similarly, limited mobile units will deliver context data to a particular *Context Provider* in the group. In case a particular *Context Provider* unit is about to go offline, a new *Context Provider* must be selected and the context data must be hand off from the old to the new provider.

Finally, in a smart room, the *Context Provider* role can be played directly by the sensors located in it and/or by a dedicated fixed station (usually a server).

3.1.3 Context Accessible Object (CAO)

Any service or object whose access is controlled by a security implementation that may have been expanded with context data constraints is assigned the role of *CAO*. For example, a battlefield map will be a *Context Accessible Object* because its access may depend on the location of the accessing users.

3.1.4 Context Security Analyzer (CSA)

An entity playing the *Context Security Analyzer* takes context information (or *risk factors)* from multiple *CPs,* and according to a security policy, it either grants or denies access to a *CAO*. *Context Security Analyzers* must take into account the level of trust amongst the different components before rendering a decision. For instance, in

many cases the confidence in the context data coming from a *CO* or a *CP* needs to be adjusted in order to reflect different degrees of reliability. Moreover, CSAs have context factor trained classifiers that allows them to compute the probability of that a context factor flows from a specific context provider.

Based on the nature of the collaboration between entities, the *Context Security Analyzer* role can be played by a fixed host, a server in a smart room or a base station in a typical wireless network setting. In an ad-hoc group, the role can be assigned to a powerful mobile device.

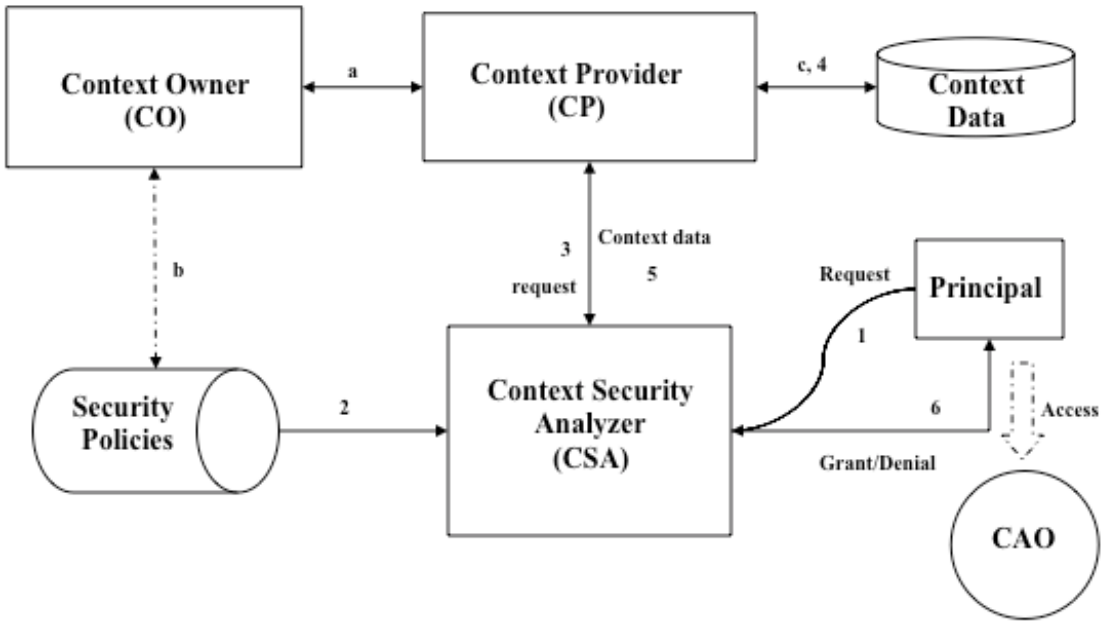


Figure 3-2 Context Based Security Framework

3.1.5 Context Based Trust Security Framework (CTSF)

Figure 3-2 illustrates the context based trust security framework. Whenever a security request from a principal (accessing entity) arrives (step 1), the *CSA* reads the security policies for the requested *CAO* and the accessing principal (step 2). Furthermore, the *CSA* may contact one or more CP_s in order to get the context data (*risk*

factors) specified in the security policies (step 3). Context providers retrieve the requested information and pass it along to the CSA (step 4 and 5). Finally, the *CAS* calculates different risk and trust values, which are used to either, grant or deny the access request (step 6).

3.2 Context Risk Factors

A context risk factor is a variable that is accessible to an entity and the movement (value change) of it may impact the accessibility of an object in a context based security system. In other words, a context risk factor is measurable information that surrounds a user, which may have an implication in the outcome of a user's task. Usually a context factor may be described as follows:

- Context type: Category of context or feature
- Value: Value of context (measured or calculated)
- Confidence: Describes the uncertainty of the value (i.e. probability)
- Source: Context value source
- Timestamp: Latest time the context occurred
- Attributes: Additional freely defined information

In the pervasive ad-hoc environment, there are many types of context risk factors. Amongst them, for instance, we can easily identify the following: location, identity, velocity, direction, network/power resources, time, suspicion level, and user intention. In this section, we define some of the context risk factors that can be considered in a context based security system as well as sample ways to provide a measurement value for such variables.

3.2.1 Location

Location can be referred to as the position in a space where an object is or could be [MWD, 13a]. In the mobile computing environment, the accuracy of a particular location depends on the signal (i.e. radio) characteristic, response time needed, mobility level or speed, network topology and the positioning scheme used. Many applications (e.g. targeting and guidance for rocket and missile systems) require not only horizontal, but also vertical positioning along speed, direction and a certain confidence level with the location estimate.

According to [Varshney, 03], an extensive wireless coverage is achieved by providing outdoor and indoor coverage to users (fixed / mobile) in a particular environment. Moreover, users can be tracked and located according to one or more of the following schemes:

- Satellite based systems. In a system such as GPS (Global Position System), 24 satellites broadcast specially coded satellite signals, which are processed by receivers to compute position, velocity and time. Mathematically four GPS satellite signals are needed to compute exact positions in three dimensions [Trimble, 06].
- Cellular Wireless Networks. In systems such as the PCS (Personal Communications Service), GSM (Global System for Mobile Communications), a user location does not change provided that he/she does not move from a particular area. The location accuracy depends directly on the size of the area [Redl, 95]. However, new infrastructures such as Enhanced Wireless Services - E911 - can allow network tracking between

within 50 to 300 meters of accuracy [FCC, 06].

- WLANS, PANS, RFID. Wireless Local Area Networks and Personal Area Networks are often used to increase the location accuracy in indoor environments [IEEE802, 07]. Radio Frequency Identification can be used to increase the location accuracy significantly. For instance, RFID-radars were initially developed to versions were specified for 0.5 meter accuracy [RFID, 06]

Due to the nature of the pervasive ad-hoc environment, multiple location and tracking schemes may be used. Each of these schemes has different degrees of accuracy with respect to an entity's location. To cope with these differences, *proximity* is a common technique used to prove an entity's location, i.e. whether a particular user is in the neighborhood of a trusted reference point [Hulsebosch, 05]. We can define proximity as follows:

Definition 1 Proximity

P*roximity* can be defined as the degree of "nearness" between two given locations [MWD, 13]. For example, mathematically a measure of proximity can be defined as follows:

Given *lx* and *ly* locations for entities *x* and *y,* respectively, the proximity *P* between *x* and *y* is denoted as

$$P(x,y) = 100\left(1 - \frac{d(lx,ly)}{MaxDist}\right)$$

where:

- *d(lx, ly)* is the Euclidian between x and y.
- *MaxDist* is the maximum distance in the evaluation area, to allow us to

represent the proximity as a percentage value.

3.2.2 Velocity

The velocity of an object is simply its speed in a particular direction that represents a physical quantity of an object's motion [TPR, 07]. For example, a way to measure the average velocity of an entity *e* moving a distance *d* (in a straight line) during a time interval (Δ_t) can be described as:

$$Ve = \frac{d}{\Delta_t}$$

There are other ways to measure velocity, for instance, in [Wan, 99] velocity is characterized in terms of velocity classes. A velocity class is defined as a range of velocity. For example, in one class, velocity is measured as the number of moves (e.g. cell changes in a cellular network) done during a period of time. More advanced velocity classes are described using of the normal distribution to not only model the velocity of a user but also to compare the cost of different location and tracking schemes. For instance [Jugl, 00] analyzed two different velocity models based on the uniform and normal distributions in order to measure the signaling cost when placing a call in a cellular network. These models found some insufficiencies that gave rise to a new a new velocity probability distribution used for analytical and performance calculations.

3.2.3 Identity

According to [Clauβ, 05], when an entity interacts with another, it usually decides the amount of internal (personal) information to be released depending on the trustworthiness (degree of knowledge) of the opposite collaborating entity. In many

occasions, an entity stores either fully or partially the identity (represented by set of attributes) of another collaborating partner according to the conditions set in a contract. When this is the case an *Identity Management System* (IDM) is used to collect and control the release of identity information. For instance, a user buying books over the Internet is required by the vendor to give an address and a credit card number in order to complete the purchase. However, if the same user is doing her taxes electronically, he/she is required to provide the tax agency with additional personal data such as social security number or a digital signature. As such, depending on the situation and the context, each entity may decide which partial identity (a subset of attributes) is needed to represent it when establishing a communication with another entity [Clauβ, 05].

Many IDM systems support the use of entity *pseudonyms* in order to group different identity attributes and provide a degree of anonymity [Clauβ, 05]. A pseudonym can represent an entity or a set of entities (i.e. a company). An example of a pseudonym is a public key certificate, which uses a digital signature to bind together a public key with a series of identity attributes (name address, etc.). A digital signature is a method for authenticating digital information analogous implemented using techniques from the field of public-key cryptography. In addition to pseudonyms, using stronger mechanisms such as Hardware Security Tokens or Biometric features users can identified. In the former, a small hardware device generates a token that when combined with a user's Personal Identification Numbers (PIN's) results in an exclusive, one-time-use pass-code [AWC, 07]. This code is used to positively identify or authenticate the user [AWC, 07]. When a biometric mechanism is used, a user registers with a system one or more of his/her physical and behavioral characteristics, which are processed by a

numerical algorithm to create a digital representation of the obtained biometric [Jain, 00]. Examples of these characteristics include fingerprints, eye retinas, facial & signature patterns, brain wave fingerprinting, etc. Typical identification methods can be classified as follows [GUIDE, 05]:

- Attribute (Knowledge) Based: Identifiers (not necessarily unique) used by an entity to authenticate itself in a particular context, e.g. Driver's license, social security number, home address and low cost RFID (Radio Frequency Identification) tags.
- Credential Based / Soft Crypto Tokens: Username/password or certificate/PIN pairs, Security Assertion Markup Language (SAML) assertions, Subscriber Identity Module (SIM) card.
- Biometric Based: Verification of a user's physical or behavioral characteristics.
- Token Based: A hardware token containing any of the previous identity data.

In a particular context, an IDM could require one or more of the identification methods in order to establish the validity of a claimed identity. As such, each method can represent a level of *trust* for the authenticity of the user's identity. Table 3-1 (adopted from [IDABC, 04]) shows the different levels of identity trust agreed to by the European governments. Using these levels, we can designate, for example, a value range to each of the levels in order to have a numerical representation of the assurance level.

A hypothetical numerical representation of the assurance levels is shown in Table 3-2. For example, level 1(minimal assurance) can take values raging from 0 to 25.

In this case, zero (0) means no authentication/identification is used or claimed. Levels 2, 3 and 4 take range values from 25 to 50, 50 to 75 and 75 to 100, respectively.

Table 3-1 Assurance Levels vs. Identification Method Agreed by the European Governments

ASSURANCE LEVEL	*IDENTIFICATION METHOD*
LEVEL 1: MINIMAL ASSURANCE	KNOWLEDGE BASED, PIN, PASSWORD
LEVEL 2: LOW ASSURANCE	ONE-TIME PASSWORD[5]
LEVEL 3: SUBSTANTIAL ASSURANCE	SOFT CRYPTO TOKEN
LEVEL 4: HIGH ASSURANCE	HARD CRYPTO TOKEN, BIOMETRIC

Using a numerical representation for the identity level, a security system could reason about permissions and deny or grant access to a resource. For example, a user that has used a biometric identification could see all the content of a classified file, while, another user who has used a one-time password could simply see an abstract of the same file.

Table 3-2 Samples of Identity Assurance Level Range Values

ASSURANCE LEVEL	*RANGE VALUES*
LEVEL 1	[0, 25)
LEVEL 2	[25, 50)
LEVEL 3	[50, 75)
LEVEL 4	[75, 100)

3.2.4 Battery Power Level

In the pervasive ad-hoc architecture, mobile and battery-powered devices (e.g. PDAs, sensors, laptops, phones, etc.) not only access and use the services of a world wide digital infrastructure, but also are considered a great source of data [Krintz, 04]. However, due to the nature of these devices, energy becomes a limiting resource; as such, past research has mainly been centered in the reduction of energy consumption at

[5] One-time passwords serve the purpose of authentication by providing a unique identification of a user, while eliminating the threat of password replays.

the hardware [Hillman, 05], operating system [Cai, 05], application/compiler [Krintz, 04] and network protocol levels [Chatzigiannakis, 04]. Nonetheless, this type of research is out of the scope of this work. Instead, we simply need to provide a numerical representation for the power level of a device. Thus, we define the power level of a device as follows:

Definition 2 Power Level

Power level is defined as the amount of energy output or battery life available to a device in order to keep operating. This availability for example can be represented as a percentage number between 0% and 100%, where 0% means no power is available and 100% means the device is either fully powered or is operating from a constant power supply (e.g. power grid). See Table 3-3

Table 3-3 Power Level vs. Percentage of Energy Output

POWER LEVEL	*% OF ENERGY OUTPUT AVAILABLE*
NO POWER	0
…	…
HALF BATTERY LIFE	50
…	…
FULLY POWERED / CONSTANT POWER SUPPLY	100

3.2.5 Connection Mode

Due to the nature of the pervasive ad-hoc environment, it is not always possible to establish and maintain a suitable connectivity between entities. This is may occur when 1) there is unsuitable channel conditions (e.g. due to noise or jamming), 2) there are network restrictions for low probability of detection, interception, and exploitation

(LPD/LPI/LPE)[6], 3) there is extremely high mobility amongst some entities and, finally 4) there are not enough power resources [Sterbenz, 02]. As a result, we could expect an entity to be operating in one of the following connection modes [Pitoura, 93]:

- Full connection mode*:* In the fully connected mode, an entity is continually connected to another entity (e.g. to a Mobile Support Station) without disruptions and noticeable degradation.
- Weak connection mode: In this mode, an entity is connected to the rest of the network through low bandwidth [Chan, 03]. This mode may occur not only when an entity is at the edge of a coverage area but also when the communication occurs over a channel that is noisy (i.e. due to bad weather conditions), jammed, or has eavesdroppers.
- Episodic Connection mode: This is a special case of the weak connection mode where a node is connected to another node with intermittent bandwidth. That is, there is seldom a complete network path that can route traffic from one entity to another entity.
- Disconnected mode: In this mode, an entity is not communicating with others due to 1) an expected failure, 2) it decides to act autonomously or 3) it tries to conserve resources. In many occasions, an entity can decide to use a trusted proxy to handle its communications while is disconnected.

A numerical representation for the connection mode can express in terms of the transmission bandwidth. Thus, we define the connection rate as follows:

[6] In some military ad-hoc networks there is the need to deny the enemy the ability to detect and exploit radio energy using techniques such as covert waveforms, directional transmissions, and reduced transmission power [Sterbenz, 02].

Definition 3 Connection Rate

The *Connection rate* of an entity can be defined as the proportion of the transmission bandwidth used during a certain period of time. For example, mathematically it can be defined as follows:

$$CR(t) = 100\left(\frac{XB(t)}{TB}\right)$$

where *TB* is the total available transmission bandwidth and *XB* the current bandwidth used at time t .

Table 3-4 describes an example of a possible mapping between the connection modes and the connection rate ranges.

Table 3-4 Sample Connection Modes vs. Connection Rate Ranges

CONNECTION MODE	***CONNECTION RATE***
DISCONNECTED	[0, 5)
EPISODIC	[5, 10)
WEAK	[10, 60)
FULL	[60, 100]

3.2.6 Honesty Level

An entity is said be honest when it tells the truth to the best of its knowledge and not only does not hide what it knows (about itself and others), but also abstains from unfair behavior [MWD, 07]. The outcome of past experiences can then give us the degree of honesty (or trust level) of an entity. For instance, if entity *e* believes entity *m* is honest, but *e* later learns that *m* was dishonest with entity *k*, then in this case, the honesty level of *m* is 50% (according to *e*), as there have been two experiences; one in which *m* was honest (with *e*) and one it which he was dishonest (with *k*).

An example of a simple measure of trust (or honesty) level with respect to an entity *e* can be simply defined as follows: Given N as the total number of events (experiences) about *e*, and P the number of positive experiences with respect to *e*, then the trust level *TL of e* denoted as TL is defined as the ratio

$$TL = 100\left(\frac{P}{N}\right).$$

The equation above captures the notion that trust is built over time by capturing the experiences with an entity; however it does not provide ways to measure a trust value given the absence of event data. Moreover, if an entity *x* gives an opinion[7] about another entity (e.g. entity *y)* to *e*, the model above does not allow entity e to asses the reliability of such an opinion. In order to provide a way to address this issue, the methodologies described in [Patel, 05] and [Teacy, 06] can be to provide an entity with means for not only measuring the trust of an entity, but also evaluating the reliability of opinions given by other entities.

3.2.7 Channel Level

The security of communication channels may be characterized by the use of encryption, which is usually defined in terms of the encryption algorithm, and the encryption-key length [Wedde, 04]. This communication may also demand the authentication method as well as any process for key exchange. In order to categorize the security of a channel, we need to take into account not only the type of applications being used but also the requirements of the requesting corporation. For instance, communication between two entities can be done in a secured or unsecured manner.

[7] Opinion is referred to the trust value *x* has for *t*

Secure communication can be either symmetric, public-key or key-exchange based. Symmetric and public-key communication protocols can be further classified by not only their key length, but also the encryption algorithm used. Table 3-5 (adopted from [Wedde, 04]) describes an example of an encryption/encoding algorithm classification strength-value mapping. For example the strength value of an unsecured plain text communication could be 1, whereas a communication done through a private channel using a 2048 bit encryption key could be considered fully secure and could have strength value of 100 (with current state of the art non quantum computing machines, it would take about 30 billion years to decode a 2048 bit encrypted data [RSA, 13]. It should be noted that encryption algorithms based on integer prime factorization have a computational complexity in both NP and co-NP - [Aaronson, 13], [Weisstein, 13] -). This classification gives rise to the following definition.

Definition 5 Channel Level

The channel level of an communication method can be defined as the value (or range[8]) that designates its strength given a tuple of the form ***<m, t, k, a>*** where *m* is the communication mode (e.g. secured), *t* is communication type or protocol (e.g. symmetric), k is the authentication-encryption key length (e.g. 128bit) and *a* is the encryption algorithm (e.g. RSA).

3.2.8 Temperature

Temperature can be defined as degree of hotness or coldness measured on a definite scale [MWD, 10]. In a pervasive ad-hoc environment such as the one in [Gaye, 03], context information such as temperature, noise level, proximity, pollution level,

[8] A range is defined so new algorithms can be included in the classification

etc. is used create electronic music in real time.

Table 3-5 Suggested Channel Level Classifications

Mode	TYPE	KEY LENGTH	ALGORITHM[9]	STRENGTH VALUE
UNSECURED			NO ENCRYPTION	[1, 5)
			COMPRESSION	[5, 10)
	ENCODING	N/A	MACBINARY	[10, 15)
			APPLEDOUBLE	[15, 20)
			BINHEX	[25, 30)
			UUENCODE	[30, 35)
			MIME / BASE64	[35, 50)
SECURED	SYMMETRIC	56 BIT	DES	[50, 60)
		128 BIT	IDEA	[60, 65)
			3DES	[65, 70)
			BLOWFISH	[70, 75)
			AES	[75, 80)
		256 BIT	BLOWFISH256	[80, 84)
			GOST	[84, 86)
			RIJNDAEL256	[86, 90)
		448 BIT	BLOWFISH448	[90, 100)
	PUBLIC KEY	1024 BIT	RSA1024	[70, 80)
			ELGAMAL1024	[70, 80)
		1572-1576 BIT	ELGAMAL1572	[80, 85)
			RSA1576	[86, 90)
		2048 BIT	RSA2048	[90, 100)
			ELGAMAL2048	[90, 100)
	KEY EXCHANGE[10]		DIFFIE-HELLMAN	[55, 60)
			ENCRYPTED KEY EXCHANGE	[60, 65)

[9] An overview of cryptographic protocols and algorithms are given in [Schneier, 96] and [Goldreich, 03]
[10] Key Exchange is used in RSA and ELGamal encryption algorithms.

There are three main ways to measure to temperature by means of different scales, which include Kelvin, Celsius and Fahrenheit. The formula for conversion between any two of them is shown in Table 3-6 [BPM, 06].

Table 3-6 Temperature Conversions

TO FIND	*FROM*	*FORMULA*
Fahrenheit	Celsius	°F = (°C × 1.8) + 32
Celsius	Fahrenheit	°C = (°F − 32) /1.8
Kelvin	Celsius	K = °C + 273.15

3.3 Risk Factor Time Series

Following the definition presented in [Hamilton, 94], a risk factor time series is a collection of observations - made by an entity - indexed by the time (and/or date) of each observation. An entity collects the risk factor data beginning at a particular time (e.g. t = 1) and ending at another (e.g. t = T). Formally, a time series is represented in vector form a follows [Hamilton, 94]:

$$\left(y_1, y_2, \ldots, y_i, \ldots y_T\right)^T$$

where y_1 represents the observation made at time 1, y_i is the observation made at i and y_T is the observation made at time T. Entities can represent earlier (before time 1) or later observations (after time T) as $(y_0, y_{-1}, y_{-2}\ldots)^T$ or $(y_{T+1}, y_{T+2}\ \ldots)^T$ respectively [Hamilton, 94]. Adopting the notation from [Hamilton, 94], the observed sample (y_1, $y_{2,}\ldots,y_T$) could the be viewed as finite segment of doubly infinite sequence denoted as $\{y_t\}_{t=-\infty}^{\infty}$:

$$\{y_t\}_{t=-\infty}^{\infty} = \{..., y_{-1}, y_0, y_1, y_2, ... y_T, y_{T+1}, y_{T+2}, ...\} \quad (3.1)$$

Typically, a time series $\{y_t\}_{t=-\infty}^{\infty}$ is identified by describing the t[th] element. For instance, a time series in which each element is equal to a constant *c*, regardless of the date of the observation *t is* written *as* [Hamilton, 94]:

$$y_t = c$$

An example of a constant time series could be the location of a device in a fixed position or for instance the power level of a fully charged device connected to an unlimited power supply.

As is done in basic arithmetic, operators over time series are defined to transform one or more time series in to a new time series. In the next section we review old and define new time series operators useful in the expansion and definitions of context based security policies that include context data history.

3.3.1 Time Series Operators

According to [Hamilton, 94], a time series *operator* transforms one time series into a new time series, by accepting as input a sequence such as $\{x_t\}_{t=-\infty}^{\infty}$ and yielding as output a new sequence $\{y_t\}_{t=-\infty}^{\infty}$. Each element of $\{y_t\}_{t=-\infty}^{\infty}$ is described in terms of the corresponding elements of $\{x_t\}_{t=-\infty}^{\infty}$ [Hamilton, 94].

Arithmetic Operators

Based on [Hamilton, 94], for a risk factor time series the *multiplication operator* can be represented as

$$y_t = \beta x_t \qquad (3.2)$$

where each value of y at a time t is the result of multiplying a scalar β by the corresponding value of x at time t.

Another risk factor time series operator is the *addition operator* [Hamilton, 94]:

$$y_t = x_t + w_t \qquad (3.3)$$

where the value of *y* at time *t* is the sum of the values that *x* and *w* take on for that time.

Since (3.2) and (3.3) obey the standard rules of algebra, an operation such that, given two scalars α and β and two time series $\{x_t\}_{t=-\infty}^{\infty}$ and $\{w_t\}_{t=-\infty}^{\infty}$, a new time series $\{y_t\}_{t=-\infty}^{\infty}$ can be defined as follows [Hamilton, 94]:

$$y_t = \alpha x_t + \beta w_t. \qquad (3.4)$$

Arithmetic operators will be useful when for instance we need to combine two or more context data time series of the same type (e.g. location) supplied by different *Context Providers (CP).*

Missing Operator

The *missing operator*, denoted as *M*, is a Boolean operator that tests if the value at a time t for a time series $\{y_t\}_{t=-\infty}^{\infty}$ has not been defined. That is, this operator checks if an observation was made at time t. The M operator is denoted as follows:

$$M(y_t) = \begin{cases} 1 & if\ y_t\ is\ undefined \\ 0 & otherwise \end{cases}$$

Percentage Missing Operator

For a context risk factor series is always important to know the degree or percentage of missing observations. For instance, we can make part of a security policy,

that the percentage of missing observations for a specific risk factor (e.g. location) does not exceed a certain threshold. As such, the *percentage missing operator* (denoted as *PeM*) for a given time series $\{y_t\}_{t=-\infty}^{\infty}$ and two time points *i* and *j* where $i < j$, $PeM_{ii}^{j} y_t$ is defined as follows:

$$PeM_{i}^{j} y_t = \frac{\sum_{k=i}^{j} M(y_k)}{(j-i+1)} \qquad (3.5)$$

The *PeM* amount can give us an indicator of the quality of the data supplied by a context provider.

Fill Operators

The *backward fill operator*, denoted as *BF*, is a unary operator that for a given time series $\{y_t\}_{t=-\infty}^{\infty}$, two time points t^i and t where $t^i < t$ (that is, there are *i* time steps between t^i and *t*) and $M(y_t) = 0$ is defined as follows:

$$BF^{i} y_t = \{y_{t-k} = y_t \quad \forall k = [1..i] \; and \; M(y_t) = 0\} \qquad (3.6)$$

For example, a time series $\{x_t\}_{t=1}^{5}$ representing the power level of a host in the time interval [1,5] can be defined as $\{x_t\}_{t=1}^{5} = (,,93,94,95)$. As seen, the first two observations (t=1, t=2) are missing. By applying the *backward fill operator* for times 1 and 3 (denoted as $BF^2 x_3$), $\{x_t\}_{t=1}^{5}$ becomes $\{x_t\}_{t=1}^{5} = (93,93,93,94,95)$.

Similarly, the *forward fill operator*, denoted as *FG*, is a unary operator that for a given time series $\{y_t\}_{t=-\infty}^{\infty}$, two time points t and t^i where $t > t^i$ and $M(y_t) = 0$ is defined as follows:

$$FF^{j} y_t = \{ y_{t+k} = y_t \quad \forall k = [1..j] \; and \; M(y_t) = 0 \} \qquad (3.7)$$

For example, a time series $\{p_t\}_{t=1}^{5}$ that presents the proximity of two objects is defined in time interval [1,5] $\{p_t\}_{t=1}^{5} = (67,68,69,,)$. As seen, the last two observations (t=4, t=5) are missing. By applying the *forward fill operator* for time 4 and 5 (denoted as $FF^2 p_3$) then $\{p_t\}_{t=1}^{5}$ becomes $\{p_t\}_{t=1}^{5} = (67,68.69,69,69)$.

These operators are useful to fill missing data in time series supplied by different context providers; thus, all elements in a particular time series are never missing.

Swap Missing Operator

Finally, the *swap missing operator* is defined as follows: given two time series $\{x_t\}_{t=-\infty}^{\infty}$ and $\{w_t\}_{t=-\infty}^{\infty}$ a time interval $(-\infty < t_i \le t_j < \infty)$ a new time series $\{y_t\}_{t=t_i}^{t_j}$ can be defined as follows:

$$\{y_t\}_{t=t_i}^{t_j} = \begin{Bmatrix} x_t \; if \; M(x_t) = 0 \\ w_t \; otherwise \end{Bmatrix} \qquad (3.8)$$

The swap-missing operator is useful when combining the content of two *context risk factor* time series and thereby reducing the number of missing observations in the resulting time series.

3.3.2 Time Unit for Context Risk Factor Series

In the pervasive ad-hoc environment not only the collection of time series information depends on the capacity (storage and power consumption) of the device performing the collection, but also on the type of *context risk factor* being collected as

well as the degree of accuracy required for context data. In this framework, each security policy specifies the time unit (e.g. 1 minute) that it expects the time series data to be supplied by the context providers. Whenever information cannot be supplied at the specified policy rate, this framework will use either the forward or backward fill operators in order to make all the *context risk factor* series of same length.

3.4 Context Risk Factor Based Security Policies

Regardless of the security schemes (DAC, RBAC, etc.) used to describe access and authorization controls, they can be extended in order to encode all the security policies this framework allows. Such policies may contain not only be simple comparisons of trust, risk and cost metrics but also expressions to test *context risk factor* time series data. Security policies can be expanded by expressions from the grammar described in Table 3-7. This BNF grammar is based on the grammars found in [Sun, 06] and [MATIS, 06].

Table 3-7 BNF for Security Policies

```
<syntax>   ::=  { <expression> }

<expression> ::=
        <numeric_expression>
|       <testing_expression>
|       <logical_expression>
|       <literal_expression>
|       <riskfactor_expression>
|       <series_operators>
|       <trust_operators>
|       "(" <expression> ")"
|       "[" <expression> "]"
|        <expression> ( ("." <expression> )  |  ( "," <expression> ) )

<numeric_expression> :=  <expression> ( "+" | "-" | "*" | "/" ) <expression>
```

```
<testing_expression> :=
        <expression> ( ">" | "<" | ">=" | "<=" | "==" | "!=" | “in”) <expression>

<logical_expression> :=
        "!" <expression>
|       <expression> ( "&&" | "||" ) <expression>
|       "true"
|       "false"

<riskfactor_expression> := (<singlevalue_factor> | <timeseries_factor>)

<singlevalue_factor> := ( <proximity_function> || <singleparam_function>)

<proximity_function> := “proximity” “(“ <identifier> “,” <identifier> ”)”

<singleparam_function>  :=
        (“identity” | “velocity” | “connection_rate“ | “power_level” | “trust_level” |
        “channel_level”)  “(“<identifier> ”)”

<timeseries_factor> := (<proximity_series> | <singleparam_series>)

<proximity_series> := “proximity_series” “(“ <identifier> “,” <identifier> ”)”

<singleparam_series> :=
        (“identity_series” | “velocity_series” | “connection_rate_series“ |
        “power_level_series” | “trust_level_series” | “channel_level_series”)
        “(“<identifier> ”)”

<trust_operators> := “trust” ”(“ <identifier> ”,” <identifier> “,” <identifier> ”)”

<series_operators> :=
        (<missing_operator> | <pem_operator>  | <fill_operators> | <swap_operator> |
         <classify_operator> | <volatily_operator>)

<missing_operator> := “missing” ”(“ <missingop_params>“)”

<missingop_params> :=
        ( <timeseries_factor> | <timeseries_factor> “,”<integer_literal>)

<classify_operator > := “classify” ”(“ <timeseries_factor> “)”

<volatily_operator> := “volatily” ”(“ <timeseries_factor>, <numeric_expression>“)”

<pem_operator> := “percentage_missing” ”(“ <pemop_params>“)”
```

```
<pemop_params> :=
        ( <timeseries_factor> |
          <timeseries_factor> “,”<integer_literal> “,”<integer_literal>)

<fill_operators> := (“forward_fill” | “backward_fill”) ”(“ <fillop_params>“)”

<fillop_params> :=
        ( <timeseries_factor> |
          <timeseries_factor> “,”<integer_literal> “,” <integer_literal> )

<swap_operator> := “swap_missing” ”(“ <swapop_params>“)”

<swapop_params> :=
        ( <timeseries_factor> “,” <timeseries_factor> |
          <timeseries_factor> “,”<timeseries_factor> “,” <integer_literal> “,”
           <integer_literal> )

<literal_expression> := ([ “-“ ]) (<integer_literal> | <float_literal>)

<integer_literal> :=  "1..9"  { "0..9" }

<float_literal>  :=
        ( <decimal_digits> "." [ <decimal_digits> ] [ <epart> ]  [ < ftype _suffix > ]  )
|       (  "." <decimal_digits>  [ <epart> ]  [ <ftype _suffix> ]  )
|       ( <decimal_digits>  [ <epart> ]  [ <ftype_suffix> ]  )

<decimal_digits> := "0..9"  { "0..9" }

<epart> :=  "e" [ "+" | "-" ] <decimal_digits>

<ftype_suffix> :=  "f" | "d"

<identifier> := "a..z,_,A..Z" {"a..z,A..Z,_,0..9,unicode character over 00C0"}
```

- In <*testing_expression*> production rule, the “***in***” operator is introduced to define an inclusion operator. That is, to test is a single value x is within a range or included in a time series expression.
- The <*riskfactor_expression*> production rule defines the single value *context risk factor* expressions used in the grammar. It can derive to

expressions <*singlevalue_factor*> and <*timeseries_factor*>.

- The <*singlevalue_factor*> rule refers to a single value for a context risk factor. For example, the power level of an entity *e* at a given time *t*. This rule can derive either a <*proximity_function*> or a <*singleparam_function*>.
- The <*proximity_function*> rule refers to the "***proximity***" function, which takes two parameters (entity identifiers) and returns the proximity between those entities. In addition, the <*singleparam_function*> production rules derives terminal expressions for the *context risk factor* functions "***identity***", "***velocity***" "***connection_rate***", "***power_level***", "***trust_level***", and "***channel_level***". These functions take one parameter – an entity identifier - and return the entity's identity certainty, connection rate, the power level, trust level and connection/channel level, respectively.
- Unlike the <*riskfactor_expression*> rule, the <*timeseries_factor*> production defines time series *context risk factor* expressions in this grammar. It can derive expressions <*proximity_series*> and <*singleparam_series*>. When implemented these two last expressions will return time series (available) data for an entity's context risk factor.
- The <*proximity_series*> production refers to the function ***"proximity_series"***, which takes two parameters (entity identifiers) and returns the available time series proximity data between those entities. The <*singleparam_series*> rule derives terminal single-parameter time series functions for the following *context risk factors*: identity, connection rate, power level, trusts level and channel level. These functions are

"***identity_series***", "***velocity_series***", "***connection_rate_series***", "***power_level_series***", "***trust_level_series***" and "***channel_level_series***".

- The *<series_operator>* rule defines the grammar for the time series operators previously defined. The productions derived from this rule are: *<missing_operator>,* <pem_operator>, *<fill_operators>* and *<swap_operator>*.
- The <missing_operator> production rule defines the overloaded function "***missing***". When this function takes only a time series parameter, it returns a new time series with zeros or ones representing all missing or present observations in the time series. When it takes two parameters, a time series *ts* and an integer *t*, it tests if the value at a time t for time series *ts* has been defined.
- The <pem_operator> production rule defines the overloaded function "***percentage_missing***". When this function takes only a time series parameter, it returns the ratio of missing observations in the time series. When it takes three parameters, a time series *ts*, an integer *i*, and an integer *j*, it returns the ratio of missing observations in the time series between time *i* and *j*.
- The *<fill_operators>* rule defines the overloaded functions "***forward_fill***" and "***backward_fill***". When they take one time series as a parameter they return a forward or backward filled time series for all missing gaps in the original series. When they take two parameters, a time series *ts* and an integer *t*, it returns a new time series that has been either forward or

backward filled from or at point *t*.

- The <*swap_operator*> rule defines the overloaded "***swap_missing***" function. When it takes two times series *X* and *Y* as parameters, it returns a modified *X* time series whose missing points have been replaced with the values from *Y*. Finally when the function takes four parameters, two time series *X* and *Y* and two integers *i and j*, it returns a modified time series *X* whose missing points from *i to j* have been replaced with values from time series *Y*.
- The <*classify_operator*> production rule defines the overloaded function "***classify***". When it takes a times series *X* parameter, it returns a tuple that contains the classification class and the probability of the estimation.
- The <*volatility_operator*> production rule defines the overloaded function "***volatility***". When it takes a time series *X* and a numeric expression as parameters, it returns the volatility of the returns X according to the numeric expression
- The <trust_operators> production rule defines the overloaded function "***trust***". When it takes two context owners identifiers A and B and context factor C identifier as parameters, it returns the trust between that A has for B under context factor C.

Chapter 4. RISK MODEL

As presented in chapter two, most of the current security systems do not take into account the variability of the context data when either calculating trust measures or evaluating a security policy. In this chapter, we describe the model used to determine the *risk* of an access control operation (policy) measured according to the changes of risk factors (known henceforth as context variables).

We start by describing not only a possible scenario in which this research is situated but also the key challenges that arise from it. Partial results of this research has been published in [Sanchez, 07] and [Sanchez, 08]

4.1 Scenario

In 1991 the US Army established the concept of Land Warrior [FAS, 07] in order to integrate small arms with high-tech equipment with the aim of increasing the lethality, survivability and command and control of soldiers in the battlefield. One of the components of this system is the Integrated Helmet Assembly Subsystem (IHAS) which is a computer and sensor display mechanism mounted on the soldier's helmet to provide him with not only views of computer-generated graphical data, digital maps, intelligence information and troop locations but also the capability of transmitting information back to commanders.

As soldiers move about the field they get battle information from their battalion commander (i.e. a soldier with the rank of Mayor). In order to minimize leaks, only the company commander (i.e. a lieutenant) can receive enemy intelligence after the company commander successfully identifies himself and provides an encryption key (in

this case, data accessibility is restricted based on the identity and encryption type). Moreover, in order to receive such intelligence data, the company commander has to be within certain proximity (e.g. between 60 and 80 meters) of the battalion commander (again, data accessibility is restricted according to location). To make sure that all the data is delivered the IHAS system must have a minimum of power (e.g. 60%) available (in this case, data transfer is based on availability of power resource). Corporal soldiers receive updated battle maps according to their proximity of the company commander and how fast they move away from him (in this case, data transfer is restricted according to velocity). Finally, in order to receive any data, the connection rate between any two soldiers needs to be above a specific threshold in order to guarantee that the communications are not being jammed (again data transfer is limited to a specific connection mode).

4.2 Challenges

According to the scenario described in the previous section, following are some of the major security challenges that remain to be addressed by an adopted security model:

- Since a soldier receives data according to his position and rank, information may be compromised if that soldier falls into the enemy's hands. In such a case, the enemy can make invalid context claims (e.g. position and identity) to get data. Therefore, it becomes necessary to adopt a model to correlate and validate the information being sent by soldiers.
- As a soldier moves in the field it is not always possible for his commander to have permanent real-time communication with him due to various reasons,

such as power constraints, jamming and atmospheric conditions. For example, in this situation, the soldier can only report his location periodically. As his position changes with time, uncertainty is introduced when evaluating and deciding what data he should have access to at a particular time. The question that arises is by the time the data reaches the soldier, whether such information is still relevant, as he may have moved to a new location. In this case, we would like to have a security model that uses the soldier's reported location history to not only predict future locations, but more importantly, to measure the degree of change (volatility) of the context variable (location) in order to arrive at a more accurate security decision. For example, the battalion commander may decide to deny access to battle data if the soldier is likely to fall out of [safety] range.

- Due to the dynamicity of the context variables in the battlefield, the security model should use this context data to produce a risk value when it evaluates a security request.
- Since a soldier's context variables may change in an uncertain way, any context based data request should have a time window where it is valid (evaluation horizon) before it is revaluated. For instance, if a soldier accessing information about friendly units in his vicinity becomes a prisoner and if the system security implementation does not enforce a time window for data-request reevaluation, the enemy could either simply continue accessing the information or at least do some signal intelligence.
- War and battles require preparation and training through the use of war

games. It then becomes necessary to use a simulation framework where context variables can be modified in order to do a what-if analysis measuring the risk (degree of trust) of different data requests under diverse situations. Moreover, a what-if analysis could also be used not only to predict future scenarios but also to reproduce past situations.

The challenges described above require the security model to be able to represent the behavior of context variable data. Furthermore, the model should also take into account the time horizon under which a data request is valid. In addition, it should make possible the detection of correlations amongst context variables. Finally, the model should provide ways to calculate risk measures and provide a framework to simulate new or replicate old situations. In this chapter we first present a way to model context variables. We then employ the model to predict future values of a context variable and produce a risk measure for evaluating a security policy based on that context variable.

4.3 Modeling Context Variables

In a pervasive mobile environment, context variables of a user may change in a non- predetermined way. For instance, a student moving about a college campus, not only goes to different classrooms to attend classes based on his/her regular schedule, but also may stop at or go to unplanned places (cafeteria, library, etc.). That is, context variables follow a stochastic process since their values may change over time in an uncertain way. Using a stochastic process model gives us a measuring tool to characterize the future change in a process' value.

One stochastic process that can be employed as a measuring tool is the

Logarithmic Random Walk Model (section 2.3.2). In this random walk, forecasts for each of the context variables' future value changes - using only its past variations – can be constructed. In forecasting changes, we attempt to characterize the risk (impact) that such changes have in the evaluation of a security policy. We choose then this model because

1) It gives as a uniform mechanism to model a wide range of context type data.
2) It describes the evolution of the context variables over time.
3) It gives us a distribution for the future value changes of a context variable.

However, before defining the random walk model we need to introduce the concepts of a context variable time series and variable return.

4.3.1 Context Variable Time Series

As mentioned in Chapter 3, risk factor or context variable time series is a collection of observations indexed by the time of each observation. An entity collects the context variable data beginning at a particular time (e.g. t=1) and ending at another (e.g. t=N). Formally, a context variable time series is represented in a vector form as follows [Hamilton, 94] and [RMG, 96]:

$$\{P_t\}_{t=1}^{N} = (P_1, P_2, \ldots, P_i, \ldots P_N)$$

where P_1 represents the observation made at time 1 for a continuous context variable P, P_i is the observation made at time i and P_N is the observation made at time N. Finally, a time series is identified by describing the t^{th} element. For instance, a time series in which each element is equal to a constant c (regardless of the observation time) is written as [Hamilton, 94]:

$$P_t = c$$

4.3.2 One-time (single period) value return horizon

The change in the value of a variable can be expressed in a variety of forms, such as, absolute value change, relative value change, and log value change. When a value change is defined relative to some initial value, it is known as a *return* [RMG, 96]. That is, the changes over time of a variable's value can be measured and modeled in terms of continuously compounded returns (log value changes). Table 4-1 shows the definition of Absolute, Relative and Log value change for a context variable *P* between time *t* and *t-1* (denoted as P_t and P_{t-1}). Table 4-2 shows an example of a 10 observation time series (1 minute time unit) of proximity values (in meters) between two entities and their absolute (**Abs**), relative (**Rel**) and logarithmic (**Log)** changes [Korpipää, 03].

Table 4-1 Definitions of Absolute, Relative and Log Changes of a Variable

Absolute value change	$D_t = P_t - P_{t-1}$
Relative value change	$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$
Log value change (return)	$r_t = \ln(1 + R_t) = \ln\left(\frac{P_t}{P_{t-1}}\right)$ $r_t = (p_t - p_{t-1})$ where $p_t = \ln(P_t)$

4.4 Random Walk Model

A random walk is a formalization of the idea of taking successive steps, each in a random direction. It may be thought of as a model for an individual walking on a straight line who, at each point of time, takes one step either to the right or to the left with different probabilities [Weisstein, 10].

Table 4-2 Examples of Absolute, Relative and Log Changes of a Proximity Variable Time Series Between Two Entities

t	Time	Value	Abs	Rel	Log
1	8:00	66.68			
2	8:01	67.03	0.350	0.0052	0.0052
3	8:02	67.66	0.635	0.0094	0.0094
4	8:03	67.44	-0.219	-0.0032	-0.0032
5	8:04	67.54	0.096	0.0014	0.0014
6	8:05	67.60	0.059	0.0008	0.0008
7	8:06	67.48	-0.119	-0.0017	-0.0017
8	8:07	67.42	-0.063	-0.0009	-0.0009
9	8:08	67.73	0.310	0.0045	0.0045
10	8:09	67.65	-0.078	-0.0011	-0.0011

Random walk models have been applied in several fields. For instance, in economics, a random walk is used to model shares prices and other factors and in wireless networking, they are used to model node movement.

Formally, a *single value random walk* model for a context variable can be stated as follows:

$$P_t = \mu + P_{t-1} + \sigma_t \varepsilon_t,\ \varepsilon_t \sim IIDN(0,1) \qquad (4.1)$$

or

$$P_t - P_{t-1} = \mu + \sigma_t \varepsilon_t,\ \varepsilon_t \sim IIDN(0,1) \qquad (4.2)$$

where *IID* stands for "identically and independently distributed", and N(0,1) stands for the normal distribution with mean 0 and variance 1. That is, at any point in time the current value P_t depends on one fixed parameter μ (mean), one time based parameter σ_t (standard deviation), the last period's value P_{t-1}, and a normally distributed random variable ε_t. The assumption that context values are normally distributed is helpful because 1) we only need the mean and variance to describe the distribution, 2) the sum of multiple normal context variables is also normally distributed, and 3) normality is the central assumption of the mathematical theory of errors [Weisstein, 10a].

Since *returns* not only have more attractive statistical properties than values, but also are often preferred to absolute value changes because the latter do not measure changes in terms of the given values [RMG, 96] it is better to model the **log value** p_t (Table 4-1) as a random walk with normally distributed changes, that is:

$$p_t = p_{t-1} + \mu + \sigma_t \varepsilon_t, \;\; \varepsilon_t \sim IIDN(0,1) \quad (4.3)$$

Therefore, since we are modeling log changes, the expression for values is simply obtained by taking the inverse of the logarithm (e^x), that is

$$P_t = P_{t-1} e^{\mu + \sigma_t \varepsilon_t}, \quad \varepsilon_t \sim IIDN(0,1) \qquad (4.4)$$

In [RMG, 96] three core assumptions are made. First of all, returns in different periods are not auto correlated. Secondly, the variance of the returns (*volatility*) scales with time (it remains constant for different time horizons). Finally, the random walk model is relaxed by assuming that log values have a mean μ set to zero (in [RMG, 99], it is shown that for short horizon periods, the volatility is much larger than the expected return; thus, the forecast of the future return distribution is dominated by the volatility estimate. In other words, when dealing with short horizons, using a zero expected return assumption is as good as any other mean estimate). Therefore, the model can be represented as:

$$p_t = p_{t-1} + \sigma_t \varepsilon_t, \; \varepsilon_t \sim IIDN(0,1) \qquad (4.5)$$

in terms of returns

$$r_t = \sigma_t \varepsilon_t, \; \varepsilon_t \sim IIDN(0,1) \qquad (4.6)$$

The standard deviation (*volatility*) of the value changes (returns) - σ_t – which we assume varies with time can be estimated using an Exponential Moving Average (EMA) ([Kenney, 62], [Chou, 75]) of past observations. That is, at any given time *t*, the

variance can be computed as follows:

$$\sigma_t^2 = \frac{\sum_{i=1}^{t-1} \alpha^{i-1} \left(r_{t-i}\right)^2}{\sum_{i=1}^{t-1} \alpha^{i-1}} \qquad (4.7)$$

where the parameter α *(0 < α <=1)* (***smoothing factor***) determines the relative weights applied to the observations (*returns*), thus, allowing the latest observations to carry the highest weight, while still not discarding older observations entirely in the standard deviation estimate. In addition, when $K \to \infty$, and using the convergence property of the geometrics series

$$\sum_{i=1}^{T} \alpha^{i-1} \cong \frac{1}{1-\alpha} \quad 0 < \alpha < 1$$

equation (4.7) can be rewritten as follows:

$$\sigma_t^2 = (1-\alpha)\sum_{i=1}^{\infty} \alpha^{i-1} r_{t-i}^2 \quad (4.8)$$

In a recursive manner, equation (4.8) can be rewritten as:

$$\begin{aligned} \sigma_t^2 &= (1-\alpha)\sum_{i=1}^{\infty} \alpha^{i-1} r_{t-i}^2 \\ &= (1-\alpha)\left(r_t^2 + \alpha r_{t-1}^2 + \alpha^2 r_{t-2}^2 + \ldots\right) \\ &= (1-\alpha) r_t^2 + \alpha\left(1-\alpha\right)\left(r_{t-1}^2 + \alpha r_{t-2}^2 + r_{t-3}^2\right) \\ &= \alpha\sigma_{t-1}^2 + (1-\alpha) r_t^2 \end{aligned} \qquad (4.9)$$

Table 4-3 shows the calculation of the volatility for the proximity data in Table 4-2. The smoothing factor α used is 0.95. The standard deviation (square root of the variance) is equal to 0.394% (variance is 0.00155%).

Table 4-3 Estimating Standard Deviation (Volatility) for Proximity Values with α=0.95

T	Value	Return	σ^2_t
1	66.683		
2	67.033	0.00523	0.00394
3	67.668	0.00943	0.00379
4	67.449	-0.00324	0.00204
5	67.545	0.00142	0.00226
6	67.604	0.00087	0.00237
7	67.485	-0.00176	0.00258
8	67.422	-0.00093	0.00278
9	67.732	0.00459	0.00331
10	67.654	-0.00115	0.00115

Equation (4.9) represents one time-unit (e.g. 1 minute) calculation of the variance defined over the period *t-1* through *t*, where each *t* represents one time-unit. Therefore, in order to make forecasts for horizons greater than one-time unit, and taking the assumption stated previously, the variance estimate of a context variable data return for H time units is stated as follows:

$$\sigma_H^2 = H\sigma_t^2 \qquad (4.10)$$

The equation above gives a simple way to calculate the *volatility* of H time units (e.g. hour) from the 1 time unit (e.g. minute) *volatility*.

4.5 Single Variable Monte Carlo Scenario Generation Algorithm

Given a time series $\{P_t\}_{t=1}^{N}$ of N observations for a context variable P, the procedure to generate scenarios is to generate standard normal variates and use equation (4.4) to produce future values. The algorithm to simulate future values for one variable is described in Table 4-4. Table 4-5 shows 5 simulation trials for a proximity value with P_N=68 meters and σ_N = 0.394% (N=10).

Table 4-4 Scenario Generation Algorithm

Input: $\{P_t\}_{t=1}^{N}$
1. Choose the number of scenario trials *T* and smoothing factor α
2. Compute the *N-1* log value changes (i.e. returns) from $\{P_t\}_{t=1}^{N}$. The result of this calculation is a time series of returns of the form $\{r_t\}_{t=2}^{N}$
3. Using $\{r_t\}_{t=2}^{N}$ compute the variance series $\{\sigma_t^2\}_{t=2}^{N}$ using equation (4.7). Compute the *volatility* (standard deviation) at time *t=N,* that is $\sigma_N = \sqrt{\sigma_N^2}$
4. In order to simulate values for the next H time units (evaluation horizon), use equation (4.10) to compute the horizon *volatility* $\sigma_H = \sigma_N \sqrt{H}$
5. Define a time series $\{S_t\}_{t=1}^{T}$ to store the context variable simulated values.
6. For each trial *i* $(1 \le i \le T)$ 1. Generate or simulate an identically and independently distributed normal value *Z*, that is, $Z = \sim N(0,1)$ 2. Compute the i^{th} simulated value S_i using equation (4.4) [with $\mu=0$] as follows: $S_i = P_N \exp(\sigma_H Z)$ where $\exp(x) = e^x$ and P_N is the value of the context variable at time N
Output: $\{S_t\}_{t=1}^{T}$

Table 4-5 Scenario Trials For a Proximity Value, where P_N=68m, with H=1, σ_N=0.394% and T=5

Trial	Z~N(0,1)	S_i (meters)
1	0.41405	68.11102
2	-1.53471	67.59006
3	1.58510	68.42601
4	0.30701	68.08230
5	-0.37158	67.90052

This collection of future values -S_i- provides a sampling of the context variable distribution from which we can compute a risk measure.

4.5.1 Security Policies and Risk

To measure the risk of a context variable, in a context based security policy, the following is assumed:

1. Risk can be measured according to the changes of the context variables *(e.g.*

location, power resource, etc.).

2. Without loss of generality, the values of a context variable are always positive. Moreover, the [log] changes in the context variables can be modeled using as a random walk (following a normal distribution).
3. Initially, each security policy definition includes a smoothing factor α, the number of scenario trials T for the Monte Carlo method, an evaluation horizon H to determine how long the permissions are valid once they have been granted, the time series frequency F (a factor of H) required for evaluation and, finally, the number of observations in the time series N.
4. The following section (4.5.2) describes an example of a simple context based security policy with a single context variable.

4.5.2 Example of a Context Based Security Policy

In our army scenario, a simple security policy can be stated as follows: a soldier can receive the location of his unit members, provided his *proximity* to the commanding officer is no less than 70 meters. The commanding officer requires that the location of the soldier be reported every minute (e.g. Table 4-3) (F =*1minute*), and that the *volatility* (standard deviation) of the data not exceed 0.01. Every minute the policy is reevaluated ($H = 1$). The commanding officer uses a smoothing factor of 0.94 (α = 0.94) and generates 100 Monte Carlo scenarios ($T = 100$) before returning a decision.

At the moment of the policy evaluation the soldier presents the following to the commanding officer:

1. His current position, based on which his *proximity* is calculated (e.g. 68 meters).

2. His location history (*N* observations) at the time frequency specified by the policy (*F*).

4.5.3 Risk

Before making a decision, the commanding officer computes the difference between the Monte Carlo estimate and the soldier's current proximity; this difference is called an *error*. Steps 5 and 6 of the algorithm in Table 4-4 are modified as follows:

Table 4-6 Modified Steps 5 and 6 of Monte Carlo scenario generation algorithm in Table 4-4

5. Define the time series $\{S_t\}_{t=1}^{T}$ and $\{E_t\}_{t=1}^{T}$ to store the context variable simulated values and errors.
6. For each trial *i (1≤ i ≤ T)* 1. Generate or simulate an identically and independently distributed normal value Z, that is, $Z = \sim N(0,1)$ 2. Compute the ith simulated value S_i using equation (4.4) [with μ=0], as follows: $S_i=P_N\ exp(\sigma_H Z)$ where $exp(x) = e^x$ and P_N is value of the context variable at time N. *3.* $E_i = S_i - P_N$
Output: $\{S_t\}_{t=1}^{T}$ and $\{E_t\}_{t=1}^{T}$

Table 4-7 shows the five computed errors for the simulated values in Table 4-5.

Table 4-7 Error Estimation for the Simulation Trials Described in Table 4-5

Trial	**Z~N(0,1)**	S_i **(meters)**	E_i
1	0.41405	68.11102	0.11102
2	-1.53471	67.59006	-0.40994
3	1.58510	68.42601	0.42601
4	0.30701	68.08230	0.08230
5	-0.37158	67.90052	-0.09948

To measure the risk, we use the ***percentile*** function over the values of $\{E_t\}_{t=1}^{T}$ to

determine the value below which a certain percent of $\{E_t\}_{t=1}^{T}$ observations fall. That is, using the Monte Carlo algorithm, we simulate multiple hypothetical trials for future value (and errors) based on the normal random walk model and then calculate the maximum error amount (*risk*) that would be incurred in the estimation for a given evaluation horizon *H* and *confidence level (γ)*.

Formally, the p^{th} percentile of the $\{E_t\}_{t=1}^{T}$ values is defined as the magnitude that exceeds *p* percent of the values, that is:

$$risk_P = \left| percentile\left(\{E_t\}_{t=1}^{T}, (1-\gamma)\right) \right| \quad (4.11)$$

Mathematically, the p^{th} percentile (denoted by α) of a continuous probability distribution, is given by the following formula:

$$p = \int_{-\infty}^{\alpha} f(r)\,dx$$

where *f(r)* represents the Probability Density Function (PDF).

When speaking of percentiles, we often refer to the percentiles of a normal distribution [Kim, 01]. Moreover, because of the assumption that context values are *normally distributed*, then the p^{th} percentile becomes

$$p = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} e^{-x^2/2}\,dx$$

Figure 4-1 illustrates the positions of some selected percentiles of the standard normal distribution.

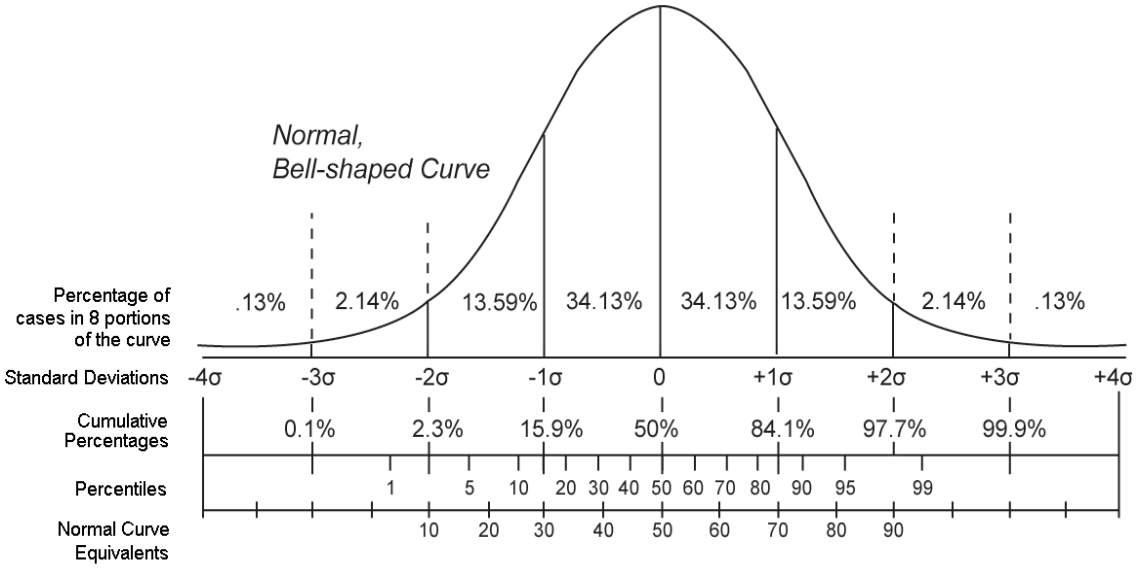


Figure 4-1 Normal Distribution Standard Deviations and Percentiles [Kemp, 13]

4.5.4 Complexity Analysis

It can be easily seen that the algorithm in Table 4-6 has a complexity of $O(N)+O(TN)+O(N\log N)$. Steps 2 and 3 (Computing returns and volatility) require $O(N)$ runtime. Step 6 (Computing the scenario trials) takes $O(TN)$. Finally computing the risk (percentile function) takes $O(N\log N)$.

4.5.5 Security Policy Evaluation

In order to define a context based security policy, under the Random Walk - Monte Carlo method framework previously presented, we require the following parameters:

1. Smoothing factor (α)
2. The number of scenario trials (T)
3. Evaluation horizon (H)
4. Time series frequency (F)

5. Number of observations required in a context variable time series (N)

6. The confidence level (γ) to be used for risk evaluation

7. For each context variable in the policy, a risk (error) tolerance is defined.

Table 4-8 shows the description of the example Context Based Security Policy described in Section 4.5.2.

Table 4-8 A Description of the Context Based Security Policy Described in Section 4.5.2

Name: Unit member location access policy.
Description: A soldier can access location of his unit members according to his current position and proximity of his commanding officer.
Entities: • *so*: Soldier • *co*: Commanding Officer Data: • *LD*: Location data
Context policy parameters: • $F = H = 1$ minute • $T = 200$ • $\gamma = \alpha = 0.95$ • $N = 10$
Pseudo policy definition: Access to *LD* is granted to *so* iff 1. *proximity(so, co)* < 70 meters 2. *volatility (proximity(so,co,N))* <= 0.01 3. *error(proximity(so,co,N))* <=0.6
Functions: • *proximity(x,y): returns the proximity between entities x and y* • *proximity(x,y,n): returns a time series with the last n proximity values between x and y .* • *volatility(ts): returns the volatility of the time series ts.* • *error(ts): returns the estimation error through a Monte Carlo simulation for time series ts.*

In this table we can see also the minimum and maximum simulated error values as well as their standard deviation.

According to [RMG, 04], the choice of smoothing factor is linked to the

evaluation horizon. For security policies, we would prefer to have stable volatility estimate is desirable. Therefore following [RMG, 04] stable volatility estimates can be constructed using a smoothing factor that ranges from 0.94 to 0.97.

Finally, in order to know what confidence level γ to be used in a security policy, we could use the following formula described in [RMG, 06]:

$$\gamma = 1 - \frac{1}{T+1} \qquad (4.12)$$

where T is the number of scenario trials specified in the security policy.

Table 4-9 Simulation Results for Proximity Data in Table 4-7

Proximity Simulation	Error	Simulated Value
percentile (risk)	0.415	
Minimum	-0.581	67.418
Maximum	0.7838	68.783
Std Deviation	0.264	0.2640
(P_t – risk)		67.585
(P_t + risk)		68.415

4.6 Expanded Random Walk Model

In Section 4.4 introduced the concept of a random walk model for a single context variable. So for a context security policy in which all the context variables are independent of each other, we simply have evaluate the algorithm in Table 4-6 to calculate the risk of each context variable separately. However, in many situations context variables are related to each other. Therefore we want to modify our previous model in order to take into account the correlation amongst the context variable of a security policy.

4.6.1 Expanded Example

Let us expand the scope of the interaction between a soldier and his commander in the example given in 4.5.2. In this example, a soldier that moves about in the battlefield and dynamically retrieves enemy data from his nearby commander. In order to receive intelligence data, a soldier must first identify himself with his company commander and his platoon leader (data is restricted according to identity). The intelligence data is delivered provided that the soldier and his commanders (company and platoon leader) are within a 25-kilometer radius of a passing ScanEagle [Boeing, 07] UAV. Moreover, in order to avoid leaks, the platoon leader has to be within 150 meters of the company commander. To make sure that all the data is delivered, the soldier's mobile unit must have a minimum of 60% power availability. Finally, in order to receive any data, the connection rate amongst the soldiers needs to be above a specific threshold to guarantee that the communications are not being jammed (again data transfer is limited to a specific connection mode).

4.6.2 New Challenges

As the example points out, the security implementation that allows the soldier to receive the intelligence data must take into account the variability and correlation of the commanders and the UAV location information, the communication connection rate and power of his mobile unit. Thus, in order to make a reliable security decision, the security system should try to predict how much the different variables (i.e. commanders and UAV location, etc.) would change by the time the enemy data is delivered to and used by the soldier, since by the time the data is delivered, any of the entities (solider, commanders or UAV) may have moved to a new place making the security decision

invalid. Moreover, the security system should be able to detect, tolerate and ignore invalid context assertions made by compromised or hostile entities. Furthermore, the security implementation should be able to use a certain amount of data history for each of context variables in order to not only predict future values but also to measure the degree of change (volatility) of the context variables in order to arrive at a more accurate security decision. In addition, the security system needs to determine the risk involved when making a security decision because of the changeability and correlations of the context variables. Finally, due to the dynamic nature of a context variable, each security decision should hold only for a certain degree of time.

As such we need extend our previous work to model the behavior of context variables in a security policy using a random walk framework that takes into correlations amongst them. Finally, we further extend the previous to include a risk measure for the overall change in a policy according to the changes of its constituting context variables.

4.6.3 Random Walk for Multiple Context Variables

Using equation (4.6), a context-based security policy with M context variables, the behavior of the returns of each of the context variables can be described as:

$$r_{1,t} = \sigma_{1,t}\varepsilon_{1,t},\ r_{2,t} = \sigma_{2,t}\varepsilon_{2,t}\ \ \ldots.\ r_{M,t} = \sigma_{M,t}\varepsilon_{M,t}$$

Since the context variables may be related amongst themselves, we have to account for their movements relative to one another (that is, the context variables may be statistically dependent). Thus, the linear association between each pair of returns must be quantified. These movements are captured by pair-wise correlations (ρ_{ij}).

Therefore, the εt's should come from a multivariate normal (MVN) distribution [Rose, 02] then:

$$\begin{bmatrix} \varepsilon_{1,t} \\ \varepsilon_{2,t} \\ \dots \\ \varepsilon_{N,t} \end{bmatrix} \sim MVN\left(\begin{bmatrix} \mu_1 \\ \mu_2 \\ \dots \\ \mu_M \end{bmatrix}, \begin{bmatrix} 1 & \rho_{12,t} & \cdots & \rho_{1M,t} \\ \rho_{21,t} & 1 & \cdots & \rho_{2M,t} \\ \dots & & & \\ \rho_{M1,t} & \rho_{M2,t} & \cdots & 1 \end{bmatrix} \right) \quad or\ \varepsilon \sim MVN(\mu_{G,t}, R_t)$$

where R_t represents the correlation matrix of (ε_1, ε_2,… ε_N) and the mean and variance are represented by [RMG, 96]:

$$\mu_{G,t} = \sum_{i=1}^{M} w_i \mu_i \ and\ \sigma_{G,t}^2 = \sum_{i=1}^{M} w_i^2 \sigma_i^2 + 2 \sum \sum_{i<j} w_i w_j \sigma_{ij,t}^2$$

Moreover, the term σ_{ij}^2 represents the covariance between returns for the context variables i and j. We must remember that the covariance of two random variables X and Y is defined as:

$$\sigma_{XY}^2 = E[(X - \mu_X)(Y - \mu_Y)] = E(XY) - E(X)E(Y)$$

Since E(X)=μx and E(Y)=μy (E is the mathematical expectation), both of which are equal to zero according to our model, then the covariance is simply defined as

$$\sigma_{XY}^2 = E(XY)$$

Finally, let's recall that the correlation coefficient of two random variables X and Y can be calculated as follows [Mendenhall, 07]:

$$\rho_{XY} = \frac{\sigma_{XY}^2}{\sigma_X \sigma_Y}$$

where σ_X and σ_Y are the standard deviations of X and Y, respectively. Let's recall that the correlation coefficient has the following properties [Davidian, 07]:

- ρ_{ij} scales the information (on association in the covariance) in accordance

with the magnitude of variation in each random variable, creating a "unitless" measure.

- $\rho_{ij} = \rho_{ji}$.
- If $\sigma_{ij} = \sigma_i\sigma_j$, then $\rho_{ij} = 1$. Similarly, if $\sigma_{ij} = -\sigma_i\sigma_j$, then $\rho_{ij} = -1$.
- ρii is always equal to 1, as random variable is perfectly positively correlated with itself.
- It may be shown that correlations must satisfy $-1 \leq \rho_{ij} \leq 1$.

Using the Exponential Moving Average (EMA) ([Kenney, 62], [Chou, 75]), an expression to estimate the covariance and correlation of context variable log changes (returns) can be constructed. As such the covariance formula is defined as [Mendenhall, 07]:

$$\sigma^2_{XY|t} = \frac{\sum_{i=1}^{T} \alpha^{i-1} \left(r_{X,t-i} - \bar{r}_1 \right) \left(r_{Y,t-i} - \bar{r}_2 \right)}{\sum_{i=1}^{T} \alpha^{i-1}}$$

4.6.4 Multiple Variable Scenario Generation Algorithm

In section 4.5, an algorithm was derived to apply the Monte Carlo method to one context variable. For the case of multiple context variables, we need to take into account the correlation amongst the context variables when generating future values. Table 4-10 describes the modified algorithm [Sanchez, 08].

The simulated log R* needs to be computed in such a way that the correlations amongst the context variables returns are maintained. For instance, by using the

Cholesky factorization [Golub, 96], it can be shown that the formulae to generate the simulated returns for three context variables would be as follows:

$$
\begin{aligned}
R_1 &= \sigma_1 Z_1, \\
R_2 &= \sigma_2 \left(\rho_{12} Z_1 + \sqrt{1-\rho_{12}^2}\, Z_2 \right), \\
R_3 &= \sigma_3 \left(\rho_{13} Z_1 + \frac{\rho_{23} - \rho_{12}\rho_{13}}{\sqrt{1-\rho_{12}^2}} Z_2 + \sqrt{1-\rho_{13}^2 - \frac{(\rho_{23} - \rho_{12}\rho_{13})^2}{1-\rho_{12}^2}} Z_3 \right)
\end{aligned}
$$

where σ_i is the volatility (standard deviation) for context variable i, ρ_{ij} is the correlation between context variables i and j and each Z is a generated IID normal value.

In general, generating correlated variates for M context variables can expressed as follows [RMG, 06]:

$$
\begin{aligned}
R_1 &= \sum_{k=1}^{n} a_{1,k} Z_k \\
R_2 &= \sum_{k=1}^{n} a_{2,k} Z_k \\
&\cdots \\
R_M &= \sum_{k=1}^{n} a_{M,k} Z_k
\end{aligned}
$$

or in matrix form

$$
R = \begin{pmatrix} R_1 \\ R_2 \\ \cdots \\ R_M \end{pmatrix}, Z = \begin{pmatrix} Z_1 \\ Z_2 \\ \cdots \\ Z_M \end{pmatrix}, A = \begin{pmatrix} a_{1,1} a_{1,2} a_{1,3} \ldots a_{1,n} \\ a_{2,1} a_{2,2} a_{2,3} \ldots a_{2,n} \\ \ldots\ldots\ldots\ldots \\ a_{M,1} a_{M,2} a_{M,3} \ldots a_{M,n} \end{pmatrix} \equiv R = AZ
$$

Table 4-10 Scenario Generation Algorithm for Multiple Context Dependent Variables

Input: $\left(\{V_1\}_{t=1}^{N},\{V_2\}_{t=1}^{N}\ldots\{V_M\}_{t=1}^{N}\right)$
1. Choose the number of scenario trials T and smoothing factor α
2. For each context variable time series *i (1≤ i ≤ M)* 2.1 Compute the *N-1* returns from $\{V_{i,t}\}_{t=1}^{N}$ The result of this calculation is a time series of returns of the form $\{r_{i,t}\}_{t=2}^{N}$ 2.2 Using $\{r_{i,t}\}_{t=2}^{N}$ compute the variance series $\{\sigma_{i,t}^2\}_{t=2}^{N}$ using equation 4.7 and then compute the *volatility* (standard deviation) at time *t=N,* that is $\sigma_{i,N}=\sqrt{\sigma_{i,N}^2}$ 2.3 Define a time series $\{S_{i,t}\}_{t=1}^{T}$ to store the context variable simulated values 2.4 For each trial *j (1≤ j ≤ T)* 2.4.1 Compute the simulated log return *R** 2.4.2 Compute the jth simulated value S_j as follows: $S_j = V_N^i \exp(R^*)$ where *exp(x)* = e^x and V_N^i is the value of context variable i at time *N*
Output: $S_G=\left(\{S_1\}_{t=1}^{T},\{S_2\}_{t=1}^{T}\ldots\{S_M\}_{t=1}^{T}\right)$

The matrix A must satisfy the covariance requirements. By eliminating the random vectors R and Z, we have the following:

$$
\begin{aligned}
R &= AZ \\
RR^T &= AZ(AZ)^T = AZZ^TA^T \\
E[RR^T] &= E[AZZ^TA^T] = AE[ZZ^T]A^T \\
E[RR^T] &= AIA^T \\
\Sigma &= AA^T
\end{aligned}
$$

Several methods can be used to solve the set of equations for AA^T and generate correlated variates, for instance, Cholesky Decomposition [Golub, 96], Singular Value Decomposition [Golub, 96] and Return Space Decomposition [Benson, 97]. The authors in [RMG, 01], [RMG, 06] and [Benson, 97] showed that the matrix A can be expressed as follows: matrix A must satisfy the covariance requirements. By eliminating the random vectors R and Z, we have the following:

$$A = \frac{1}{\sqrt{\sum_{i=1}^{n} \alpha^i}} \begin{pmatrix} \alpha^{\frac{1}{2}} r_{1,1} & \alpha^{\frac{2}{2}} r_{1,2} & \alpha^{\frac{3}{2}} r_{1,3} \ldots & \alpha^{\frac{n}{2}} r_{1,n} \\ \alpha^{\frac{1}{2}} r_{2,1} & \alpha^{\frac{2}{2}} r_{2,2} & \alpha^{\frac{3}{2}} r_{2,3} \ldots & \alpha^{\frac{n}{2}} r_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ \alpha^{\frac{1}{2}} r_{M,1} & \alpha^{\frac{2}{2}} r_{M,2} & \alpha^{\frac{3}{2}} r_{M,3} \ldots & \alpha^{\frac{n}{2}} r_{M,n} \end{pmatrix}$$

where n=N-1 is the number of log returns, α is the smoothing factor and $r_{i,j}$ is the return of context variable *i* at time j.

The advantages of using the Return Space Decomposition (RSD) over the Cholesky and SVD factorizations can be summarized as follows [RMG,06]:

- In both, Cholesky and SVD factorizations, the decomposed matrix does not easily provide an intuitive understanding of how the future values are generated and the change of a single value of a context risk factor requires a new decomposition. Finally, the Cholesky factorization requires that the correlation matrix be PD (positive definite), and SVD requires PSD (positive semi-definite).
- Volatilities and correlations do not have to be computed when using Return Space Decomposition.

Table 4-11 describes the modified algorithm for *M* context dependent variables using the Return Space Decomposition:

4.6.5 Risk

As explained in section 4.5.3, due to the dynamic nature of context variables, we will measure the *risk* of each context variable (and for a group of context variables) as the maximum *error* amount that is incurred in the estimation of future values (for each context variable) during an evaluation horizon for a given confidence level. In this framework, to measure the risk of a context variable, the following is assumed:

Table 4-11 Scenario Generation Algorithm for Multiple Dependent Context Dependent Variables Using Space Return Decomposition

Input: $\left(\{V_1\}_{t=1}^{N},\{V_2\}_{t=1}^{N}\ldots\{V_M\}_{t=1}^{N}\right)$
1. Choose the number of scenario trials T and smoothing factor α
2. 2.1 Compute the *N-1* returns for each of the M context variables time series $\{V_{i,t}\}_{t=1}^{N}$ 2.2 Compute the α weights vector $lwv=\left(\alpha_1^{\frac{1}{2}},\alpha_2^{\frac{2}{2}},\alpha_3^{\frac{3}{2}},\ldots,\alpha_n^{\frac{n}{2}}\right)^T$ and the $ssq=1\Big/\sqrt{\sum_{i=1}^{n}\alpha^i}$
3. <u>For each context risk factor time series *i (1≤ i ≤ M)*</u> 3.1 Compute the A Matrix, where each $a_{ij},(1\le i\le M,1\le j\le N)=r_{ij}\cdot lmv\cdot ssq$ where r_{ij} is the return of context variable *i* at time *j*. 3.2 Define a time series $\{S_{i,t}\}_{t=1}^{T}$ to store the context variable simulated values <u>3.3 For each j trial *(1≤ j ≤ T)*</u> 3.3.1 Compute an N size vector $Z=(z_1,z_2,z_3,\ldots,z_n)^T, z_i\sim IID\ N(0,1)$ 3.3.2 For each context variable *k* <u>*(1≤ k ≤ M)*</u>, compute $SLR=(A_k\circ Z)\cdot\sqrt{H}$ where A_k is the row-vector corresponding to context variable *k* and the evaluation horizon *H*. 3.3.3 Compute the j^{th} simulated value S_j as follows: $S_{i,j}=V_N^i\exp(SLR)$ where *exp(x)* = e^x and V_N^i is the value for the context variable *i* at time *N* 2.1 Compute the *N-1* returns from $\{V_{i,t}\}_{t=1}^{N}$. The result of this calculation is a time series of returns of the form $\{r_{i,t}\}_{t=2}^{N}$
Output: $S_G=\left(\{S_1\}_{t=1}^{T},\{S_2\}_{t=1}^{T}\ldots\{S_M\}_{t=1}^{T}\right)$

1. Risk can be measured according to the changes of the context variables (e.g. location, power resource, etc.).

2. Without loss of generality, the values of a context variable are always positive. Moreover, the [log] changes in the context variables can be modeled using as a random walk (following a normal distribution).
3. Since the context variables are assumed to follow a normal distribution, an important property of the normal distribution is that the sum of normal random variables is itself normally distributed.[11]

To calculate the maximum *error* amount for each context variable, we need to create a set of time series $E = \left(\left\{ E_{1,t} \right\}_{t=1}^{T}, \left\{ E_{2,t} \right\}_{t=1}^{T} \ldots, \left\{ E_{i,t} \right\}_{t=1}^{T}, \ldots \left\{ E_{M,t} \right\}_{t=1}^{T} \right)$, where $\left\{ E_{i,t} \right\}_{t=1}^{T}$ is the estimation error series for context variable *i* and defined as follows:

$$\left\{ E_{i,t} \right\}_{t=1}^{T} = S_{i,j} - V_N^i$$

where V_N^i is the value for the context variable *i* at time *N*

Again, to measure the risk, we use the ***percentile*** function to determine the proportion of values in the time series $\left\{ E_{i,t} \right\}_{t=1}^{T}$ that a specific magnitude will not be exceeded by a specific magnitude. This means that, for a context variable, this measure is the maximum error amount (*risk*) that would be incurred in the estimation during the evaluation horizon *H* for a given *confidence level (γ)*.

Formally, the p^{th} percentile of the $\left\{ E_{i,t} \right\}_{t=1}^{T}$ values is defined as the magnitude that exceeds *p* percent of the values, that is:

$$risk_{i,P} = \left| percentile\left(\left\{ E_{i,t} \right\}_{t=1}^{T}, (1-\gamma) \right) \right|$$

[11] These random variables must be drawn from a multivariate distribution.

4.6.6 Complexity Analysis

We developed and implementation of space return decomposition as described in Table 4-11 with a complexity of $O(MN)+O(MTN)+O(MN\log N)$. Step 2 (Computing returns and lmv) requires $O(MN)$ runtime. Step 3 (Computing the A matrix and the scenario trials) takes $O(MTN)$. Finally, computing the risk (percentiles) takes $O(MN\log N)$.

4.6.7 Security Policies

In our army scenario of section 4.6.1, a simple security policy can be stated as follows: a Soldier can receive the location of his unit members, provided his *proximity* to the company commander (Captain) is no less than 70 meters. Moreover, the proximity between the Captain and the soldier's unit leader Sergeant) needs to be no less than 50 meters (The Sergeant and the Soldier are within line of sight of each other). The commanding officer requires that the location of the Soldier and the Sergeant be reported every minute and that the *volatility* (standard deviation) of the data not exceed 0.01. Every minute the policy is reevaluated ($H = 1$). The commanding officer (Captain) uses a smoothing factor of 0.95 ($\alpha = 0.95$) and generates 100 Monte Carlo scenarios ($T = 100$) before returning a decision. Table 4-12 shows the description of the example Context Based Security Policy described above.

Table 4-12 An Example of an Expanded of the Context Based Security Policy in Table 4-8

Name: Unit member location access policy.
Description: A soldier can access location of his unit members according to his current position and proximity of his commanding officer.
Entities: 1) *so*: Soldier, 2) *co*: Commanding Officer, 4) se: Sergeant 3) *LD*: Location data
Context policy parameters: 1) H = 1 minute, 2) T = 200, $y = \alpha$ = 0.95, $N = 10$
Access to *LD* is granted to *so* iff 4.*proximity(so, co)* < 70 meters & proximity(co, se) < 50 meters & proximity (so, se) < 5meters 5.*volatility (proximity(so,co,N))* <= 0.01 & *volatility (proximity(co,se,N)) <=0.01* 6.*error(proximity(so,co,N))* <=0.6
***Functions**:* • *proximity(x,y): returns the proximity between entities x and y* • *proximity(x,y,n): returns a time series with the last n proximity values between x and y .* • *volatility(ts): returns the volatility of the time series ts* • *error(ts): returns the estimation error through a Monte Carlo simulation for time series ts.*

4.6.8 Security Policy Evaluation

In order to describe a context based security policy, under the Random Walk - Monte Carlo method framework presented in Sections 4.6.3, 4.6.4 and 4.6.5, we require the following parameters:

1. Smoothing factor (α)
2. The number of scenario trials (T)
3. Evaluation horizon (H)
4. Time series frequency (F)
5. Number of observations required in a context variable time series (N)

6. The confidence level (γ) to be used for risk evaluation
7. For each context variable in the policy, a risk (error) tolerance should be defined.

The following steps are used to evaluate the policy:

1. A soldier so sends a request to his commanding officer co for the location of his unit members.
2. The co retrieves the locally stored security policy, reads it and determines which context variables (i.e. proximity) are necessary for the evaluation of the policy
3. The co sends a request asking both the so and the se for their proximity information. Such a request contains:
    a. The context variable to be gathered (e.g. proximity)
    b. The number of past observations (N) in the context variable time series.
4. Upon receiving the requests both the so and se return the required time series information to the co.
5. The co runs the modified Monte Carlo algorithm (Table 4-11) to calculate the 1-minute volatility. It then calculates the risk of the context variables (proximity) with the specified confidence level (y) using the percentile function.
6. Once the data is computed the co determines if access to the data LD can be granted for the next 1 minute (H = 1) before the policy needs to be revaluated.

Chapter 5. TRUST MODEL

For a pervasive mobile computing environment, context data (e.g. location, network status, available power, etc.) is often used as way to control access to a set of services or information. These context-aware security services are required to be non-intrusive and easily adaptable to a changing environment or context [Hulsebosch, 05], as they can be deployed at and referenced by multiple types of static and mobile units (PDAs, laptops, servers, etc.). Unfortunately, traditional security implementations such as access control lists (ACL) [Swift, 02] and role base access controls (RBAC) ([Newmann, 01], [Park, 04]) cannot be easily adapted to the pervasive environment as they are ill equipped to handle security operations that include the user or service context. This is due to not only the highly dynamic number of entities in the system (it would not only be unfeasible to keep a reference to every entity since) at any given time, but also traditional security implementations cannot cope with the variability and uncertainty of the context variables.

To address the aforementioned drawbacks, current access controls systems for pervasive mobile environments are often implemented with the notion of trust [Chakraborty, 06]. Trust is then used to provide a form of control that allows an entity to reason about the reliability of others [Teacy, 06]. More specifically, trust can be used to account for uncertainty about the motivation and capabilities of other entities to perform actions as agreed [Patel, 05].

Trust for an entity is often built up over time by accumulating personal experience with it. Keeping track of an entity's behavior history allows other entities to judge how this trustee entity would perform in a given situation. Moreover, when direct

experience (behavior) for a particular entity is either limited or unavailable, some trust systems (i.e. [Shand, 04], [Chakraborty, 06], [Teacy, 06]) implement ways to ask other entities about their experiences with the former entity. This collective opinion of others regarding a particular entity is known as a recommendation. Recommendations are used to help assess the trustworthiness of an entity. However, in these current trust systems, trust and recommendations values are computed using only the outcomes (positive/negative) of the interacting entities and failed to take into account the dynamic nature of context information used to reach either a particular outcome or security decision. By ignoring the variability, uncertainty and reliability of the context data, these systems failed to account the risk (or negative impact) of a security decision.

The above observations motivate us to revisit the problem of access controls, trust and risk in pervasive mobile environments. In Chapter 4 ([Sanchez, 07] and [Sanchez, 08]) have allowed us to define a framework to evaluate a context security policy by computing the risk of context variables according to their changes. In this chapter, we expand that previous work by introducing a multi-level trust framework (*ContextTrust*) in which trust for an entity is computed using not only on the results past interactions, but also on the recommendations given by other entities and the quality of the provided context data.

Before introducing the details of *ContextTrust*, let's review some concepts and definitions that will be used later in this chapter

5.1 Definitions

In order to define a trust model, it is important to define the concepts of trust and risk.

- ***Trust***: Trust is defined as "the extent to which one entity is willing to depend on another entity in a given situation with a feeling of relative security, even though negative consequences are possible" [Jøsang, 04]. Under *ContextTrust*, a number between -1 and 1 represents an entity's trust level. In addition, trust is not symmetrical, that is, if an entity A trusts another entity B, it does not mean that B trusts A. Finally, ContextTrust assumes that trust is not transitive, that is, if A trusts B and B trusts C, it does not necessarily means that A trusts C.
- ***Risk***: The word "risk" refers to situations in which it is possible but not certain that some undesirable event will occur. In everyday usage, 'risk' is often used synonymously with 'probability' of an unwanted event, which may or may not occur [Hansson, 07]. In *ContextTrust* risk is defined in terms of a risk function. To measure the risk of a context factor (or context variable), we use the **percentile** function to determine the maximum error amount (risk) that would be incurred in the estimation during the evaluation horizon H for a given confidence level (γ).

5.2 ContextTrust

In section 3.1.5, we illustrated the steps that are taken whenever a security interaction between two entities occurs.

1. A security request from a principal (accessing entity) arrives (Step 1) to a Context Security Analyzer (CSA).
2. The CSA reads the security policies for the requested Context Security

Object (CAO) and the accessing Principal.

3. The CSA may contact one or more Context Providers (CPs) to get the requested context data (risk factors) time series specified in the security policy.
4. Context Providers retrieve the requested context information.
5. Context Providers pass the retrieved context information as well as their trust values with respect to the Principal to the CSA.
6. The CAS calculates different trust and risk values to either grant or deny the access request.

Under *ContextTrust* whenever a security interaction occurs, the trust between the CAS and the Principal is modified. Therefore, following the syntax in [Chakraborty, 06], *ContextTrust* defines a trust relation between two entities A and B for a time window Wt as follows:

$$(A \rightarrow B)_{Wt} = \left[{}_{A}D_{B}, {}_{A}E_{B}, {}_{\varphi}R_{B} \right]$$

where

- ${}_{A}D_{B}$ represents the trust between the entities A and B based on the context data provided by B (or on behalf of B) to A.
- ${}_{A}E_{B}$ represents the trust derived from the experience or interactions between A and B.
- ${}_{\varphi}R_{B}$ represents the cumulative effect of all recommendations that A receives about B.

We assume that the value of a trust relation is expressed in terms of a numerical value in the range $[-1,1] \cup \{\psi\}$. A negative value is used to indicate distrust, whereas a

positive value denotes trustworthiness. A value of 0 (zero) indicates trust-neutrality. To indicate a lack of value due to insufficient information we use the special symbol ψ . In many ad-hoc networks a trust level below 0.5 is considered untrustworthy [Velloso, 08].

5.2.1 Computing the Data Trust Component

Given an N size time series $\left\{V_{i,t}^{C}\right\}_{t=1}^{N}$ of values for a context factor *C*, we model data trust component (for *C*) in terms of the classification accuracy and the percentage of missing elements in $\left\{V_{i,t}^{C}\right\}_{t=1}^{N}$. That is, *ContextTrust* rewards with more trust a time series with higher classification probability and low degree of missing observations (higher degree of quality).

Formally, the data trust component denoted as ${}_{A}D_{B}^{C}$ (without loss of generality, the letter *C* was added as a superscript to the original notation) is calculated as follows:

$$ {}_{A}D_{B}^{C} = \beta_{C}\left(1-\frac{2\sum_{i=1}^{M} f_{miss}\left(\left\{V_{i,t}^{C}\right\}_{t=1}^{N}\right)}{M}\right)+\Upsilon_{C}\left(1-\frac{2\sum_{i=1}^{M} 1-PET_{A}^{i}\left(\left\{V_{i,t}^{C}\right\}_{t=1}^{N}\right)}{M}\right) $$

Equation 5-1 Data Trust Component

where

- M is the number of context providers for context variable C values.
- $\left\{V_{i,t}^{C}\right\}_{t=1}^{N}$ is the N size times series values for context variable C supplied by context provider *i.*
- $f_{miss}\left(\left\{V_{i,t}^{C}\right\}_{t=1}^{N}\right)$ is a function that returns the percentage of missing points in

$\left\{V_{i,t}^{C}\right\}_{t=1}^{N}$ (section 3.3.1.2)

- $PET_{A}^{i}\left(\left\{V_{i,t}^{C}\right\}_{t=1}^{N}\right)$ is the probability that A correctly classifies the time series $\left\{V_{i,t}^{C}\right\}_{t=1}^{N}$ provided by context provider *i* on behalf of B [sections 2.4.2 and 2.4.4].
- The factors $\beta_C \; and \; \Upsilon_C$ are user defined weights to indicate the level of relevance that entity A – in the calculation of ${}_{A}D_{B}^{C}$ - gives to not only the percentage of missing points, but also the probably that A classifies correctly the time series provided by B. $\beta_C \; and \; \Upsilon_C$ are positive values such that $\beta_C + \Upsilon_C \leq 1$ and whose values are initially set to 0.5 to reflect equal relevance. However, for a context security policy the proper values for $\beta_C \; and \; \Upsilon_C$ would depend not only on previous experimentation (simulation), but also on the context variable type and the physical environment (i.e. building, military setting, etc.) in which security policy being evaluated.

The term

$$\left(1-\frac{2\sum_{i=1}^{M}1-PET_{A}^{i}\left(\left\{V_{i,t}^{C}\right\}_{t=1}^{N}\right)}{M}\right)$$

shows that the higher the classification probability, the more trust is given. The factor of 2 in this term gives us a trust range from [-1, 1] instead of [0, 1]. Finally, the term

$$\left(1 - \frac{2\sum_{i=1}^{M} f_{missing}\left(\left\{V_{i,t}^{C}\right\}_{t=1}^{N}\right)}{M}\right)$$

indicates that the higher the number of missing points a context variable time series has, the less trust A should have for B since the quality of the data supplied by the context providers is not as high. Again, note the factor of 2, which increases the range to [-1, 1] from [0, 1].

5.2.2 Computing the Experience Trust Component

We model experience in terms of the number of events (positive and negative) during a specific window encountered by a truster *A* regarding a trustee *B* when assessing the risk for context variable C.

Before introducing the formula for computing the experience component, we need to define the concept of a *positive* (*negative*) experience for a context variable C.

Positive Experience for a Context Variable C

Let's recall our random walk model

$$S_{t+H} = V_t \cdot e^{(\sigma_{t+H} \cdot \varepsilon_t)}, \varepsilon_{t+H} \sim IID\, N(0,1)$$

That is, at any point in time, we simulate the future value S_{t+H} of a context variable C based on one time based parameter σ_{t+H} (standard deviation), the last period's value V_t, and a normally distributed random variable ε_{t+H}. Thus, using the modified Monte Carlo algorithm in (Chapter 4) to generate multiple simulated future values, we can calculate the maximum error ($risk_v$) (see section 4.5.3) amount that would be incurred in the estimation of *V's future* value during an evaluation horizon *H*

for a given *confidence level (γ)*. As a result, we expect the [real] future value V_{t+H} to be within the range [V_t - $risk_v$, V_t + $risk_v$]. Therefore, we shall say that for a context variable C, it is a *positive experience* occurs when

$$V_{t+H} \in \left[V_t - risk_v, V_t + risk_v\right]$$

Formula Description

ContextTrust incorporates the calculated risk (Chapter 4, [Sanchez, 07], [Sanchez, 08]) of a context variable C into a trust measure by rewarding with more trust whenever a positive experience occurs. That is, trust is increased when the [actual] future value of a context variable *C* does not exceed the risk in our prediction. Inversely, whenever our prediction is incorrect, *ContextTrust* decreases the trust amount for the interacting entities. Formally, the experience component trust between entity A and B for a context variable *C*, during a time window W (denoted as ${}_A E_B^{C,W}$) is calculated as follows:

$${}_A E_B^{C,W} = \left[\frac{P_{A,B}^C}{P_{A,B}^C + N_{A,B}^C}\left(\frac{\sum_{i=1}^{P_{A,B}^C}\left|\frac{V_i^C - risk_i^C}{V_i^C}\right|}{P_{A,B}^C}\right)\right] - \left[\frac{N_{A,B}^C}{P_{A,B}^C + N_{A,B}^C}\left(\frac{\sum_{i=1}^{N_{A,B}^C}\left|\frac{V_i^C - risk_i^C}{V_i^C}\right|}{N_{A,B}^C}\right)\right],$$

Equation 5-2 Experience Trust Component

where

- W is a time window for which positive and negative experience data are stored
- $P_{A,B}^C$ is the number of positive experiences during W between entities A and

B for context variable C

- $N_{A,B}^{C}$ is the number of negative experiences during W between entities A and B for context variable C
- V_i^C is the [actual] value of context variable C for the i^{th} interaction
- $risk_i^C$ is the risk for context variable C calculated at the time of the i^{th} interaction

As we can see, ${}_A E_B^{C,W}$ is based on both two main terms: successful and unsuccessful [past] interactions. The first term

$$\left[\frac{P_{A,B}^{C}}{P_{A,B}^{C} + N_{A,B}^{C}} \left(\frac{\sum_{i=1}^{P_{A,B}^{C}} \left| \frac{V_i^C - risk_i^C}{V_i^C} \right|}{P_{A,B}^{C}} \right) \right]$$

denotes the trust amount that is awarded according to the calculated risk. Whenever, the calculated risk is less; that is our prediction $\left(V_i^C - risk_i^C\right)$ is more accurate the more trust is awarded. If $P_{A,B}^{C}$ is 0 then the whole first term is set to 0

The second term

$$\left[\frac{N_{A,B}^{C}}{P_{A,B}^{C} + N_{A,B}^{C}} \left(\frac{\sum_{i=1}^{N_{A,B}^{C}} \left| \frac{V_i^C - risk_i^C}{V_i^C} \right|}{N_{A,B}^{C}} \right) \right]$$

denotes the trust amount that is subtracted whenever a negative experience occurs. In the first term (successful interaction term), the more accurate is, the more trust is given. As seen in the formula negative experiences with low risk are penalized with a smaller decrease trust amounts. That is, we want to make sure that a negative interaction with

low risk should decrease the trust by a smaller factor. If $N^{C}_{A,B}$ is 0 then the whole second term is set to 0.

5.2.3 Computing the Recommendation Component

The recommendation component trust is evaluated not only on the basis of the time series correlations supplied by the context providers of a context variable C, but also in the accuracy of classification. That is, *ContextTrust* rewards with more trust highly correlated data and penalize uncorrelated data. For instance, if two context providers supplied uncorrelated data, the trust for the recommendation should decrease, as it is possible that either or both of the context providers are supplying inaccurate data for C. Under *ContextTrust*, truster A uses the both trust values it has on the recommenders (C context providers) and the trust values the recommenders have on the trustee entity B to weight the actual correlations.

Formally, the recommendation component for a context variable C - ${}_{\varphi}R^{C}_{B}$ - whose values are supplied on behalf of entity B by M context providers, is computed as follows:

1) If entity A has local time series information about entity B for context variable C, then

$$ {}_{\phi}R^{c}_{B} = \alpha_{C}\left(\frac{\sum_{i=1}^{M}\tau^{C}_{A,i}\cdot\tau^{C}_{i,B}\cdot \mathrm{P}\left(X^{C}_{i,A}\right)\cdot f_{\delta}\left(\tau^{C}_{A,i},\tau^{C}_{i,B}\right)}{\sum_{i=1}^{M}\tau^{C}_{A,i}\cdot\sum_{i=1}^{M}\tau^{C}_{i,B}}\right)+\beta_{C}\left(\frac{\sum_{i=1}^{M}\tau^{C}_{A,i}\cdot PET\left(V^{C}_{i}\right)\cdot f_{\delta}\left(\tau^{C}_{A,i}\right)}{\sum_{i=1}^{M}\tau^{C}_{A,i}}\right) $$

Equation 5-3 Recommendation Trust Component with Local Information

2) If entity A does not have (does not store) local time series information about entity B for context variable C, then, we need to examine two cases according to the number of recommenders (M):

   i. If M=1 (call this recommender Z) then

      ${}_{\varphi}R_B^C$ = The time decayed trust A has for Z about context C - $v_{A,Z}^t$ (see section 5.2.4)

   ii. If M > 1 then ${}_{\varphi}R_B^C$ defined as:

$$ {}_{\phi}R_B^c = \frac{\sum_{i=1}^{M-1}\sum_{j=i+1}^{M}\left(\frac{X_{i,j}^C+1}{2}\right)\cdot\left(\frac{\tau_{A,i}^C+\tau_{A,j}^C}{2}\right)\cdot\left(\frac{\tau_{i,B}^C+\tau_{j,B}^C}{2}\right)\cdot f_\delta\left(\tau_{A,i}^C,\tau_{A,j}^C,\tau_{i,B}^C,\tau_{j,B}^C\right)}{M_\delta} $$

Equation 5-4 Recommendation Trust Component without Local Information

where

- M is the number of context providers or recommenders for context variable C
- $PET_A^i\left(\left\{V_{i,t}^C\right\}_{t=1}^N\right)$ is the probability that A correctly classifies the time series $\left\{V_{i,t}^C\right\}_{t=1}^N$ provided by context provide *i* on behalf of B [sections 2.4.2 and 2.4.4].
- The factors α_C *and* β_C are defined weights to indicate the level of relevance A – in the calculation of ${}_{\varphi}R_B^C$. Finally, α_C *and* β_C are positive values and $\alpha_C + \beta_C \le 1$
- $X_{i,j}^C$ represents the uncertainty of the recommendation about context C for

entities *i*, *j*. Under *ContextTrust* this uncertainty is expressed as $\rho_{i,j}^{\alpha}$, that is, the correlation of the time series returns given by providers *i* and *j* about context variable C.

- $\mathrm{P}\left(X_{i,A}^{C}\right)$ is a function defines as follows

$$\mathrm{P}\left(X_{i,A}^{C}\right)=\begin{Bmatrix} X_{i,A}^{C} \;\; if \;\; X_{i,A}^{C} \geq 0 \\ \varepsilon \;\; otherwise \end{Bmatrix}$$

ε represents a tiny value defined by A to make sure $\mathrm{P}\left(X_{i,A}^{C}\right)$ is always great or equal to zero. Thus avoiding two negative high correlation values generate a high positive value.

- α is a system defined smoothing factor used to calculate the volatility (and correlations) of context variable time series
- $\tau_{A,i}^{C}$ and $\tau_{A,j}^{C}$ are the trust values entity A has for recommenders *i* and *j*, respectively for context factor C
- $\tau_{i,B}^{C}$ and $\tau_{j,B}^{C}$ are the trust values recommenders *i* and *j* have for entity B for context factor C
- $f_{\delta}\left(x_1 \cdots x_k\right)$ is a delta function which defined as follows:

$$f_{\delta}\left(x_1 \cdots x_k\right)=\begin{Bmatrix} 0 \;\; if \;\; \exists x_i \; (i=1...k) \; x_i \leq \tau_{\min} \; and \; \tau_{\min} > 0 \\ 1 \;\; otherwise \end{Bmatrix}$$

The delta function $f_{\delta}\left(x_1 \cdots x_k\right)$ can take up to *k* trust values and allows *ContextTrust* to decide which correlation values are included in the calculation of ${}_{\varphi}R_{B}^{C}$. That is, *ContextTrust* decides (based on a threshold) which context providers have enough trust to be included in the trust

computation.

- $\tau_{\min}$ is the system (or policy) defined minimum acceptable threshold used to include a recommendation in the computation of ${}_{\varphi}R_B^C$
- M_δ is the number of times $f_\delta\left(\tau_{A,i}^C,\tau_{A,j}^C,\tau_{i,B}^C,\tau_{j,B}^C\right)$ is equal to 1. If M_δ is 0 then ${}_{\varphi}R_B^C$ becomes undefined.

When there is more than one recommender and no local context information, the term,

$$\left(\frac{X_{i,j}^C+1}{2}\right)\cdot\left(\frac{\tau_{A,i}^C+\tau_{A,j}^C}{2}\right)\cdot\left(\frac{\tau_{i,B}^C+\tau_{j,B}^C}{2}\right)$$

allows *ContextTrust* to weigh the correlation values by the average trust entity A has over the two context providers *i* and *j* and the average trust the context providers have over B. The lower trust amongst A, B, *i* and *j*, the lower the trust estimate.

5.2.4 Computation of the Trust Value

Once the trust components have been calculated, we need to compute the over all trust an entity A has for entity B for context factor C during a time window. The over all trust is simply computed as the weighted sum of the component trust values. Formally, context trust defines $\tau_{A,B}^C$ as follows:

$$\tau_{A,B}^C = W \circ (A \overset{C}{\rightarrow} B)_{W_t} = W_D \cdot {}_A D_B^C + W_E \cdot {}_A E_B^C + W_R \cdot {}_{\varphi} R_B^C$$

where *W* is a weight vector of the form W = [W_D, W_E, W_R] - *data, experience, recommendation* - such that $W_D + W_E + W_R = 1$ and $W_D, W_E, W_R \in [0,1]$. These weights represent the level of importance entity A gives to each of the context trust component values in the computation of the overall trust value.

The value for a normalized trust relationship allows us to revise the terms trust and distrust as follows [Teacy, 06]:

$$\tau_{A,B}^{C} = \begin{cases} [-1,0) \Rightarrow \textit{it is distrust} \\ 0 \Rightarrow \textit{it is neutral} \\ (0,1] \Rightarrow \textit{it is trust} \\ \psi \Rightarrow \textit{undecided} \end{cases}$$

That is, when the value of $\tau_{A,B}^{C}$ falls in the range [-1, 0] it implies the entity B is distrusted. If the value is 0 there is neither trust nor distrust, whereas when it falls in the range (0, 1] the entity B is trusted. A value of Υ is to indicate the trust is undefined.

Finally, adapting from [Teacy, 06] and due to effects such as forgetfulness of the human mind, the decay of trust over time is captured by the following expression.

$$v^{t+\Delta t} = v^{t} \cdot e^{-\left(v^{t} \cdot \Delta t\right)^{2k}}$$

Equation 5-5 Trust Decay Over Time

where

- $v^{t+\Delta t}$ is (without loss of generality the superscript c is omitted) the trust value $\tau_{A,B}^{C}$ between entity A, B for context C at time $t + \Delta t$.
- v^{t} is the trust value $\tau_{A,B}^{C}$ between entity A, B for context C at time t (i.e. current time)
- *k* is the time effect factor for that entity has for context C.

Chapter 6. PERFORMANCE ANALYSIS

This chapter presents the results of our simulation of the major components of *ContexTtrust* presented in Chapters 4, 5.

We initially present how different data mining classifiers contribute to the trust calculation. We then follow a performance analysis of the risk calculation done in *ContextTrust*.

6.1 Shapelets and classifiers

Following [Lines, 12], our first objective is to analyze how using different classifiers contribute to trust calculation. In our experiment, we use two real life data sets, GeoLife ([Zhang, 10]) and Intel Lab Data ([Bodik, 04]). The former is a GPS trajectory dataset was collected by Microsoft Research Asia GeoLife project for 182 users in a period of over three years (from April 2007 to August 2012).. This dataset is described by a sequence of time-stamped points, each of which contains latitude longitude and altitude information. These trajectories were recorded using different GPS devices with a variety of sampling rates (i.e. every 1~5 second). Guided by [Zheng, 12] and using visual inspection of the data, we picked 4 users who move in a similar direction and chose about 5000 data points, by sampling every minute, which correspond to more than 80 minutes worth observations in one day (1250 points per user). The aim is to try to classify and predict the identity of a user through the analysis of their location (proximity) time series.

The Intel Lab Data is collection of measurements from 54 sensors deployed at the Intel Berkeley Research lab [Tennenhouse, 04]. The time-stamped data was

collected using Mica2Dot sensors [Crossbow, 13], which recorded topology humidity, temperature, light and voltage values once every 31 seconds between February 28th and April 5th of 2004. By sampling this dataset every minute, we chose one-hour worth of temperature observations for 4 different sensors - 7,8,9,10 (see Figure 6-1) and restrict to 1250 data entries per sensor. We chose those sensors because of the proximity to each other.

To run our experiments, we used a quad core Intel Core I7 MacBook Pro with 16G of RAM. We followed [Benson, 97] to determine the recommended time series size for risk computation. In this case we chose 30, 100, 200, 300, 400, and 500 data points. Moreover, in order to find the best *K* time series shapelets, we used the code provided by [Lines, 12] and modified it to better exploit several computing cores. We restricted the maximum length of the extracted shapelets to be less than 100 points. Finally, we also use also the Weka data mining software [Hall, 09] for the different classifiers implementations (we use the default Weka classifier settings, except for Random Forests in which case we use 100 trees)

6.1.1 GeoLife Data

In this experiment, we used the GeoLife (GL) data to measure the classification accuracy using 8 different classifiers. Table 6-1 shows the average percentage classification accuracy and standard deviation for J48, C4.5, 1-NN, Naive Bayes (NB), a Bayesian Network (BN), Random Forest (RF), Rotation Forest (RoF) and a Support Vector Machine (SVM), using N/2 shapelets where N is the time series size. We chose N/2 to not only avoid introducing bias into the results, but also be consistent across all sub-data sets.

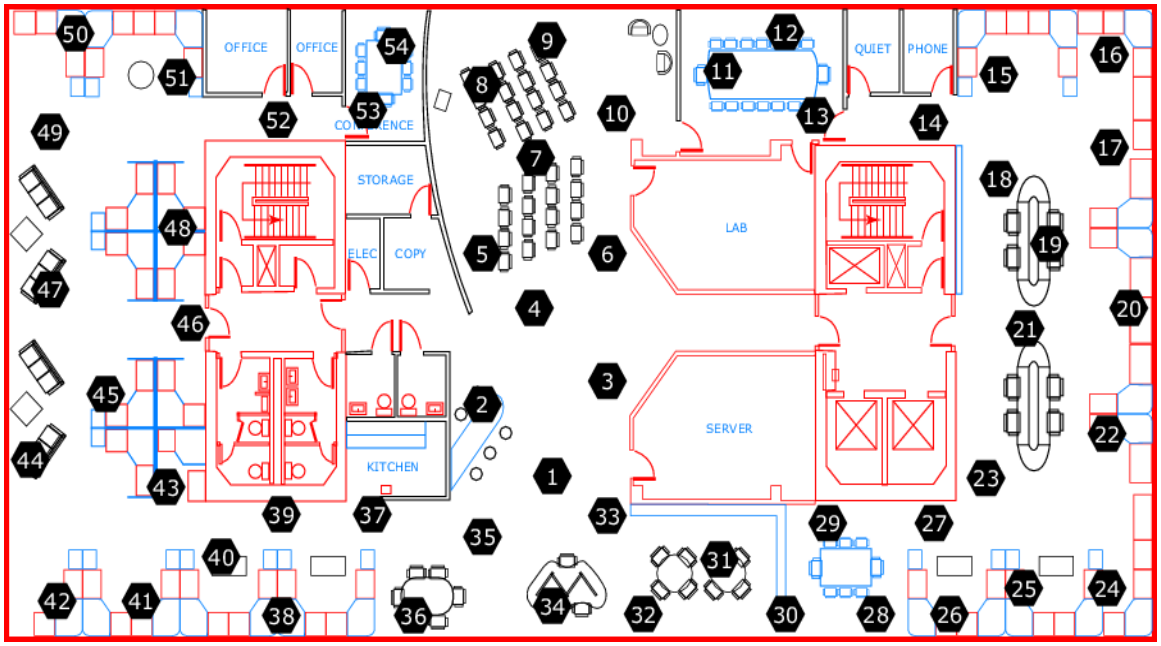


Figure 6-1 Intel Lab Data Sensor Location [Bodick, 04]

After trying with different shapelet sizes (i.e. 10, 15, 20, 40, 50, N/2), we concur with the results presented in [Lines, 12] and determined that using N/2 shapelets give us the best classification accuracy. Table 6-2 shows the algorithm used to create Table 6-1. As shown, we loop 10 times and extract from the original GeoLife data 6 sub-datasets with 40 time series each (10 per user). The time series sizes in each sub-dataset are 30, 100, 200, 300, 400 and 500 respectively. Once the data has been extracted we proceed to compute the *K* best shapelets (*ComputeShapelets*) restricting the size of each shapelets to be less than 100 points. Next we calculate the classification accuracy using the different classifiers (*CalculateClassificationAccuracy*). Finally, we average the average the accuracy amongst the 10 runs.

Table 6-1 Percentage Accuracy and Standard Deviation for 8 Classifiers Constructed with N/2 Shapelets

Data	J48	C4.5	1-NN	Naive Bayes	Bayes Net	Random Forest	Rotation Forest	SVM
GL-30	39.16 ±2.94	27.50 ±4.41	42.80 ±7.85	33.00 ±2.95	43.30 ±3.64	41.60 ±4.53	36.60 ±5.22	41.60 ±6.25
GL-100	30.50 ±4.15	31.25 ±3.83	29.25 ±2.88	35.50 ±8.01	37.50 ±8.55	45.15 ±4.81	33.75 ±4.60	40.00 ±6.10
GL-200	38.30 ±4.60	35.00 ±5.45	31.66 ±5.14	34.17 ±6.06	37.50 ±7.86	42.50 ±5.16	32.50 ±6.83	38.50 ±6.30
GL-300	50.00 ±4.55	45.83 ±3.05	49.16 ±3.34	36.20 ±5.56	46.67 ±7.66	50.83 ±4.38	47.50 ±6.38	37.61 ±6.12
GL-400	58.33 ±4.82	49.17 ±2.41	42.50 ±4.36	37.50 ±6.89	41.67 ±7.53	53.32 ±5.78	50.03 ±7.78	38.33 ±5.82
GL-500	65.83 ±5.19	51.00 ±2.82	41.66 ±3.53	37.50 ±7.24	41.67 ±8.49	72.05 ±4.78	53.33 ±7.78	42.50 ±5.94

It can be readily seen in Table 6-1, that tree classifiers (J48, C4.5, Random Forest and Rotation Forest) offer the best classification accuracy. Moreover, as the length of the time series increases, the better classification accuracy is achieved, with Random Forest improving the most overall, specially as we achieved similar accuracy measures as the work presented in [Ye, 09]. Table 6-3 details one tailed paired t-tests using a cutoff α of 0.05 between Random Forest and J48, C4.5, 1-NN, Naive Bayes, Bayes Net and SVM classifiers for the GL-300, GL-400 and GL-500 sub-datasets in order to reject the null hypothesis stating "The Random Forest classifier performs the same as the J48, C4.5, 1-NN, Naive Bayes, Bayes Net and SVM classifiers". Since we want to keep the experiment wise error rate to α=0.05, we use a Bonferroni adjustment [Jensen, 00] and divide α by the number of comparisons we make (in our case 6). That is, for any one comparison to be considered significant, the obtained p-value would have to be less than 0.00833 - and not 0.05. The significant values are bolded in Table 6-3.

Table 6-2 Algorithm to Compute Data in Table 6-1

```
Input: GeoLife (GL) data
T=0
For I = 1 to 10
  For S in (30, 100, 200, 300, 400, 500)
    K= S/2
    Extract from GL a sub-dataset GL-S for users 1 through 4 from time T to T+S
    S-GLS = ComputeShapelets(GL-S, K)
    For C in (C4.5, 1-NN, NB, BN, RF, RoF, SVM)
      accuracy = CalculateClassificationAccuracy(C, S-GLS)
      results[S][C][I] = accuracy
  T+=500
For S in (30, 100, 200, 300, 400, 500)
  For C in (C4.5, 1-NN, NB, BN, RF, RoF, SVM)
    m.average [S][C] =SUM(results[S][C]) / 10.0
    m.stdev[S][C] = STDEV(results[S][C])
Output: m_average, m_stdev
```

Table 6-3 One Tailed Paired T-Tests (α<0.05) Between Random Forest and J48, C4.5, 1-NN, Naïve Bayes, Bayes-Net and SVM classifiers using the GL-300, GL-400 and GL-500 sub-datasets

Data	J48	C4.5	1-NN	Naive Bayes	Bayes Net	SVM
GL-300	0.0126	0.0272	**0.0016**	0.0246	0.0242	0.0054
GL-400	0.0168	0.0129	0.0145	0.0162	0.0123	0.0243
GL-500	**0.0004**	**0.0078**	**0.0010**	0.0136	**0.0065**	**0.0232**

As we can see, as not only the number of shapelets but also the time series size increase, we can obtain very promising classification results when we use a random forest classifier.

Table 6-4 shows the average time in seconds to find the best N/2 shapelets from the GeoLife sub-data sets (*ComputeShapelets* function).

Table 6-4 Average Time to Find N/2 Shapelets

Data	Time (secs)
GL-30	1.22
GL-100	49.47
GL-200	65.91
GL-300	276.73
GL-400	690.53
GL-500	1270.58

One of the main motivations behind our work using shapelets for time series classification is to produce classification decisions that are interpretable. As such, Figure 6.2 shows a sample path for user 1 using the GL-30 sub-dataset. We convert each Latitude-Longitude coordinate by calculating the arc-length distance [Weisstein, 13b] to a reference point. As seen, the general areas where the best and matching and non-matching shapelets occur are shown. Because the GL-30 sub-dataset has the smallest time series of all, we found both a matching and non-matching shapelets in the same time series. When this occurs, our classification accuracy decreases. Figures 6-3 and 6-4 illustrate the six best matching and nonmatching shapelets extracted from GL-30. It is important to notice that the non-matching shapelets were recovered by extracting the shapelets with the minimum information gain and separation gaps (Section 2.4.3).

Similarly, Figure 6-5 illustrates two GL-100 sub-dataset sample paths for user 1 and 3. In this case, one of the best K matching shapelets comes from User 1. In the same manner one of the non-matching shapelet (least information gain) comes from user 3. Figures 6-6 and 6-7 illustrate the six best matching and nonmatching shapelets extracted from GL-100.

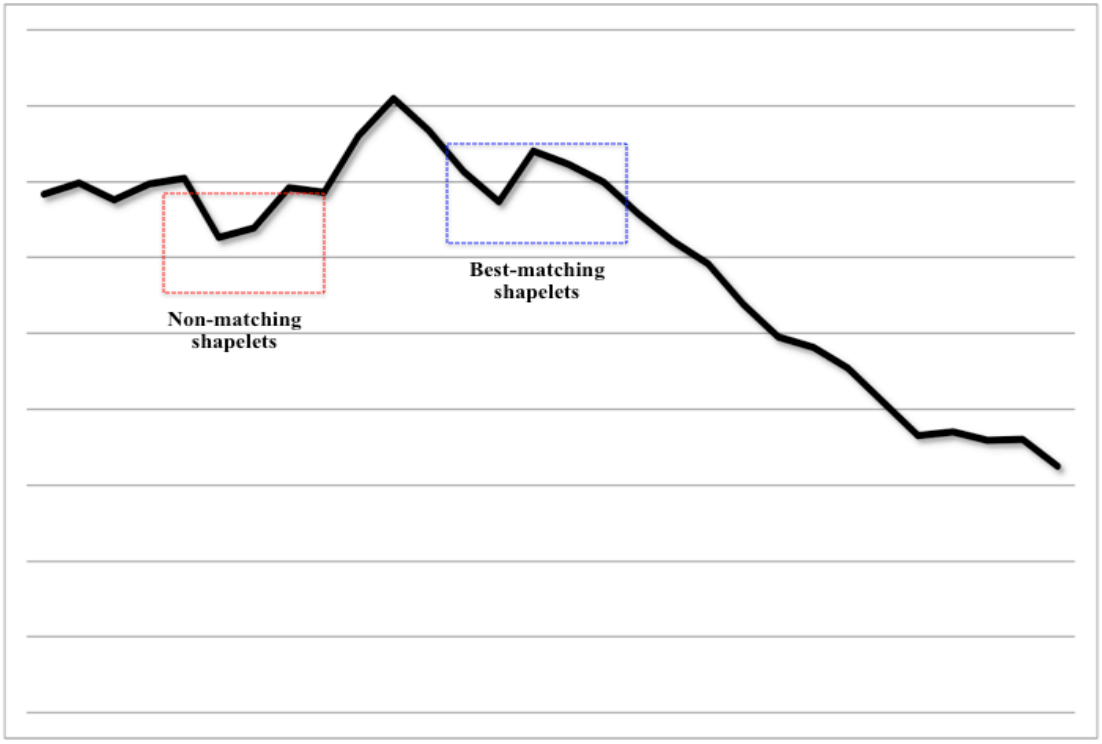


Figure 6-2 GL-30 Sample Path for User 1 and its Shapelet Matching and Non-Matching Areas

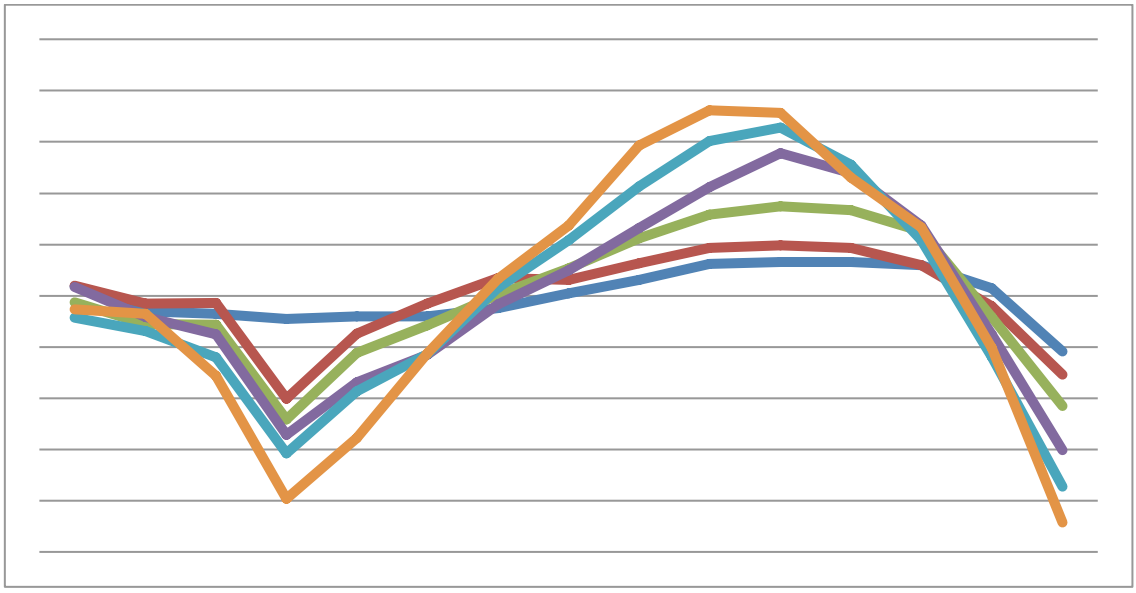


Figure 6-3 An Illustration of the Six Best Matching Shapelets Extracted From GL-30

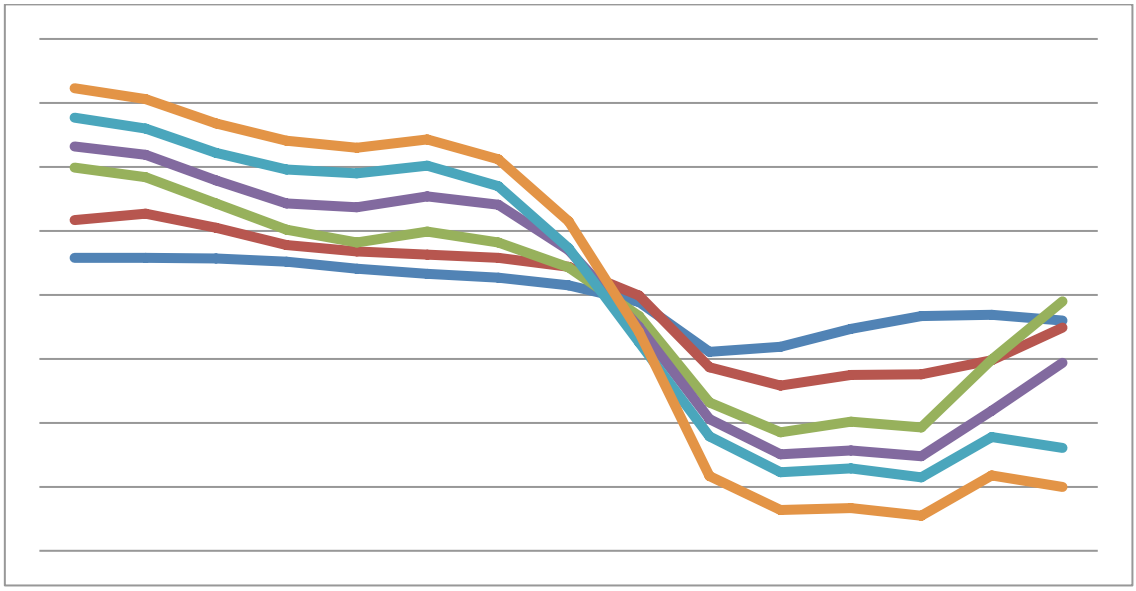


Figure 6-4 An Illustration of the Six Non-Matching Shapelets Extracted From GL-30

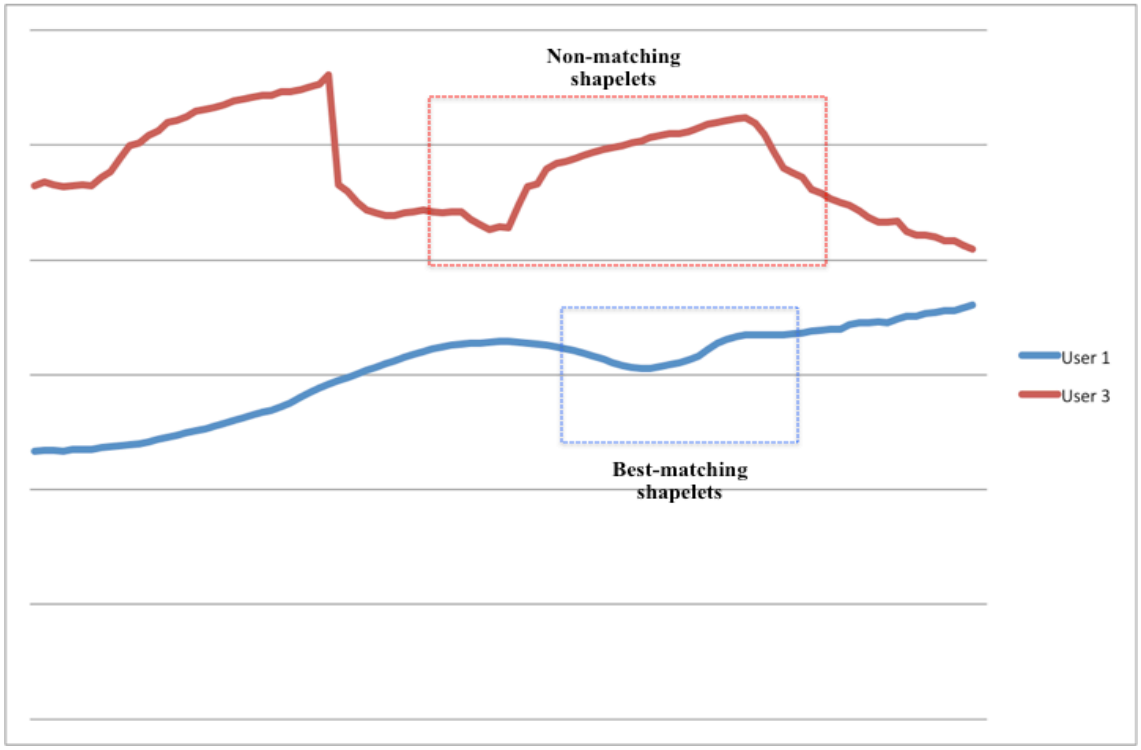


Figure 6-5 GL-100 Sample Paths for User 1 and 3 and their Shapelet Matching and Non-Matching Areas

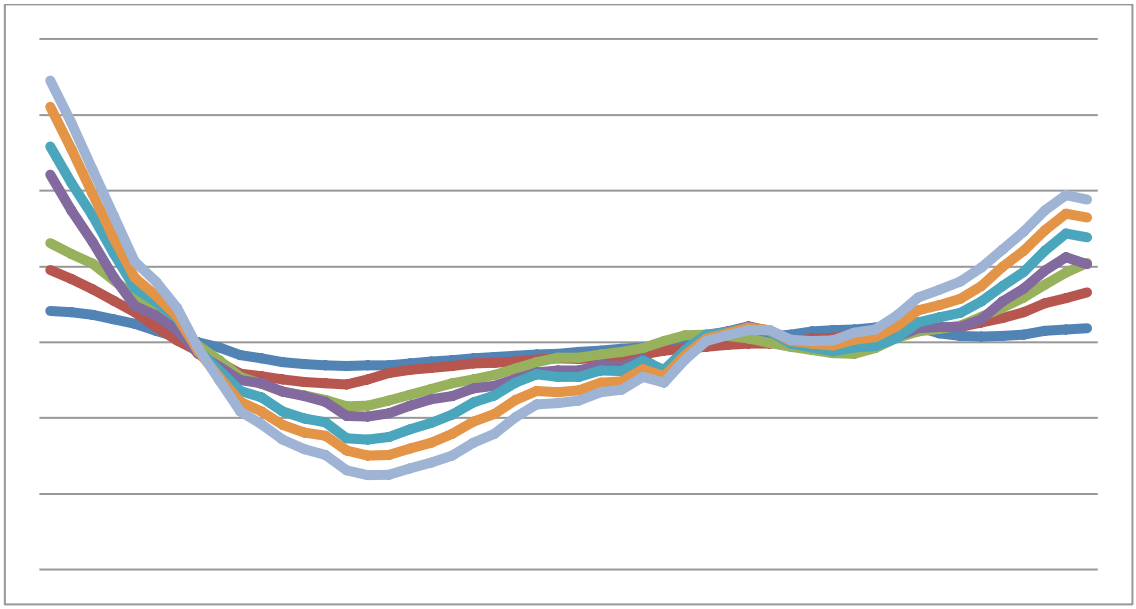


Figure 6-6 An Illustration of the Six Best Shapelets Extracted From GL-100

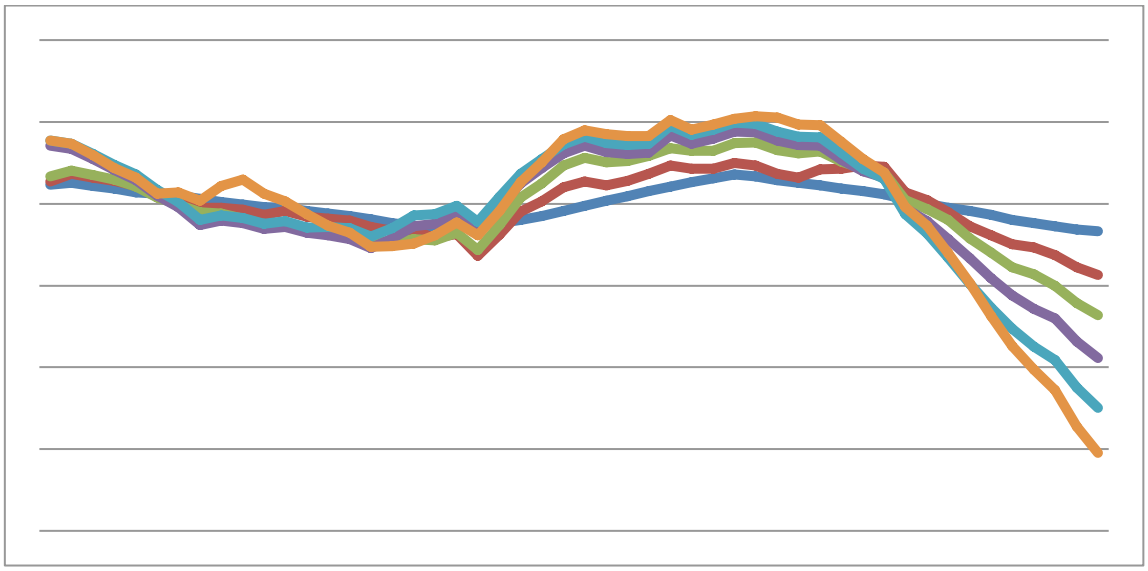


Figure 6-7 An Illustration of the Six Best Non-Matching Shapelets Extracted From GL-100

Using 10 different test datasets, we computed the average *ContextTrust* classifier data trust (Equation 1 with $\beta_C = 0$ *and* $\Upsilon_C = 1$) for 8 classifiers (see Figure 6-8) using the trained shapelets in GL-100. To create each test set, we drew at random 10 time series of size 100 from the original GeoLife data, from to users 1 through 4.

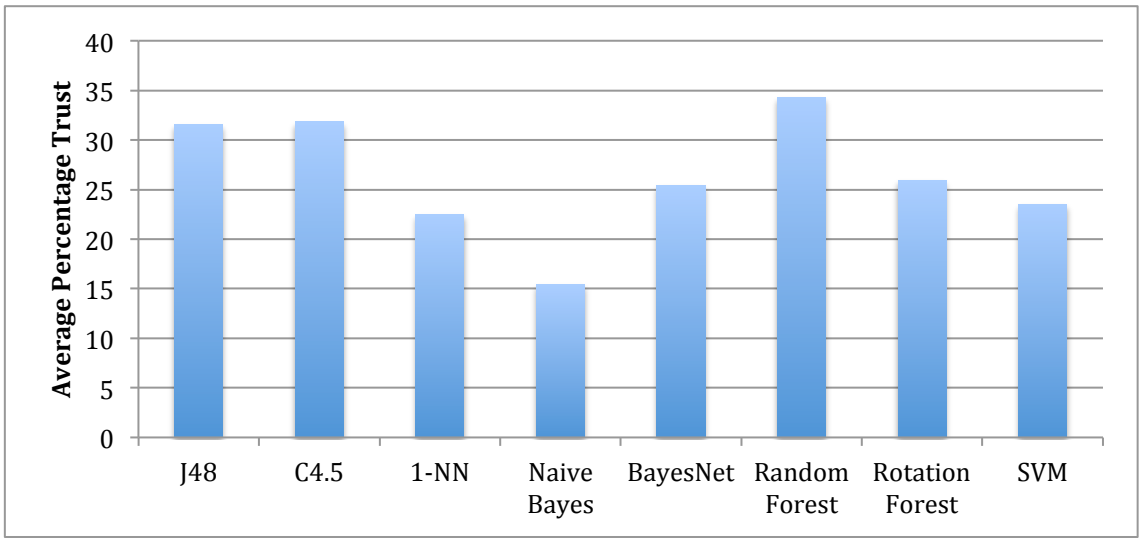


Figure 6-8 Average Percentage Data Trust for 8 Classifiers Contribution using GL-100

The purpose of this experiment was to estimate how accurately our time series shapelet classification algorithm performs in a possible real scenario (using a trained classifier to classify previously unseen records). Table 6-5 details one tailed pair t-tests using a cutoff α of 0.05 between random forest and the remaining classifiers using Figure 6-8 data in order to reject the null hypothesis stating "The mean percentage data trust between random forest and the remaining classifiers is the same." Using a Bonferroni adjustment we divide α by the number of comparisons made (in this case 7). That is, for any one comparison to be considered significant, the obtained p-value would have to be less than 0.00714. As seen, there is not any statistical difference. However, we believe that we may get higher classification trust values, if we use bigger trained shapelet datasets (i.e. GL-300, GL-400) use a better testing procedure and remove the restriction to have the shapelet size less than 100 data point since that would allow us to better capture the time series features of in the training dataset.

Table 6-5 Two Tailed Paired T-Tests (α<0.05) Between Random Forest and NN, Naïve Bayes, Bayes-Net, Rotation Forest and SVM classifiers for data in Figure 6-8

J48	C45	1NN	Naïve Bayes	Bayes Net	Rotation Forest	SVM
0.0369	0.0369	0.0218	0.0172	0.0240	0.0277	0.0243

6.1.2 Intel Lab Data

As done for the GeoLab data in the previous section, we use the Intel Lab data (ILD) to measure the classification accuracy for J48, C4.5, 1-NN, Naive Bayes, Bayesian Network, Random Forest, Rotation Forest and a Support Vector Machine classifiers using N/2 shapelets. Following a similar procedure as shown in Table 6-2, Table 6-6 shows the algorithm used to create the data in Table 6-7. Each ILD sub-dataset has 40 time series (10 per sensor) and the time series sizes in each sub-dataset are 30, 100, 200, 300, 400 and 500 respectively.

As in Table 6-1, Table 6-7 shows that as the length of the time series increases, the better classification accuracy is achieved.

Table 6-6 Algorithm to Compute Data in Table 6-5

```
Input: Intel Lab Data (ILD)
T=0
For I = 1 to 10
  For S in (30, 100, 200, 300, 400, 500)
    K= S/2
    Extract from ILD a sub-data set ILD-S for sensors 7 through 10 from time T to T+S
    S-ILD = ComputeShapelets(ILD-S, K)
    For C in (C4.5, 1-NN, NB, BN, RF, RoF, SVM)
      accuracy = CalculateClassificationAccuracy(C, S-ILD)
      results[S][C] += accuracy
  T+=500
For S in (30, 100, 200, 300, 400, 500)
  For C in (C4.5, 1-NN, NB, BN, RF, RoF, SVM)
    m.average [S][C] =SUM(results[S][C]) / 10.0
    m.stdev[S][C] = STDEV(results[S][C])
Output: m.average, m.stdev
```

Table 6-7 Percentage accuracy for 8 classifiers constructed with N/2 shapelets for ILD data

Data	J48	C4.5	1-NN	Naive Bayes	Bayes Net	Random Forest	Rotation Forest	SVM
ILD-30	36.60 ±3.93	41.60 ±4.90	20.84 ±5.92	34.17 ±8.13	38.33 ±4.92	40.00 ±4.60	35.00 ±5.65	33.33 ±3.91
ILD-100	47.50 ±6.98	47.50 ±6.62	35.84 ±4.13	31.60 ±6.30	40.00 ±7.21	43.33 ±6.50	41.67 ±7.30	30.00 ±9.31
ILD-200	41.60 ±7.07	41.60 ±7.41	30.00 ±7.67	33.30 ±11.97	41.60 ±8.14	37.50 ±8.61	48.34 ±8.94	26.66 ±10.70
ILD-300	42.50 ±8.29	45.84 ±8.85	33.34 ±7.57	28.30 ±10.01	39.17 ±7.24	43.33 ±9.50	42.50 ±10.19	25.83 ±11.54
ILD-400	46.66 ±7.67	43.30 ±8.97	25.83 ±8.85	23.33 ±11.46	41.30 ±9.09	46.66 ±8.89	37.50 ±11.87	22.50 ±9.82
ILD-500	41.67 ±8.62	45.80 ±9.45	25.01 ±10.60	27.50 ±12.73	39.17 ±11.91	50.83 ±9.79	40.83 ±10.28	24.16 ±10.84

Table 6-8 One Tailed Paired T-Tests (α<0.05) Between Random Forest and J48, C4.5, 1-NN, Naïve Bayes, Bayes-Net and SVM classifiers using the ILD-300, ILD-400 and ILD-500 sub-datasets

Data	J48	C4.5	1-NN	Naive Bayes	Bayes Net	SVM
ILD-300	0.0343	0.0943	0.0053	0.0176	0.0215	0.0172
ILD-400	0.0893	0.0437	0.0016	0.0162	0.0123	0.0243
ILD-500	0.0438	0.0862	0.0246	0.0136	**0.0065**	0.0232

Table 6-8 details one tailed paired t-tests using a cutoff α of 0.05 between random forest and J48, C4.5, 1-NN, Naive Bayes, Bayes Net and SVM classifiers for the ILD-300, ILD-400 and ILD-500 sub-datasets in order to reject the null hypothesis stating "The Random Forest classifier performs the same as the J48, C4.5, 1-NN, Naive Bayes, Bayes Net and SVM classifiers. Again, using a Bonferroni, we divide α by 6 (the number of comparisons we make). That is, for any one comparison to be considered significant, the obtained p-value would have to be less than 0.00833. The significant values are bolded in Table 6-8. As we can see there is no statistical difference between random forest and the rest of the classifiers (except for Bayes Net). Therefore for the Intel Lab Data, any classifier would give us the same level of accuracy. We believe this occur because there are missing and truncated entries in the

original Intel Lab data. The missing data entries occurred because sensor performance degraded as the power of their batteries drained [Bodik, 04].

In like manner, Figure 6.9 illustrates two sample readings for sensors 8 and 9 from ILD-30 and the general areas where the best matching and non-matching shapelets occur. Figures 6-10 and 6-11 illustrate the six best matching and nonmatching shapelets extracted from ILD-30. Likewise, Figure 6-12 illustrates two sample readings from ILD-100 for sensors 8 and 10 along with the general areas where the best matching and non-matching shapelets occur. Figures 6-13 and 6-14 illustrate the six best matching and nonmatching shapelets extracted from ILD -100.

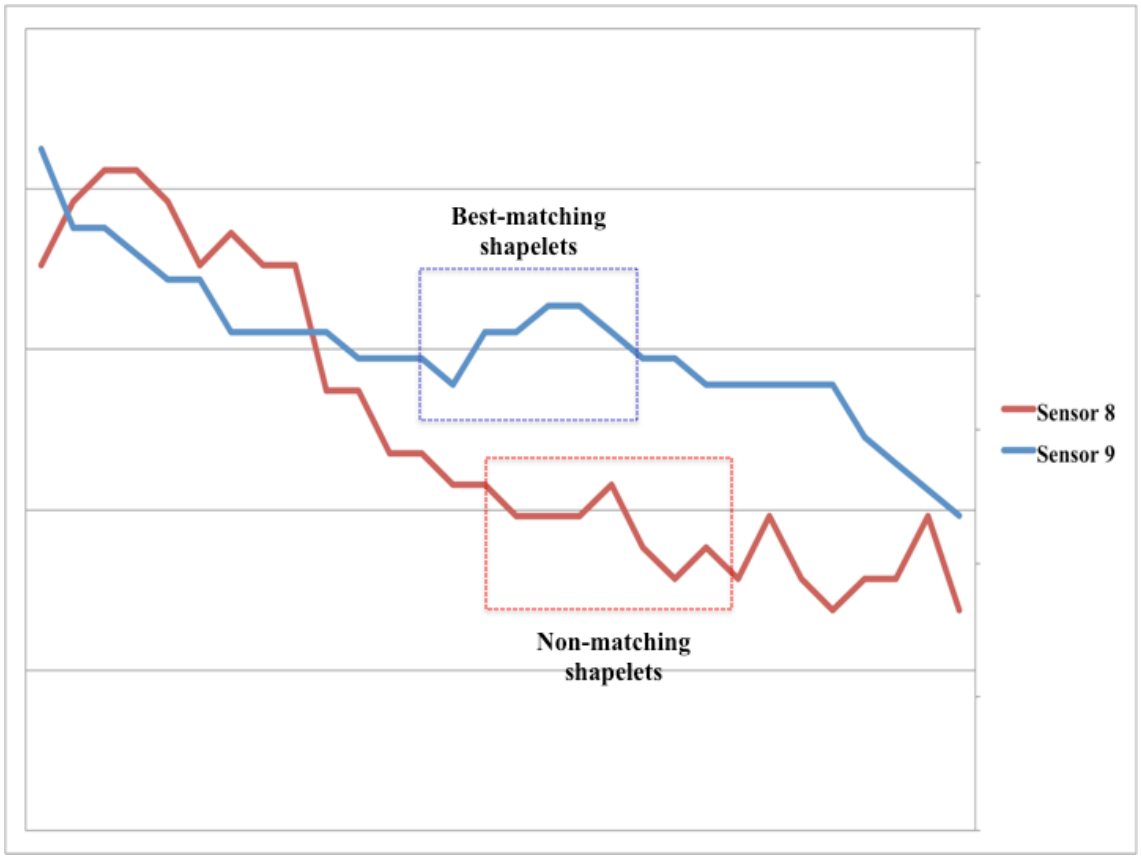


Figure 6-9 ILD-30 Sample Temperature Readings from Sensors 8 And 9 and Their Shapelet Matching and Non-Matching Areas

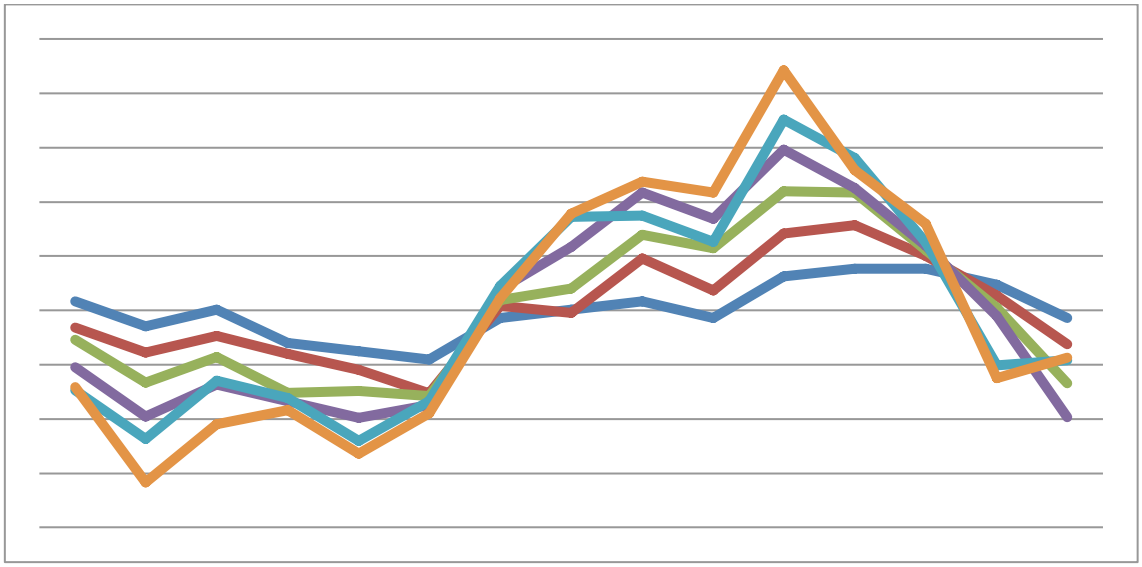


Figure 6-10 An Illustration of the Six Best Shapelets Extracted From ILD-30

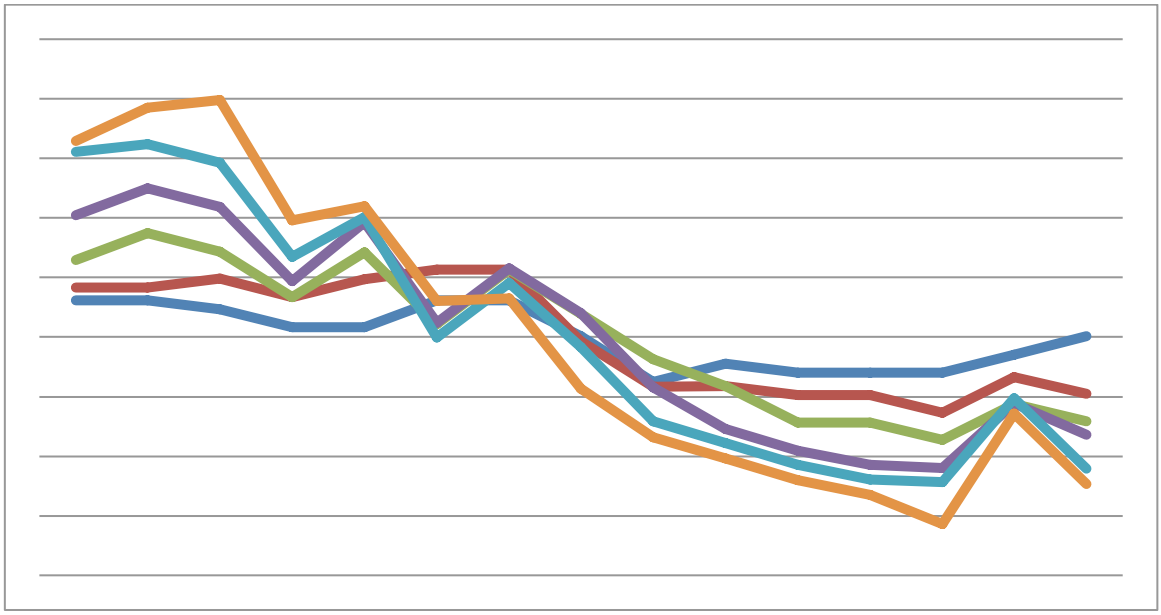


Figure 6-11 An Illustration of the Six Best Non-Matching Shapelets Extracted from ILD-30

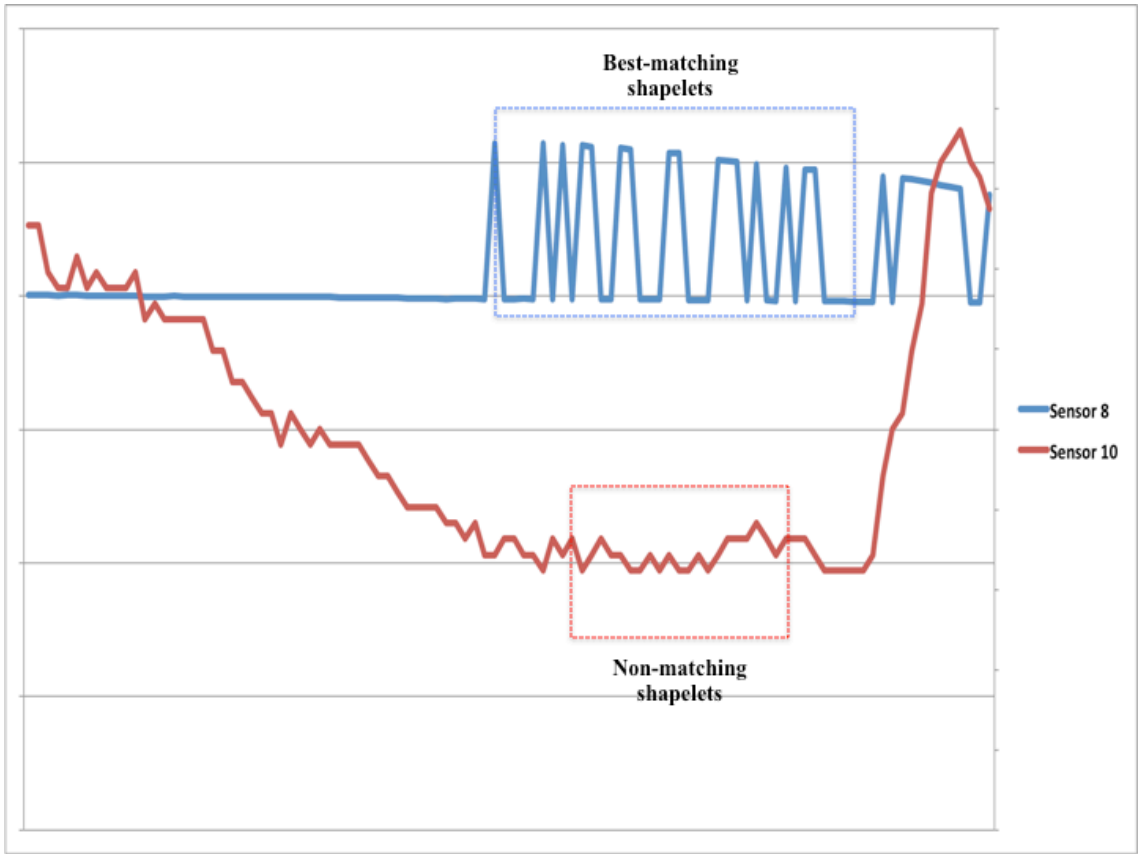


Figure 6-12 ILD-100 Sample Temperature Readings from Sensors 8 And 10 and their Shapelet Matching and Non-Matching Areas

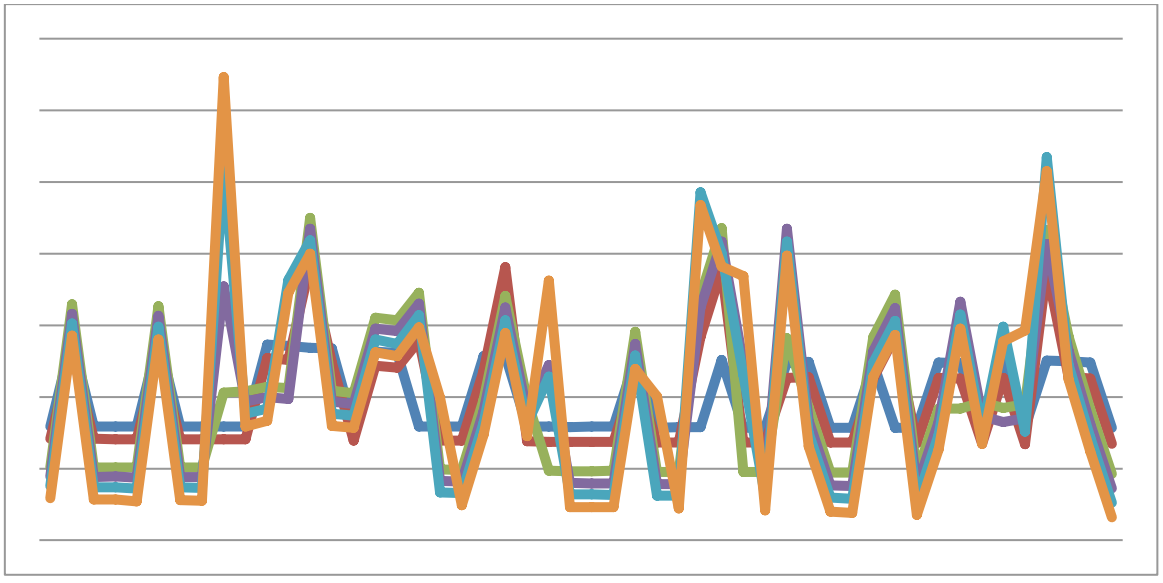


Figure 6-13 An Illustration of the Six Best Shapelets Extracted From ILD-100

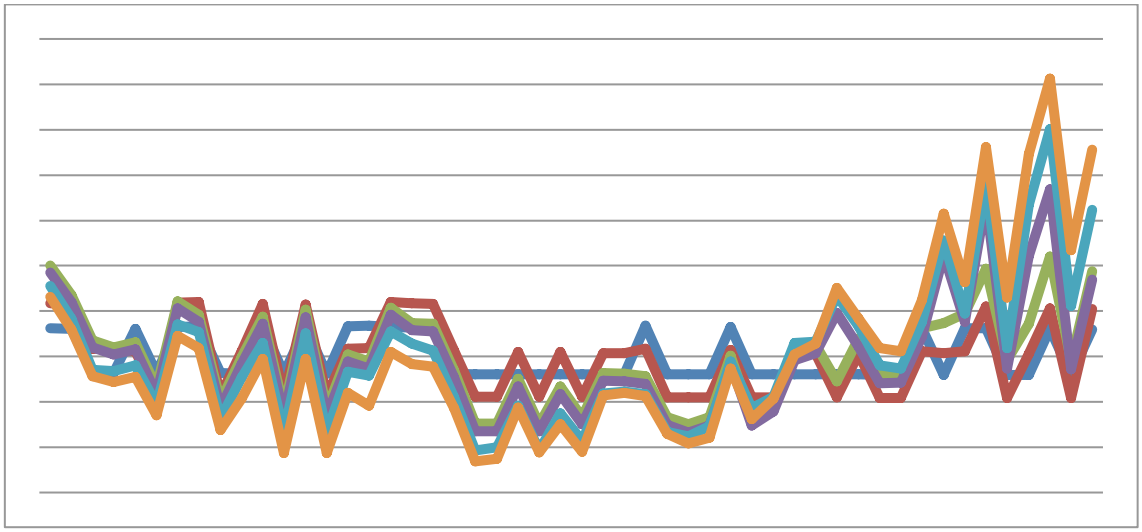


Figure 6-14 An Illustration of the Six Best Non-Matching Shapelets Extracted From ILD-100

We computed *ContextTrust* classifier data trust (Equation 1 with parameters $\beta_C = 0$ *and* $\Upsilon_C = 1$) for 8 different classifiers (see Figure 6-15) using the trained shapelets in ILD-100. We created 10 different datasets; each one containing 10 time series of 100 data points each. Each time series was drawn at random from the readings of sensors 7 through 10 from the original Intel Lab Data.

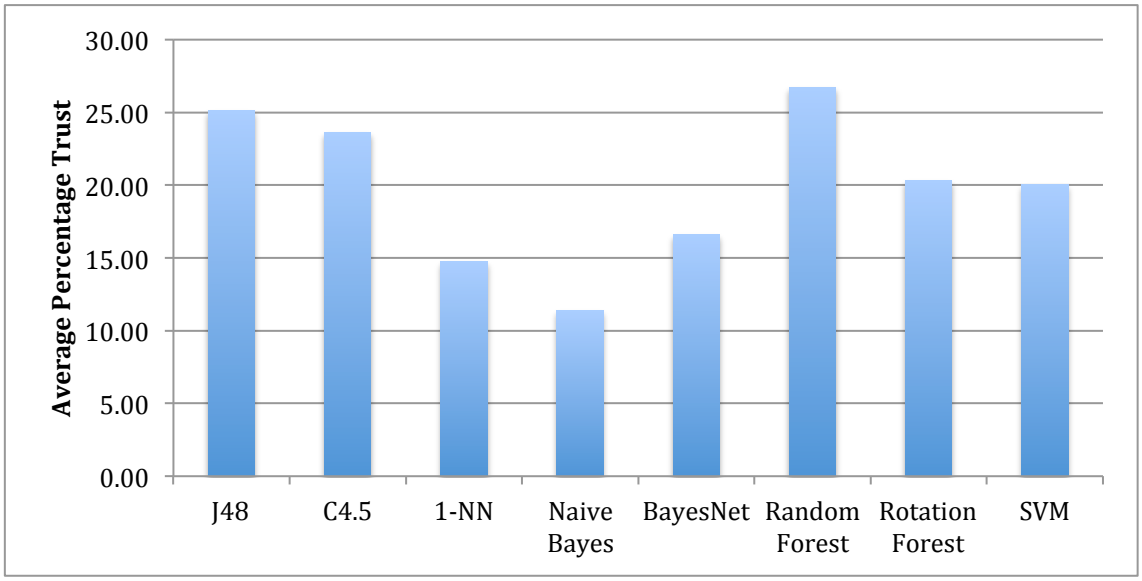


Figure 6-15 Average Percentage Data Trust Contribution for 8 Classifiers Using the IDL-100

The purpose of this experiment was to estimate how accurately our time series shapelet classification algorithm performs while classifying previously unseen temperature time series. Table 6-10 details one tailed paired t-tests using a cutoff α of 0.05 between random forest and the remaining classifiers using the data from Figure 6-15 in order to reject the null hypothesis stating "The mean percentage data trust between random forest and the remaining classifiers is the same." Using a Bonferroni adjustment and divide α by the number of comparisons we make (in this case 7). That is, for any one comparison to be considered significant, the obtained p-value would have to be less than 0.00714. As seen, there is not any statistical difference.

Table 6-9 One Tailed Paired T-Tests (α<0.05) Between Random Forest and NN, Naïve Bayes, Bayes-Net, Rotation Forest and SVM classifiers for data in Figure 6-15

J48	C45	1NN	Naïve Bayes	Bayes Net	Rotation Forest	SVM
0.0890	0.0890	0.0222	0.0152	0.0216	0.0289	0.0150

6.2 Risk Calculation

To analyze the performance of *ContextTrust*, when running on a host a *Context Security Analyzer* (see section 3.1.4), we measure the memory usage and risk calculation time. In this scenario there are four agents (context owners – see section 3.1.1) submitting their time series (one per agent) for risk calculation. Using the GeoLife data, we sampled four time series with 500 data points each.

For this simulation, we used a quad core Intel Core I7 MacBook Pro, with 16G of RAM; however, we not only restricted the execution of the risk engines to use one thread but also limit the amount of memory used in the overall process to 512 mega

bytes in order to better simulate the performance of the risk calculation in a mobile host. We also pre-generated 10000 random numbers using a Mersenne twister pseudo random number generator [Matsumoto, 98] in order to make the results reproducible.

6.2.1 Independent Factors

We implemented the algorithm in section 4.5 and varied the number of scenario trials. We kept constant the smoothing factor (0.94), confidence level (0.95) parameters since they don't affect the overall execution of the algorithm. Figure 6-16 shows the running time in milliseconds using 1000, 2000, 3000, 4000, 5000, and 10000 scenario trials. Similarly, Figure 6-17 shows the memory (Heap and Non-Heap) usage in kilobytes for the same scenario runs as Figure 6-16. Finally, Figure 6-18 shows the average accuracy of the risk calculation using a 2 percent margin of error for 1000, 2000, 3000, 4000, 5000, and 10000 scenario trials over 1,5, 10,15, 20 time units evaluation horizons. The averages were computed by trying to predict future proximity values repeating the process 50 times per scenario trial and evaluation horizon. As seen the random walk model in section 4.4 models reasonable well the GeoLife data.

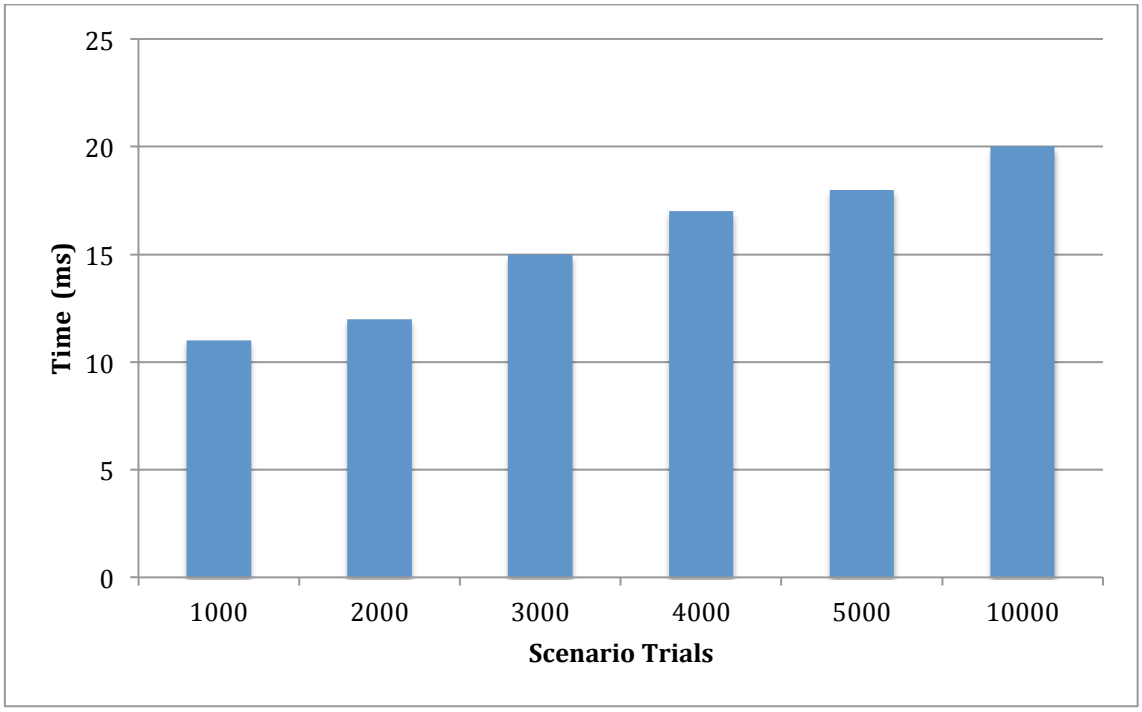


Figure 6-16 Execution Time in Milliseconds for Independent Factors Engine with Different Scenario Trials

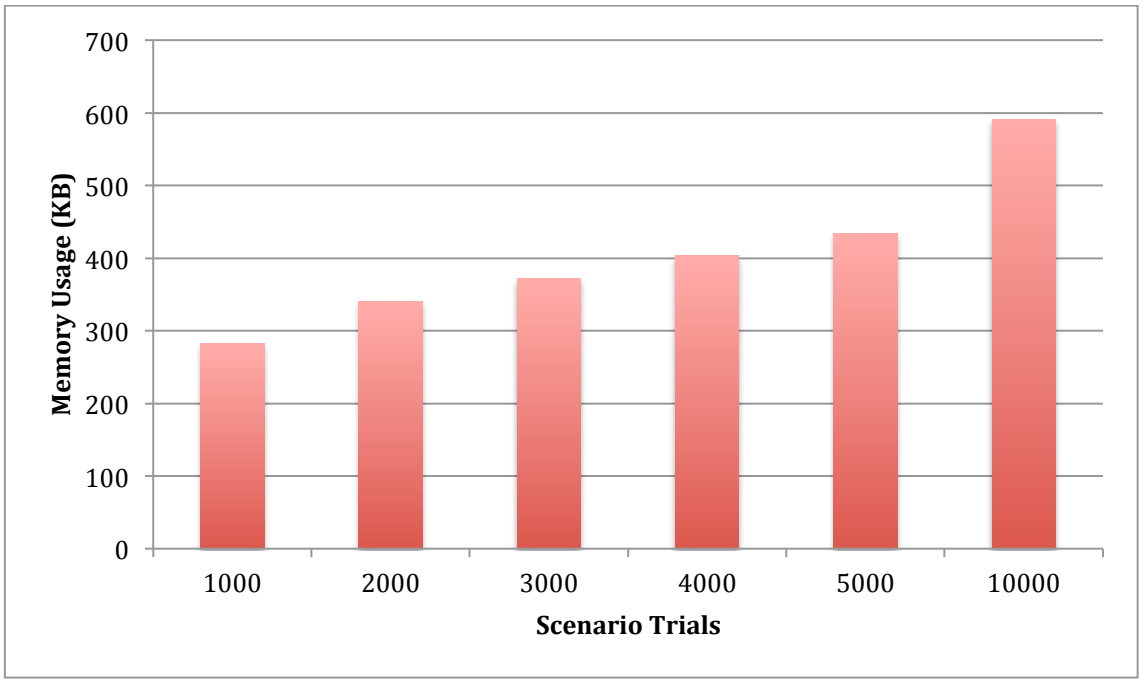


Figure 6-17 Memory Usage in KB for Independent Factors Engine With Different Scenario Trials

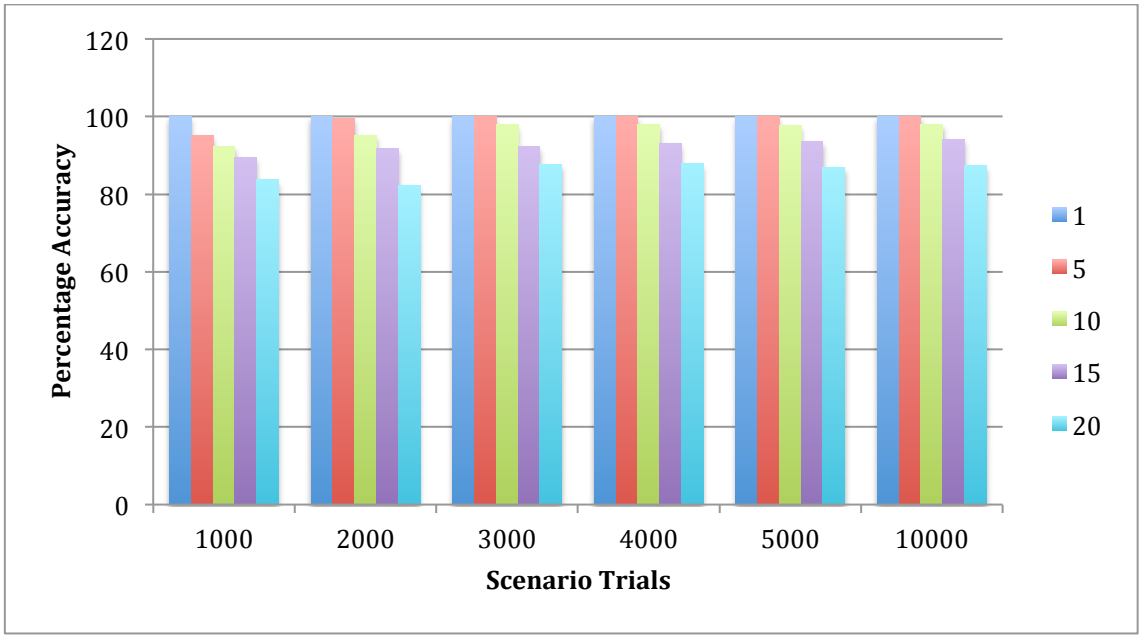


Figure 6-18 Average of Risk Calculation Accuracy Using Different Scenario Trials Over Five Different Evaluation Horizons

6.2.2 Dependent Factors

We implemented the algorithm in section 4.6.5 and in a similar manner; we varied the number of scenario trials while keeping constant the smoothing factor (0.94), and confidence level (0.95). We are able to keep both parameters constant since they don't affect the overall execution of the algorithm.

Figure 6-19 shows the running time in milliseconds for 1000, 2000, 3000, 4000, 5000, and 10000 scenario trials. Similarly, Figure 6-20 shows the memory usage in kilobytes for the same scenario runs.

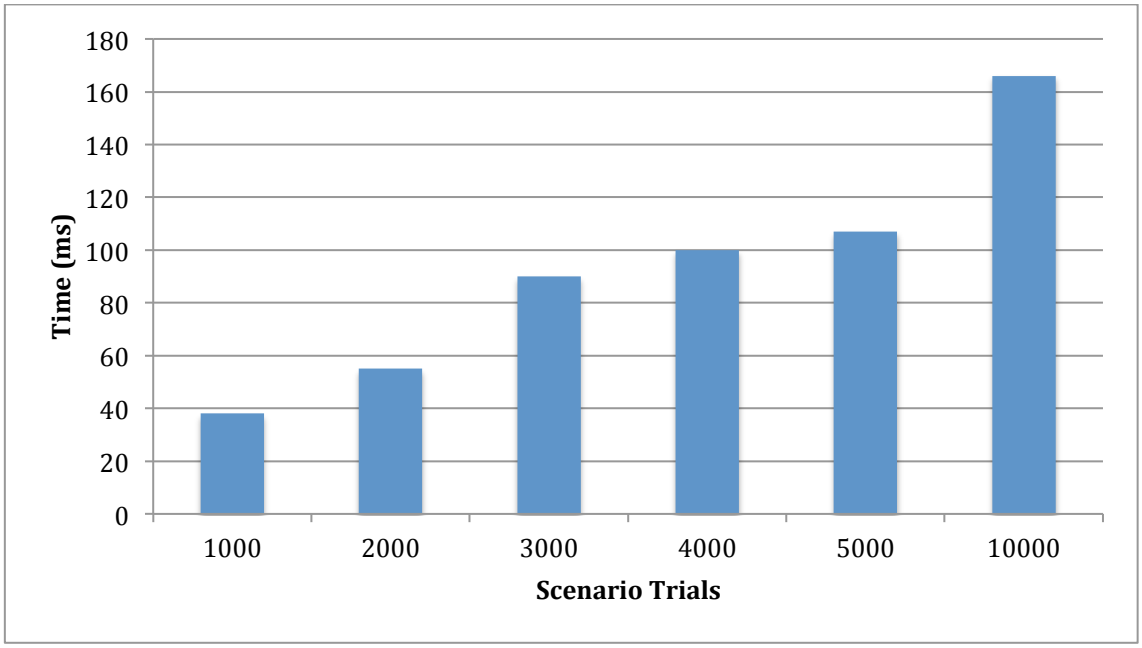


Figure 6-19 Execution Time in Milliseconds for Dependent Factors Engine with Different Scenario Trials

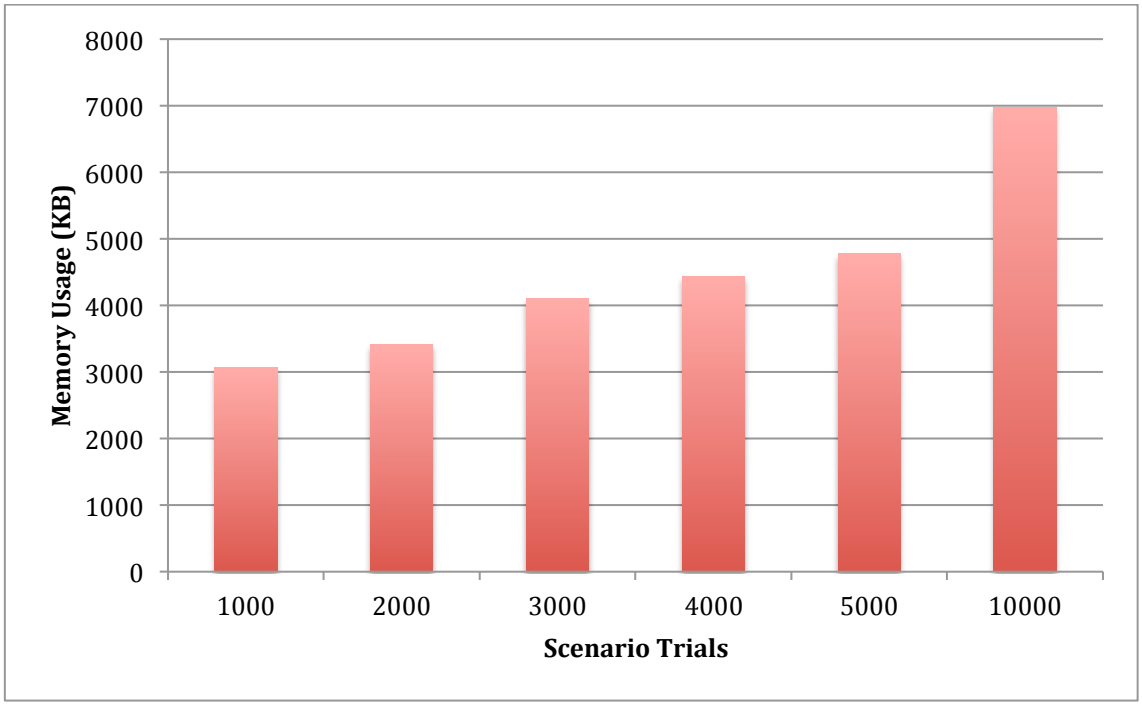


Figure 6-20 Memory Usage in KB For Dependent Factors Engine with Different Scenario Trials

In order to analyze how *ContextTrust* trust measure behaves during a time window, we implemented the equations in section 5.2. Figure 6-21 shows the trust

evolution in 50 units of time. At each unit of time a trust calculation is performed (section 5.2.4). We use a random forest classifier over and 1 to 4 recommenders using the GeoLife data. Each recommendation is the original time series "disturbed" by a small amount to simulate the difference in measurements for each recommender.

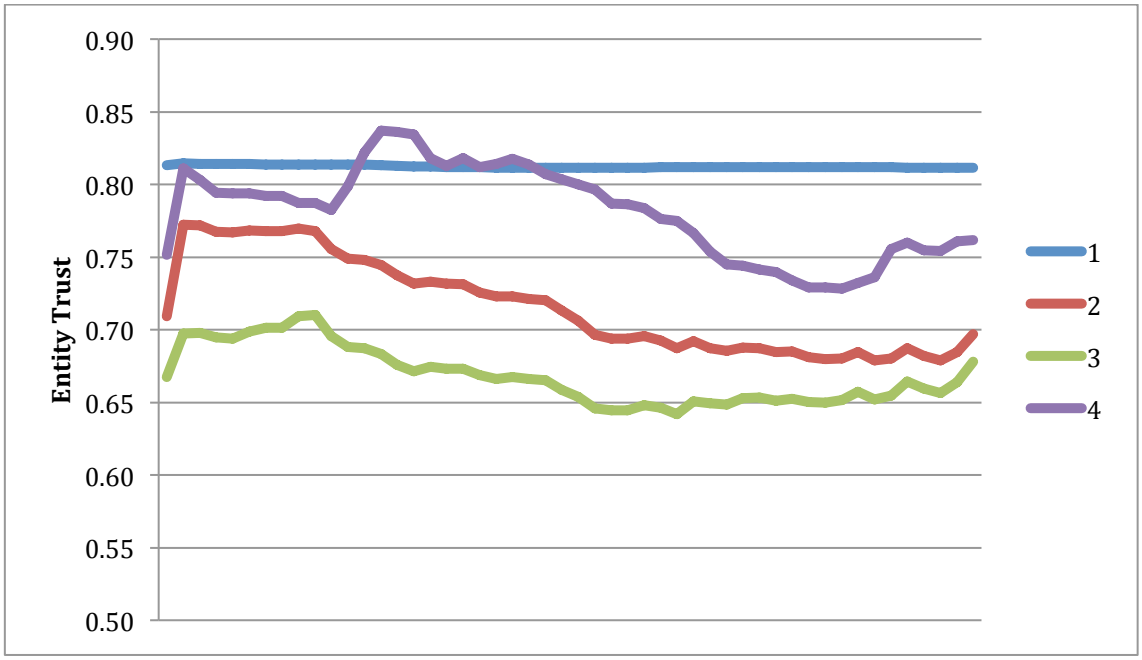


Figure 6-21 Trust Evolution of an Entity Using up to Four Recommenders Over 50 Time Units Using a Random Forest Classifier

As we can see, with one recommender the trust measure tends to be steady. This is because the disturbances to the recommended series were small. As expected, the more random disturbances in the recommended time series the more miss-classifications and negative events (that is risk calculations do not meet an error threshold) occur. Ultimately we are interested in exploring how sensitive *ContextTrust* is when computing trust values in with multiple recommenders. As seen in Figure 6-21 the more different the recommendations, the less trust is awarded. This may lead to cases where recommenders can try to mount a bad-mouthing attack against an entity by

issuing false recommendations. *ContextTrust* is designed in such a way that recommendations are trust weighted. That is, recommenders with low trust values do not contribute as much in the overall trust computation as high trust recommenders. However, *ContextTrust* may be still vulnerable bad-mouthing attack, as the system does not perform shape discord or anomaly detection (see [Wei, 06b]) on the recommended time series in order prune colluding series.

ContextTrust may need to be enhanced in order to handle some forms of Sybil attacks [Douceur, 02] and newcomer attacks [Zeckhauser, 00]. In a Sybil attack, a malicious entity may create faked time series records in order to pretend to be another one. In the newcomer attack, a malicious agent removes its bad history by registering as a new user. The defense against the Sybil attack and newcomer attack usually relies on authentication schemes [Yu, 06]. By using time series classifiers for entity authentication, *ContextTrust* attempts lessen the effects of both attacks, since such classifiers would be trained for a specific entity data.

Although not explicit, *ContextTrust* can easily adapted to handle Conflicting Behavior Attack [Wang, 10]. In this attack a malicious entity can impair other entities recommendation trust by performing differently to different contexts. For example, the attacker may always behave well to one group of users and behave poorly to another group. In order to handle such attacks, *ContextTrust* can be expanded so that when a recommendation is issued by a recommender entity, the system may ask other agents about the recommendations, the recommender entity has issued previously with regard to other [related] context variables.

Finally, by introducing Equation 5.5, *ContextTrust* may protect against on-off

attacks [Perrone, 06]. An on-off attack means a malicious entity behaves well and badly alternatively. To defend against such attack, an entity can manipulate the time effect factor in Equation 5.5 to force other entities rebuild up the trust when they cease to interact for a predetermined amount of time.

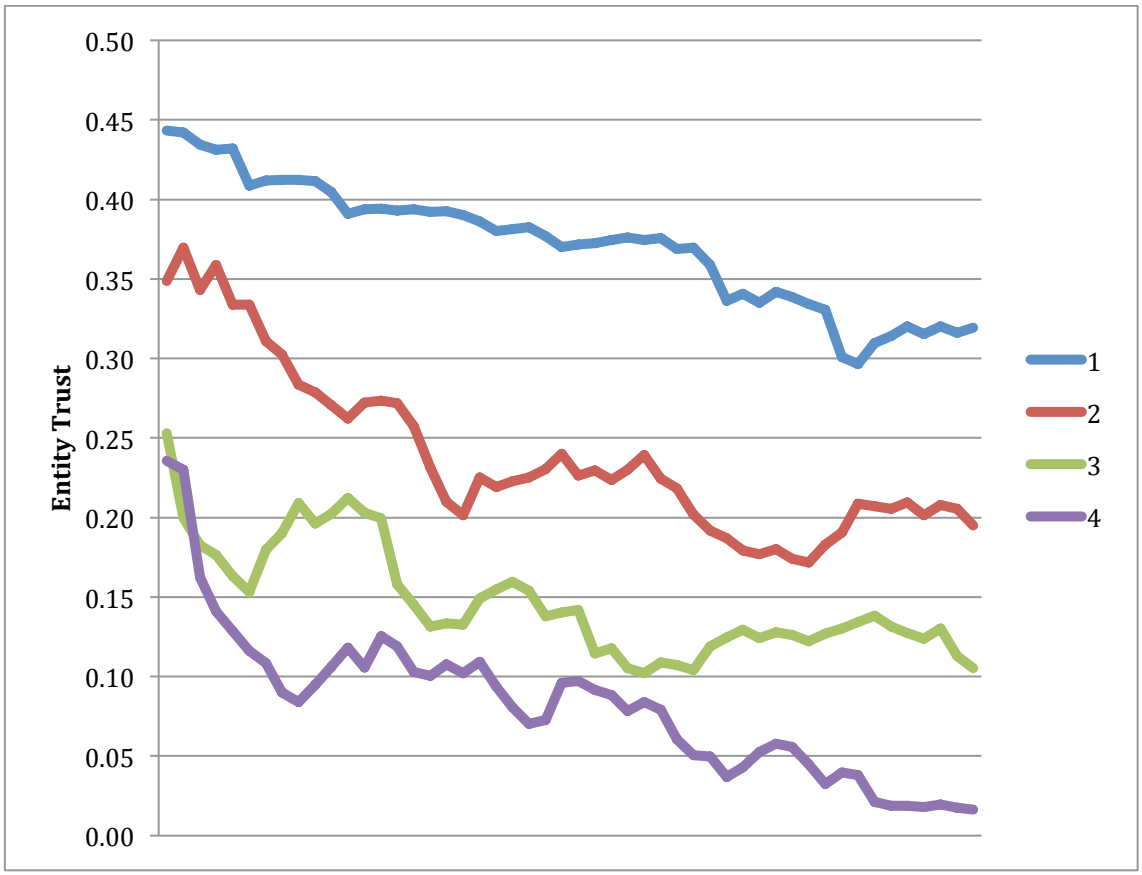


Figure 6-22 Trust Evolution of an Entity with 1 To 4 Recommenders over 50 Time Units Using A Random Forest Classifier and up to 40 Percent of Invalid Recommendations

Figure 6-22 shows the trust evolution for an entity over 50 time units using random forest classifier with and 1 to 4 recommenders using the GeoLife data. In this experiment up to 40 percent recommendations are invalid. That is, up to 40 percent of the data points in a recommendation time series are "disturbed" by a large random percent amount in order to simulate invalid/malicious recommendations. As expected, the more disturbed recommendations the less trust the entity gets. This is due to the fact

that not only the random forest classifiers yield low probabilities of classifications for the recommended time series, but also, the volatilities of the recommend time series are higher and the random walk model does not predict future values accurately.

Chapter 7. CONCLUSIONS AND FUTURE WORK

Having described a mechanism for assessing risk and trust, the purpose of this chapter is to outline both what we have achieved, and what questions remain unanswered. In doing so, we first give an overview of what we have discussed so far. Then we present a view of the contributions we have made and discuss the system limitations. Finally, we discuss possible future work.

7.1 Dissertation Summary

Risk and trust are prevalent concepts in our current society. This is particularly true in situations in which one entity needs to rely on information from another –perhaps– previously unseen entity. With ever increasing number of information sources, modern collaboration systems must not only assemble a set of disparate information systems into a coherently interoperating whole, but also make sure that the interactions amongst different entities participating in the system are secure. Moreover, these collaboration systems must take into account not only the environment in which few fixed infrastructure nodes exist, but also that access to information depends on some context information (i.e. location, velocity, etc.). Because of the lack of fixed infrastructures, the ever increasing number of users and the variability of context data, it is unreasonable to expect that every entity in a collaboration environment stores not only a reference (identity) to but also relevant context data relating to any other entity it may interact with. It is for these reasons that traditional security implementations are poorly equipped to handle security operations that include context-based access.

In response to that, we have developed a risk and trust assessment mechanism

that could be employed by entities in pervasive ad-hoc computing environments, in which contextual information is used to not only control access a resource but also compute the trustworthiness of the accessing entity. In particular, this mechanism use time series of context information in order to assess the risk emanating from the change in the context data and derive trust measures for the entities that produce it.

Before addressing these goals, in Chapter 2 we reviewed existing methods in the literature for solving these and similar problems. In addition, we briefly introduced ways of modeling context using stochastic processes in order to be used as a risk measurement tools. Moreover, we introduce data mining related concepts about time series classifications. Chapter 3 defined a context based security framework that can be used in a pervasive ad-hoc environment. Building on this, Chapter 4 introduced the Random Walk model used to determine the risk of a context variable and its application in the design security policies. Finally, in order to calculate trust, Chapter 5 introduced a framework to allow entities to reason about the reliability of others. We called this framework *ContexTrust*.

7.2 Research Contributions

The main contributions of this dissertation stem from the specification of a general framework for modeling risk and trust, and the development of *ContexTtrust.* Together, these show how, an entity can not only asses the risk of a context variable based on its changes, but also evaluate the trustworthiness of other entities it interacts with. Moreover, because *ContextTrust* uses simple statistical methods and trained classifiers in order to derive trust measurements, it makes it suitable to run on mobile devices. Furthermore, using of a random walk model for risk assessment gives us a

principled mathematical foundation to represent the behavior of a context variable. In addition, we introduced a method called space return decomposition to evaluate a context-based security policy that takes into account the relatedness (i.e. correlation) of context variables. Finally, the use of time series classification methods allows entities to reason about the truthfulness of the data being evaluated.

As part of our general framework (Chapter 3), we specified a set of guidelines for modeling context based security policies. In addition, *ContextTrust* implements simple measures to compare data from different sources (recommenders) in cases where there is a need to verify the reliability of a context risk factor. This is especially important when a number of sources provide inaccurate information. Moreover, by using time series classifier along with recommenders *ContextTrust* provides a possible solution to the problem of whitewashing, in which entities with a poor reputation attempt to improve their standing by assuming a new identity. Finally, we provide exploratory analysis of time series shapelets when they are used, as conduits for not only entity identification but also trust computation.

7.3 Limitations

There are still some open issues for which we do not provide a solution. In particular, *ContextTrust* assumes that entities are able to obtain reliably the latest values of a context time series in order to compute experience trust. In some cases, it may be desirable to relax this assumption and allow agents to enter their own values to indicate a positive or negative experience.

In addition, *ContextTrust* neither gives any guidelines for setting risk computation parameters (i.e. smoothing factor, confidence levels), nor initial trust and

weight factor values for entity trust calculation. Moreover, *ContextTrust* assumes all context factors can be model using logarithmic random walk model, which may not always be accurate. Finally, *ContextTrust* does not identify a secure mechanism to deliver time series classifiers updates from the host(s) that train the classifiers to the hosts that use them.

7.4 Future Work

In addition to the limitations mentioned above, there are a number of methods in which our mechanisms and their application could be improved in order to better aid the security decision-making in a pervasive ad-hoc system. In particular, we identify the following in which further significant research is warranted.

Initially, using probability predicates to improve trust assessment could enhance *ContexTrust*. For instance, we could introduce concepts from Beta reputation system [Jøsang, 02], Kalman filters [Kalman, 60] or Bayesian analysis (i.e. [Teacy, 06b] and [Yuan, 06]). In addition, *ContexTrust* could allow users to incorporate values (or overrides) that either replace or enhance the computation of the experience trust (that is, users can assign a subjective importance to risk events.

One assumption that we have made in *ContexTrust* is that the behavior of entities does not change over time. For many practical applications, this is an unsafe assumption, since it is possible that some entities collude with each other in order to enhance their reputation or gain access to information. Therefore, *ContextTrust* can be enhanced with the use of algorithms that can learn the behavior of entities (see [Tran, 04]). Another assumption that we made is that entities always have an appropriate amount of information to perform a risk calculation. Therefore, more investigation is

necessary in the use of fast bootstrapping models to generate initial context data points to better deal with "cold" start entities (see [Jamali, 09]).

Finally, we need more studies related to the performance and energy resource consumption of the different *ContextTrust* algorithms in mobile units. This acquires more importance as time series classification is a very expensive operation (i.e. cross validation is usually used to find the optimal shapelet size during shapelet discovery).

7.5 Conclusions

Risk and trust are increasingly important in computer science because of the current trend toward large-scale, pervasive, open, and cloud-based systems. In this dissertation, we initially considered the use of *ContextTrust* is a small military environment in which there was substantial user input in the form of context security policies. However, for other settings, we may need more autonomous agents that are be able to assess the riskiness and trustworthiness of others and make decisions based on those assessments appropriately. These agents must be able to handle situation in which they come across comes across other entities that they have not interacted with before. In this case agents would be expected to make reasonable decisions based on the data available.

A lot of work remains to be done. Hopefully this dissertation opens a set of new avenues of research in the areas of database security and data mining.

BIBLIOGRAPHY

[Aaronson, 13] Aaronson S. "Quantum Computing since Democritus." Massachusetts Institute of Technology, ISBN 9780521199568, March 2013.

[AWC, 07] Authentication World.com. "Security Tokens." Authentication World.com - Huntington Ventures Ltd, http://www.authenticationworld.com/Token-Authentication/, accessed September 2007.

[Bagnall, 12] Bagnall A., Lines J. "Alternative Quality Measures for Time Series Shapelets." Lecture Notes in Computer Science Volume 7435, pp. 475-483, 2012.

[Bacon, 02] Bacon J., Moody K., Yao W. "A Model of OASIS Role-Based Access Control and its Support for Active Security." ACM Transactions on Information and System Security (TISSEC), Volume 5, Number 4, pp. 492-540, November 2002.

[Benson, 97] Benson P., Zangari P. "A General Approach to Calculating VaR without Volatilities and Correlations." JPMorgan/Reuters RiskMetrics Monitor, Second Quarter, 1997.

[Bardram, 03] Bardram J., Kjaer R., Pedersen M. "Context Aware User Authentication – Supporting Proximity-Based Login in Pervasive Computing." Fifth International Conference on Ubiquitous Computing, UBICOMP 2003, LNCS 2864, Volume 2864, pp.107-123, October 2003.

[Bell, 73] Bell D., LaPadula L. "Secure Computer Systems: Mathematical Foundations and Model." Hanscom AFB, Bedford, MA, Rep. FSD-TR- 73-278, Volume 1, ESD/AFSC, 1973.

[Bertino, 03] Bertino E., Correndo G., Ferrari E., Mella G. "An Infrastructure for Managing Secure Update Operations on XML Data". 8th ACM Symposium on Access Control Models and Technologies, SACMAT'03, pp. 110 –122, June 2003.

[Bertino, 05] Bertino E., Joshi J., Latif U., Ghafoor Arif. "A Generalized Temporal Role-Based Access Control Model." IEEE Transactions on Knowledge and Data Engineering, Volume 17, Number 1, pp. 4-23, January 2005.

[Biba, 77] Biba K. “Integrity Considerations for Secure Computer Systems.” MTR-3153, The Mitre Corporation, April 1977.

[Bierman, 02] Bierman E., Cloete E. ”Classification of Malicious Host Threats in Mobile Agent Computing.” Proceedings Of The 2002 Annual Research Conference of The South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology, SAICSIT, pp. 141-148, 2002.

[Bodik, 04] Bodik P., Hong W., Guestrin C., Madden S., Paskin M., Thibaux R. ”Intel Lab Data.” http://db.csail.mit.edu/labdata/labdata.html, accessed February 2013.

[Boeing, 07] “ScanEagle.” The Boeing Corporation, All rights reserved, http://www.boeing.com/defense-space/military/scaneagle/index.html, accessed November 2007.

[Bollerslev, 86] Bollerslev T. et al. "Arch models." Elsevier, Handbook of Econometrics, First Edition, Volume 4, Chapter 49, pp. 2959-3038, 1986.

[Bouganim, 05] Bouganim L., Cremarenco C., Dang-Ngoc F., Dieu N., Pucheral P. “Safe Data Sharing and Data Dissemination on Smart Devices.” Proceedings of the 2005 ACM SIGMOD International conference on Management of data, SIGMOD '05, pp. 888–890, 2005.

[Brezillon, 04] Brezillon P., Mostefaoui G.K. “Context-Based Security Policies: A New Modeling Approach.” Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, PERCOMW’04, pp. 154-158, March 2004.

[Bryce, 05] Bryce C., Dimmock N., Krukow K., Seigneur J.M., Cadhill V., Wagealla W. “Towards an Evaluation Methodology for Computational Trust Systems.” Third International Conference on Trust Management, iTrust 2005, LNCS 3477, pp. 289-304, May 2005.

[BPM, 06] Bureau International des Poids et Mesures. “The International System of Units (SI) Brochure, 8th Edition.” International Committee for Weights and Measures, 2006.

[Cahill, 03] Cahill V., Dimmock N., Wagealla W., et al. “Using Trust for Secure Collaboration in Uncertain Environments.” IEEE Pervasive Computing, Volume 2, Number 3, pp. 52-61, July-September 2003.

[Cai, 05] Cai L., LU Y.H. “Joint Power Management of Memory and Disk.” Proceedings of the conference on Design, Automation and Test in Europe - Volume 1, pp. 86-91, March 2005.

[Capra, 06] Capra L., Musolesi M. “Autonomic Trust Prediction for Pervasive Systems.” Proceedings of the 20th International Conference on Advanced Information Networking and Applications, Volume 2, AINA'06, pp. 481-488, 2006.

[Camenisch, 05] Camenisch J., Shelat A, Sommer D., et al. “Privacy and Identity Management for Everyone.” Proceedings of The 2005 Workshop on Digital Identity Management, DIM '05, pp. 20–27, November 2005.

[Carvalho, 04] Carvalho M., Cowin T., Suri N., Breedy M., Ford K. “Using Mobile Agents as Roaming Security Guards to Test and Improve Security of Hosts and Networks.” ACM Symposium on Applied Computing, SAC 2004, March 2004.

[Carbone, 03] Carbone M., Nielsen M., Sassone V. “A Formal Model for Trust in Dynamic Networks.” Basic Research in Computer Science (BRICS), Technical Report RS-03-4, University of Aarhus, January 2003.

[Castelli, 07] Castelli G., Rosi A., Mamei M., Zambonelli F. “A Simple Model and Infrastructure for Context-Aware Browsing of the World.” Fifth Annual IEEE International Conference on Pervasive Computing and Communications PerCom'07, pp. 229-238, March 2007.

[Chan, 03] Chan D., Roddick J. “Context-Sensitive Mobile Database Summarization.” Proceedings of the twenty-sixth Australasian Computer Science Conference on Conference in Research and Practice in Information Technology, Volume 16, CRIPTS '03, pp. 139-149, February 2003.

[Chakraborty, 06] Chakraborty S., Ray I. “TrustBAC – Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems.” 11th ACM Symposium on Access Control Models and Technologies, SACMAT’06, pp. 49-58, June 2006.

[Chatzigiannakis, 04] Chatzigiannakis I., Kinalis A., Kinalis S. "Wireless Sensor Networks Protocols for Efficient Collision Avoidance in Multi-Path Data Propagation." Proceedings of the 1st ACM International Workshop on Performance Evaluation of Wireless Ad-Hoc, Sensor, and Ubiquitous Networks, pp. 8-16, October 2004.

[Chen, 00] Chen G., Kotz D. "A Survey of Context-Aware Mobile Computing Research." Dartmouth Computer Science Technical Report TR2000-381, ftp://ftp.cs.dartmouth.edu/TR/TR2000-381.pdf, 2000.

[Cheng, 03] Cheng R., Prabhakar S. "Managing Uncertainty in Sensor Databases." ACM SIGMOD Record, Volume 32, Number 4, pp. 41-46, December 2003.

[Chong, 03] Chong C-Y., Kumar S. P. "Sensor Networks: Evolution, Opportunities and Challenges." Proceedings of the IEEE, Volume 91, Number 8, pp. 1247- 1256, August 2003.

[Chou, 75] Chou Y.L. "Statistical Analysis, with business and economic applications." Holt, Rinehart & Winston of Canada Ltd, ISBN 0030894220, 1975.

[Clark, 99] Clark J., DeRose S. (eds). "XML Path Language (XPath) Version 1.0." W3C Recommendation. http://www.w3c.org/TR/xpath, November 1999.

[Clauβ, 05] Clauβ S., Kesdogan D., Kölsch T. "Privacy Enhancing Identity Management: Protection Against Re-identification and Profiling." Proceedings of the 2005 workshop on Digital identity management, DIM '05, pp. 84-93, November 2005.

[Cohen, 02] Cohen E., Thomas R.K. "Models for Coalition-based Access Control (CBAC)." 7th ACM Symposium on Access Control Models and Technologies, SACMAT'02, pp. 97-106, June 2002.

[Cox, 95] Cox, J.C. Ingersoll, J.E., Ross, S.A. "A Theory of the Term Structure of Interest Rates." Econometrica, Volume 53, Number 2, pp. 385-408, March 1985.

[Crossbow, 13] Crossbow Technology. "MICA2DOT Wireless Micro-Sensor Mote." Wireless Sensor Networks, https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf, accessed May 2013.

[Dawn, 05] Dawn N. J., Bodorik P. "Sociotechnical Architecture for Online Privacy." IEEE Security & Privacy, Volume 3, Number 2, pp. 29-39, March-April 2005.

[Davidian, 07] Davidian M. "Random vectors and multivariate normal distribution." Lecture Notes for ST 732, Applied Longitudinal Data Analysis, North Carolina State University, http://www.stat.ncsu.edu/people/davidian/courses/st732/, 2007.

[Debaty, 05] Debaty P., Goddi P., Vorbau A. "Integrating the Physical World with the Web to Enable Context-Enhanced Mobile Services." ACM Mobile Network and Applications, Volume 10, Number 4, pp. 385-394, August 2005.

[Diaz, 03] Diaz P., Ribeiro C., Ferreira P. "Enforcing History-Based Security Policies in Mobile Agent." Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks, POLICY'03, June 2003.

[Dimmock, 04] Dimmock N., Belokosztolszki A., Eyers D. "Using Trust and Risk in Role-Based Access Control Policies." Proceedings of the ninth ACM symposium on Access control models and technologies, pp. 156-162, June 2004.

[Dimmock, 05] Dimmock N., Bacon J., Ingram D., Moody K. "Risk Models for Trust Based Access Controls (TBAC)." Third International Conference on Trust Management, iTrust 2005, LNCS 3477, pp. 364-371, May 2005.

[Ding, 08] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E. "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures." Proceedings of the 34th International Conference on Very Large Data Bases (VLDB), Volume 1, Number 2, pp.1542-1552, August 2008.

[Dion, 81] Dion L. "A Complete Protection Model." Proceedings of the IEEE Symposium on Security and Privacy, pp. 49-55, April 1981.

[Dittrich, 94] Dittrich K., Jonscher D. "An Approach for Building Secure Database Federations." Proceedings of the 20th International Conference on Very Large Data Bases table of contents, pp. 24-35, ISBN: 1-55860-153-8, September 1994.

[Dittrich, 94b] Dittrich K., Jonscher D. "Current Trends in Database Technology and Their Impact on Security Concepts." Proceedings of the IFIP WG11.3 Working Conference on Database Security VII, pp. 11-33, August 1994.

[Dittrich, 95] Dittrich K., Jonscher D. "Argos – A Configurable Access Control System for Interoperable Environments." Proceedings of the IFIP WG 11.3 9th Annual Working Conference on Database Security, pp. 43-60, August 1995.

[Douceur, 02] Douceur J. R., "The Sybil Attack." Proceedings of First International Workshop on Peer-to-Peer systems, IPTPS'02, 2002.

[English, 04] English C., Terzis S., Wagealla W. "Engineering Trust Based Collaborations in a Global Computing Environment." Second International Conference on Trust Management, iTrust 2004, LNCS 2995, pp. 120-134, April 2004.

[Essmayr, 95] Essmayr W, Kastner F, Preishuber S., Tjoa A. "Access Controls for Federated Database Environments: Taxonomy of Design Choices." Proceedings of the Joint IFIP TC6 and TC 11 Working Conference on Communications and Multimedia Security, Graz, Austria, September 1995.

[FAS, 07] FAS Military Network Analysis. "Land Warrior." Federation of American Scientists, http://www.fas.org/man/dod-101/sys/land/land-warrior.htm, accessed January 2007.

[FCC, 06] Federal Communication Commission FCC. "Enhanced 911 - Wireless Services." http://www.fcc.gov/911/enhanced/, accessed February 2006.

[Ferraiolo, 01] Ferraiolo D., Sandhu R., Gavrila S., Kuhn R., Chandramouli R. "Proposed NIST Standard for Role-Based Access Control." ACM Transactions on Information and System Security Volume 4, Number 3, pp. 224-274, August 2001.

[FIDIS, 05] FIDIS "Study on Mobile Identity Management." Future of Identity in the Information Society, http://www.fidis.net/486.0.html, accessed May 2005.

[Finch, 04] Finch S. R. "Ornstein-Uhlenbeck Process." The French National Institute for Research in Computer Science (INRIA), Algorithms Project, unpublished note, http://algo.inria.fr/csolve/ou.pdf, May 2004.

[Frank, 00] Frank E., "Pruning Decision Trees and Lists." PhD Thesis, Department of Computer Science, University of Waikato, New Zealand, January 2000.

[Fundulaki, 04] Fundulaki I., Marx M. “Specifying Access Control Policies for XML Documents with XPath.” Ninth ACM Symposium on Access Control Models and Technologies, SACMAT’04, pp. 61-69, June 2004.

[Galluccio, 04] Galluccio L., Morabito G., Palazzo S. “Spontaneous Group Management in Mobile Ad Hoc Networks.” ACM Wireless Networks Volume 10, Number 4, pp. 423-438, July 2004.

[Gaye, 03] Gaye L., Mazé R., Holmquist L. E. “Sonic City: The Urban Environment as a Musical Interface.” Proceedings of the 2003 conference on New interfaces for musical expression, pp. 109–115, May 2003.

[Geurts, 01] Geurts P. “Pattern Extraction for Time Series Classification.” Principles of Data Mining and Knowledge Discovery, LNCS 2168, pp. 115-127, 2001.

[Golbeck, 03] Golbeck J., Hendler J., Parsia B. “Trust Networks on the Semantic Web.” Proceedings of Cooperative Intelligent Agents, CIA03, pp.238-249, August 2003.

[Golbeck, 96] Golub G. H., Van Loan C.F. “ Matrix Computations (3rd ed.).” Johns Hopkins University Press, ISBN 0801854148, pp. 140-152, 1996.

[Goldreich, 03] Goldreich O. “Cryptography and Cryptographic Protocols.” Journal of Distributed Computing - Papers in celebration of the 20th anniversary of PODC, Volume 16, Issues 2-3, pp. 177-199, September 2003.

[Grinstead, 97] Grinstead, C.M, Snell, J.L. "Introduction to Probability." American Mathematical Society, 2nd revised edition, ISBN-13: 978-0821807491, Chapter 7, July 1997.

[GUIDE, 05] GUIDE Consortium. “Deliverable: D1.2.1.B – Identity Interoperability Services Report: Core Services Description.” GUIDE Consortium, http://istrg.som.surrey.ac.uk/projects/guide/documents.html, accessed September 2005.

[Hagen, 04] Hagen K. “Path Integrals in Quantum Mechanics, Statistics, Polymer Physics, and Financial Markets 4th edition.” World Scientific Publishing Company, ISBN 9812381074, 2004.

[Hansson, 07] Hansson S.O. "Risk." The Stanford Encyclopedia of Philosophy, Summer 2007 Edition, http://plato.stanford.edu/archives/sum2007/entries/risk, accessed August 2007.

[Hamilton, 94] Hamilton J.D. "Time Series Analysis." Princeton University Press, ISBN 0691042896, 1994.

[Hillman, 05] Hillman D. "Using Mobilize Power Management IP for Dynamic & Static Power Reduction in SoC at 130 nm." Proceedings of the Conference on Design, Automation and Test in Europe, Volume 3, pp. 240-246, March 2005.

[Hughes, 96] Hughes B.D. "Random Walks And Random Environments." Oxford University Press, ISBN 0-19-853789-1, 1996.

[Hall, 09] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann Witten I. "The WEKA Data Mining Software: An Update." SIGKDD Explorations, Volume 11, Number 1, 2009.

[Hull, 99] Hull J.C. "Options, Futures & Other Derivatives 4th edition." Prentice Hall, ISBN 0130224448, 1999.

[Hulsebosch, 04] Hulsebosch, B., Salden, A., Bargh, M. "Context-Based Service Access for Train Travelers" Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI), LNCS 3295, pp. 84-87, October 2004.

[Hulsebosch, 05] Hulsebosch R., Salden A., Bargh M., Ebben P., Reitsma J." Context Sensitive Access Control." ACM Symposium on Access Control Models and Technologies, pp. 111-119, June 2005.

[IDABC, 04] IDABC. "IDA Authentication Policy." Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens, EU Directorate General Enterprise, http://europa.eu.int/idabc/en/document/3532/5585, accessed July 2004.

[IEEE802, 07] IEEE802 "IEEE 802.11™ WIRELESS LOCAL AREA NETWORKS - The Working Group for WLAN Standards." http://www.ieee802.org/11/, accessed September 2007.

[Jain, 00] Jain A., Bolle R.M., Pankanti S. “Biometrics—The Future of Identification.” IEEE Computer, Volume 33, Number 3, pp. 46-49, February 2000.

[Jensen, 00] Jensen D., Cohen P. “Multiple Comparisons in Induction Algorithms.” Machine Learning, Volume 38, Number 3, pp. 309-338, March 2000

[Jamali, 09] Jamali M., Ester M. “TrustWalker: A Random Walk Model for Combining Trust-based and Item-based Recommendation.” Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD’09, 2009.

[Jøsang, 02] Jøsang A., Ismail R. “The Beta Reputation System.” Proceedings of the 15th Bled Electronic Commerce Conference, 2002.

[Jøsang, 04] Jøsang A., Presti S.L. “Analyzing the Relationship Between Risk and Trust.” Second International Conference on Trust Management, iTrust 2004, LNCS 2995, pp. 135-145, April 2004.

[Juang, 03] Juang W.S., Lei C.L., Liaw H.T. "Privacy and Anonymity Protection with Blind Threshold Signatures." International Journal of Electronic Commerce, Volume 7, Number 2, pp. 143–157, 2003.

[Jugl, 00] Jugl E., Boche H. “Analysis of Analytical Mobility Models with Respect to the Applicability for Handover Modeling and to the Estimation of Signaling Cost.” Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 68-75, August 2000.

[Kagal, 01] Kagal L., Finin T., Joshi A. “Trust-Based Security in Pervasive Computing Environments.” IEEE Computer Volume 34, Number 12, pp. 154-157, December 2001.

[Kalman, 60] Kalman R. E. “A New Approach to Linear Filtering and Prediction Problems.” Transactions of ASME – Journal of Basic Engineering, Volume 82, Number Series D, pp. 35-45, 1960.

[Kapadia, 04] Kapadia A., Sampemane G., Campbell R.H. “KNOW Why Your Access Was Denied: Regulating Feedback for Usable Security.” Proceedings of the 11th ACM Conference on Computer and communications, pp. 52–61, October 2004.

[Kemp, 13] Kemp J. “Normal Distributions and Scales.” Wikimedia Foundation, Inc., http://en.wikipedia.org/wiki/File:PR_and_NCE.gif, accessed May 2013

[Kenney, 62] Kenney, J. F. and Keeping, E. S. “Moving Averages.” Mathematics of Statistics, 3rd Edition, Van Nostrand Publishers, pp. 221-223, ASIN: B0007HR7SY, 1962.

[Keoh, 02] Keoh S. L., Lupu E. “Security and Middleware Services: Towards Flexible Credential Verification in Mobile Ad-Hoc Networks.” Proceedings of the Second ACM International Workshop on Principles of Mobile Computing, pp. 58-65, October 2002.

[Keogh, 01] Keogh E., Pazzani M. “Derivative Dynamic Time Warping.” Proceedings of the First SIAM International Conference on Data Mining, SDM’2001, 2001.

[Keogh, 04] Keogh E., Ratanamahatana C. “Exact Indexing of Dynamic Time Warping.” Knowledge and Information Systems, Volume 7, Number 3, pp. 358-386, March 2005.

[Kim, 01] Kim J., Mina J. “RiskGrades Technical Document Second Edition.” RiskMetrics Group, February 2001.

[Kinateder, 03] Kinateder M., Rothermel K. “Architecture and Algorithms for a Distributed Reputation System.” First International Conference on Trust Management, iTrust 2003, LNCS 2692, pp. 1-16, February 2003.

[Kinateder, 05] Kinateder M., Baschny E., Rothermel K. “Towards a Generic Trust Model – Comparison of Various Trust Update Algorithms.” Third International Conference on Trust Management, iTrust 2005, LNCS 3477, pp. 177-192, May 2005.

[Klefstad, 06] Klefstad R., Lin K.J., Zhang Y. “DIRECT: A Robust Distributed Broker Framework for Trust and Reputation Management.” The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce and E-Services, CEC/EEE'06, pp. 21-28, June 2006.

[Korpipää, 03] Korpipää P., Mäntyjärvi J., et al. “Managing Context Information in Mobile Devices.” IEEE Pervasive Computing Volume 2, Number 3, pp. 42-51, July-September 2003.

[Krintz, 04] Krintz C., Wen Y., Wolski R. “Application-Level Prediction of Battery Dissipation.” Proceedings of the 2004 International Symposium on Low Power Electronics and Design, pp. 224-229, August 2004.

[Kuntze, 06] Kuntze N., Schmidt A.U. “Transitive Trust in Mobile Scenarios.” Proceedings of the International Conference on Emerging Trends in Information and Communication Security, ETRICS 2006, pp.73-85, June 2006.

[Langeheinrich, 01] Langeheinrich M., “Privacy by Design – Principles of Privacy Aware Ubiquitous Systems.” Proceedings of UBICOMP 2001, LNCS 2201, pp. 273-291, October 2001.

[Li, 07] Li H., Mukesh S. “Trust Management in Distributed Systems.” IEEE Computer Volume 40, Number 2, pp. 45-53, February 2007.

[Li, 13] Li Z. “Time Series Shapelets: A New Primitive for Data Mining – PowerPoint Presentation.” https://wiki.engr.illinois.edu/download/attachments/186384416/timeseries.ppt, accessed April 2013.

[Limpert, 01] Limpert E., Stahel W., Abbt M. “Log-normal Distributions across the Sciences: Keys and Clues.” BioScience, Volume 51, Number 5, pp. 341–352, 2001.

[Lin, 05] Lin C., Varadharajan V., Wang Y., Pruthi V. “Trust Enhanced Security for Mobile Agents.” Proceedings of the 7th IEEE International Conference on Ecommerce Technology, CEC’05, pp. 231-238, July 2005.

[Lines, 12] Lines J., Hills J., Luke M., Bagnall A., Batista G. “A Shapelet Transform for Time Series Classification.” The 18th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 2012.

[Lu, 05] Lu H., Lin K.J., Yu T., Tai C.E. “A Reputation and Trust Management Broker Framework for Web Applications.” IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE'05, pp. 262-269, March-April 2005.

[Martin, 05] Martin J., Ignas N. “Privacy and Anonymity in Personal Networks.” Third IEEE International Conference on Pervasive Computing and Communications Workshops PERCOMW'05, pp. 130-135, March 2005.

[Massa, 04] Massa P., Bhattacharjee B. “Using Trust in Recommender Systems: An Experimental Analysis.” Second International Conference on Trust Management, iTrust 2004, LNCS 2995, pp. 221-235, April 2004.

[MATIS, 06] MATIS Geneva Team - Database Laboratory. “BNF rules of JAVA.” University of Geneva, http://cui.unige.ch/dbesearch/Enseignement/analyseinfo/JAVA/, accessed May 2006.

[Matsumoto, 98] Matsumoto. M., Nishimura T. "Mersenne Twister: A 23-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator." ACM Transactions on Modeling and Computer Simulation, Volume 8, Number 1, pp. 3–30, 1998

[McKnight, 02] McKnight, D.H. et al. “Developing and Validating Trust Measures for ecommerce: An Integrative Typology.” Information Systems Research, Volume 13, Number 3, pp. 334-359. September 2002.

[Mendenhall, 07] Mendenhall W., Scheaffer R. L., Wackerly D. “Mathematical Statistics with Applications 7th Edition.” Cengage Learning Publishers, ISBN 0495385085, October 2007.

[Merino, 05] Merino A.S., Matsunaga Y., Shah M., Suzuki T., Katz R.H. “Secure Authentication System for Public WLAN Roaming.” ACM Mobile Networks and Applications, Volume 10 Number 3, pp. 355-370, June 2005.

[Minami, 05] Minami K, Kotz D. “Secure Context-sensitive Authorization.” Proceedings of the Third IEEE Annual Conference on Pervasive Computing and Communications Workshop, PERCOMWY05, pp. 257-268, March, 2005.

[Monk, 13] MONK Project. “Analytics: Decision Tree Induction.” Metadata Offer New Knowledge, Andrew W. Mellon Foundation and CIC Library Directors, http://monkpublic.library.illinois.edu/monkmiddleware/public/analytics/decisiontree.html, access February 2013.

[Mostefaoui, 04] Mostefaoui G.K., Brezillon P. “Modeling Context-Based Security Policies with Contextual Graphs.” Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 28-33, March 2004.

[Moyer, 00] Moyer M.J., Covington M.J., Ahmand M. "Generalized Role-Based Access Control for Securing Future Applications." Proceedings of the 23rd National Information Systems Security Conference, NISSC 2000, October 2000.

[Muen, 11] Mueen A. Keogh E., Young N. "Logical-Shapelets: An Expressive Primitive for Time Series Classification." The 17th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 2011.

[Murthy, 98] Murthy S. K. "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey." Data Mining and Knowledge Discovery, Volume 2, Issue 4, pp.345-389, 1998.

[Müller, 07] Müller M. "Information Retrieval for Music and Motion." Springer, ISBN 9783540740483, 2007

[MWD, 07] Merriam-Webster Dictionary. "Honesty." Merriam-Webster, Inc., http://www.merriam-webster.com/dictionary/honesty, accessed September 2007.

[MWD, 10] Merriam-Webster Dictionary. "Temperature." Merriam-Webster, Inc., http://www.merriam-webster.com/netdict/temperature, accessed February 2010.

[MWD, 13] Merriam-Webster Dictionary. "Proximity." Merriam-Webster, Inc. http://www.merriam-webster.com/dictionary/proximity, accessed May 2013.

[MWD, 13a] Merriam-Webster Dictionary. "Location." Merriam-Webster, Inc. http://www.merriam-webster.com/dictionary/location, accessed May 2013.

[Nanopoulos, 01] Nanopoulos, A., Alcock, R., Manolopoulos, Y. "Feature-based Classification of Time-series Data." International Journal of Computer Research, Volume 10, pp. 49-61, 2001.

[Neumann, 01] Neumann G., Strembeck M. "Access Control: Design and Implementation of a Flexible RBAC-Service in an Object-Oriented Scripting Language." Proceedings of the 8th ACM conference on Computer and Communications Security, pp.58-67, November 2001.

[Nielsen, 03] Nielsen M., Carbone M., Sassone V. “A Formal Model for Trust in Dynamic Networks.” Basic Research in Computer Science (BRICS) Technical Report RS-03-4, http://www.brics.dk/RS/03/4/BRICS-RS-03-4.pdf, University of Aarhus, 2003.

[Nixon, 04] Nixon P.A., Wagealla W., English C., Terzis S. “Security, Privacy and Trust Issues in Smart Environments.” University of Strathclyde, Computer and Information Sciences, Smartlab Technical Report Smartlab-2004-01, 2004.

[OASIS, 05a] OASIS. ”eXtensible Access Control Markup Language (XACML) *TC.”* Copyright© OASIS Open, http://www.oasis-open.org/, accessed October 2005.

[OASIS, 05b] OASIS. “Security Assertion Markup Language (SAML) v2.0.” Copyright© OASIS Open, http://www.oasis-open.org/, accessed October 2005.

[Oppliger, 05] Oppliger R., Rytz R. “Does Trusted Computing Remedy Computer Security Problems?” IEEE Security & Privacy, Volume 3, Number 2, pp.16-19, March-April 2005.

[Page, 04] Page J., Zaslavsky A., Indrawan M. “A Buddy Model of Security for Mobile Agent Communities Operating in Pervasive Scenarios.” Proceedings of the 2nd workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalization - Volume 32 CRPIT '04, pp. 17-25, January 2004.

[Papoulis, 84] Papoulis, A. "Brownian Movement and Markoff Processes." Ch. 15 in Probability, Random Variables, and Stochastic Processes, 2nd Edition, New York: McGraw-Hill, pp. 515-553, 1984.

[Pafka, 01] Pafka S., Kondor I. “Evaluating the RiskMetrics Methodology in Measuring Volatility and Value-at-Risk in Financial Markets.” Physica A: Statistical Mechanics and its Applications, Volume 299, Issues 1-2, pp. 305-310, October 2001.

[Park, 04] Park J.S, Costello K.P. Neven T.M., Diosomito J.A. “Access Management for Distributed Systems: A composite RBAC approach for large, complex organizations.” ACM Symposium on Access Control Models and Technologies, pp. 163-172, June 2004.

[Patel, 05] Patel J., et al. “A Probabilistic Approach Trust Model for Handling Inaccurate Reputation Sources.” Third International Conference on Trust Management, iTrust 2005, LNCS 3477, pp. 193-209, May 2005.

[Pernul, 94] Pernul G. “Database Security.” Advances in Computers, Volume 38, pp. 1-72, Academic Press, 1994.

[Perrone, 06] Perrone F.L., Nelson S. C. A Study of On-Off Attack Models for Wireless Ad Hoc Networks.” Proceedings of the First IEEE International Workshop on Operator-Assisted (Wireless Mesh) Community Networks, OpComm2006, 2006.

[Pitoura, 93] Pitoura E., Bhargava B., “Dealing with Mobility: Issues and Research Challenges.” Purdue University, Department of Computer Sciences, Technical Report CDS-TR-93-070, November 1993.

[Prasad, 05] Prasad R.V., et al. ”Architectures for Intra-personal Network Communication.” Proceedings of the 3rd ACM international workshop on Wireless mobile applications and services on WLAN hotspots WMASH '05, pp. 115-118, September 2005.

[Prasanna, 08] Prasanna P., Gruenwald L., Vallur A., Atiquzzaman M., "A Survey of Data Replication Techniques for Mobile Ad-Hoc Network Databases." The International Journal on Very Large Data Bases Volume 17, Number 5, pp. 1143-1164, August 2008.

[Provost, 00] Provost P., Domingos P. " Well-Trained PETs: Improving Probability Estimation Trees." Stern School of Business, NYU, working paper #00=04-IS, 2000.

[Quercia, 06] Daniele Q. Hailes S., Capra L. “B-trust: Bayesian Trust Framework for Pervasive Computing.” Fourth International Conference on Trust Management, iTrust 2006, LNCS 3986, pp. 298-312, May 2006.

[Quinlan, 86] Quinlan, J. R. “Induction of Decision Trees.” Kluwer Academic Publishers, Machine Learning, pp. 81-106, 1986.

[Rakthanmanon, 12] Rakthanmanon T, Keogh E. “Making Time-Series Classification More Accurate Using Learned Constraints.” Proceedings of 4th SIAM International Conference on Data Mining, pp.11-22, 2004.

[Ratanamahatana, 04] Ratanamahatana CA, Zhu Q., Zakaria J., Keogh E. “Searching and mining trillions of time series subsequences under dynamic time warping.” Proceedings of The 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 262-270, August 2012.

[Redl, 95] Redl S.M., Weber M.K., Oliphant M.W. "An Introduction to GSM." Artech House, ISBN 978-0890067857, March 1995.

[RMG, 96] JPMorgan/Reuters. “RiskMetrics™ – Technical Document 4th Edition.” RiskMetrics Group, http://www.riskmetrics.com, New York 1996.

[RMG, 99] Kim J., Malz A.M., Mina, J. “LongRun” Technical Document, RiskMetrics Group, 1999.

[RMG, 01] Mina J., Xiao J. Y. “Return to RiskMetrics: The Evolution of a Standard.” RiskMetrics Group, April 2001.

[RMG, 04] RiskMetrics Group. “WealthBench Technical Document Preliminary Client Version.” RiskMetrics Group, January 2004.

[RMG, 06] RiskMetrics Group. “Risk University.” RiskMetrics Group, http://riskuniversity.org/, accessed May 2006.

[Rokach, 02] Rokach L., Maimon O. “Top-Down Induction of Decision Trees Classifiers - A Survey.” IEEE Transactions On Systems, Man And Cybernetics: Part C, Volume 1, Issue 11, November 2002.

[Rodríguez, 00] Rodríguez, J.J., Alonso, C.J., Boström, H. “Learning First Order Logic Time Series Classifiers: Rules and Boosting.” Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 299-308, September 2000.

[Rose, 02] Rose, C., Smith, M. D. "The Multivariate Normal Distribution." Section 6.4 in Mathematical Statistics with Mathematica, New York: Springer-Verlag, pp. 216-235, 2002.

[Ross, 03] Ross, S.H. “Introduction to Probability Models.” Academic Press, 8th edition, Chapter 10, ISBN-13: 978-0125980555, January 2003.

[RSA, 13] RSA Laboratories. "TWIRL and RSA Key Size." EMC Corporation, http://www.rsa.com/rsalabs/node.asp?id=2004, accessed, May 2013.

[Ryusuke, 05] Ryusuke M. et al. "Policy-based Access Control for Task Computing Using Rei." Proceedings of the Policy Management for the Web (PM4W) Workshop at WWW2005, May 2005.

[Ryutev, 05] Ryutev T., Zhou L., Neuman C., Leithead T., Seamons E. "Adaptive Trust Negotiation and Access Control." Proceedings of the tenth ACM symposium on Access control models and technologies, SACMAT'05, pp.139-146, June 2005.

[Saleeb, 08] Saleeb H. "Data Warehousing and Data Mining." New York University, Computer Science Department, G22.3033-003 Classification and Prediction Course Lecture, http://www.cs.nyu.edu/courses/spring08/G22.3033-003/

[Sanchez, 01] Sanchez C., Gruenwald L. "An agent based architecture using XML for Mobile Federated Database Systems." Proceeding MDM '01 Proceedings of the Second International Conference on Mobile Data Management, MDM '01, pp. 273-274, 2001.

[Sanchez, 07] Sanchez C., Gruenwald L., Sanchez, M. "A Monte Carlo Framework to Evaluate Context Based Security Policies in Pervasive Mobile Environments." Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access, MobieDE'07, pp. 41-48, June 2007.

[Sanchez, 08] Sanchez, C., Gruenwald, L., Sanchez, M., "Evaluating Security Policies in Pervasive Mobile Environments Using Context Information." Proceedings of 4th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, 2008.

[Sastry, 03] Sastry, N., Shankar, U., Wagner, D. "Secure Verification of Location Claims." Proceedings of the 2003 ACM workshop on Wireless security, pp. 1-10, September 2003.

[Schechter, 05] Schechter S. "Toward Econometric Models of the Security Risk from Report Attacks." IEEE Security and Privacy Volume 3, Number 1, pp. 40-44, January-February 2005.

[Schneier, 96] Schneier B. "Applied Cryptography: Protocols, Algorithms, and Source Code." C. J. Wiley & Sons Inc., 2nd edition, ISBN 0-471-11709-9, 1996.

[Shafiq, 05] Shafiq B., Joshi J., Bertino E., Ghafoor A. "Secure Interoperation in Multidomain Environment Employing RBAC Policies." IEEE Transactions on Knowledge and Data Engineering, Volume 17, Number 11, pp. 1557-1577, November 2005.

[Shand, 03] Shand B., Dimmock N., Bacon J. "Trust for Ubiquitous, Transparent Collaboration." Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, PERCOM'03, pp.153-160, March 2003.

[Shand, 04] Shand B., Dimmock N., Bacon J. "Trust for Ubiquitous, Transparent Collaboration." Wireless Networks Journal, Volume 10, Number 6, pp. 711-721, November 2004.

[Sheng. 03] Sheng Z., Richard Y. "Verifiable Distributed Oblivious Transfer and Mobile Agent Security." Proceedings of the 2003 Joint Workshop on Foundations of mobile computing, pp.12-21, September 2003.

[Sterbenz, 02] Sterbenz P., Krishnan R., Rosales R., Jackson J. et al. "Survivable Mobile Wireless Networks: Issues, Challenges, and Research Directions." Proceedings of the 3rd ACM workshop on Wireless security, pp.31-40, September 2002.

[Stuart, 99] Stuart, A, Ord, J. K. "Kendall's Advanced Theory of Statistics." Classical Inference & the Linear Model, 6th ed., Volume 2A, New York: Oxford University Press, p. 865, 1999.

[Sun, 06] Sun Developer Network (SDN). "Java Programming Language." Sun Corporation, http://java.sun.com, accessed June 2006.

[Swift, 02] Swift M., Hopkins A., Brundrett P., Dyke C.V., Garg P., Chan S. Goertzel M., Jensenworth G. "Improving the granularity of access control for Windows 2000." ACM Transactions on Information and System Security (TISSEC), Volume 5, Number 4, pp. 398-437, November 2002.

[Tan, 05] Tan P., Steinbach M., Kumar V. "Introduction to Data Mining." ISBN, 0321321367, May 2005.

[Teacy, 06] Teacy W.T.L., Patel J., Jennings N.R., Luck M. "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources." Journal of Autonomous Agents and Multi-Agent Systems, Volume 12, Number 2, pp. 183-198, March 2006.

[Teacy, 06b] Teacy W.T.L. “Agent-Based Trust and Reputation in the Context of Inaccurate Information Sources.” PhD Thesis, Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, University Of Southampton, December 2006.

[Tennenhouse, 04] Tennenhouse D. “Intel's Open Collaborative Model of Industry-University Research.” Research-Technology Management, Volume 47, Number 4, pp. 19-26, 2004.

[Thibadeau, 06] Thibadeau R. "Trusted Computing for Disk Drives and Other Peripherals." IEEE Security and Privacy, Volume 4, Number 5, pp. 26-33, September-October, 2006.

[Thompson, 04] Thompson C. “Everything is Alive”. IEEE Internet Computing, Volume 8, Number 1, pp. 83-86, January-February 2004.

[Trimble, 06] Trimble®. “GPS Tutorial.” Trimble Navigation Ltd, http://www.trimble.com/gps/index.shtml, accessed August 2006.

[Toivonen, 06] Toivonen. S., Lenzini. G., Uusitalo. I. “Context-aware Trust Evaluation Functions for Dynamic Reconfigurable Systems.” Proceedings of the Models of Trust for the Web workshop (MTW’06), in Conjunction with the 15th International World Wide Web Conference (WWW2006), May 2006.

[TPR, 07] The Physics Classroom. “Speed and Velocity.” The Physics Classroom, http://www.physicsclassroom.com/Class/1DKin/U1L1d.html, accessed September 2007.

[Tran, 04] Tran T., Cohen R. “Improving User Satisfaction In Agent-Based Electronic Market- Places By Reputation Modeling And Adjustable Product Quality.” Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 828–835, 2004.

[Uryasev, 00] Uryasev S. “Introduction to the Theory of Probabilistic Functions and Percentiles (Value-at-Risk).” Probabilistic Constrained Optimization: Methodology and Applications, Kluwer Academic Publishers, ISBN 978-0792366447, pp. 1-25, November 2000.

[Varshney, 03] Varshney U. "Location Management for Mobile Commerce Applications in Wireless Internet Environment." ACM Transactions on Internet Technology (TOIT) Volume 3, Issue 3, pp. 236-255, August 2003.

[Velloso, 08] Velloso P., Laufer R., Carlos O., Duarte M., Pujolle G. "A Trust Model Robust to Slander Attacks in Ad Hoc Networks." 17th International Conference on Computer Communications and Networks (ICCCN 2008), August 2008.

[W3C, 05] W3C. "The Web Ontology Language (OWL) Recommendation". Word, Wide Web Consortium, http://www.w3c.org/TR/owl-guide, accessed October 2005.

[Wan, 99] Wan G., Lin E. "Cost Reduction in Location Management Using Semi-real time Movement Information." Wireless Networks, Volume 5, Number 4, pp. 245-256, July1999.

[Wang, 02] Wang H., Cao J., Zhang Y. "Ticket-based Service Access Scheme for Mobile Users." Proceedings of the twenty-fifth Australasian conference on Computer science, ACSC2002, Volume 4, pp. 285-292, January- February 2002.

[Wang, 03] Wang H., Cao J., Zhang Y. "A Flexible Payment Scheme and its Permission-role Assignment." Proceedings of the 26th Australasian computer science conference, ACSC2003, Volume 16, pp. 189-198, February 2003.

[Wang, 04] Wang H., Sun L., Zhang Y., Cao J. "Anonymous Access Scheme for Electronic-Services." Proceedings of the 27th Australasian computer science conference, ACSC2004, Volume 26, pp. 295-304, January 2004.

[Wang, 10] Wang B., Ray K.J. "Cognitive Radio Networking and Security." Cambridge University, ISBN-13: 978-0-521-76231-1, October 2010.

[Want, 06] Want R. "Introduction to RFID Technology." IEEE Pervasive Computing, Volume 5, Number 1, pp. 25-33, January-March 2006.

[Wedde, 04] Wedde H.F., Lishka M. "Role-Based Access Control in Ambient and Remote Space." Proceedings of the ninth ACM symposium on Access control models and technologies, SACMAT'04, pp. 21-30, June 2004.

[Wei, 06] Wei L., Keogh E. “Semi-Supervised Time Series Classification.” Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 748-753, 2006.

[Wei, 06b] Wei L., Keogh E., Xiaopeng X., Wei L. “SAXually Explict Images: Finding Unusual Shapes.” Proceedings of the Sixth IEEE International Conference on Data Mining, ICDM 2006.

[Weisstein, 06] Weisstein E.W. "Wiener Process." MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/WienerProcess.html, accessed August 2006.

[Weisstein, 10] Weisstein E. W. "Random Walk--1-Dimensional." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/RandomWalk1-Dimensional.html, accessed March 2010.

[Weisstein, 10a] Weisstein E. W. "Normal Distribution." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/NormalDistribution.html, accessed March 2010.

[Weisstein, 13] Weisstein E. W. "Prime Factorization." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/PrimeFactorization.html , accessed May 2013.

[Weisstein, 13b] Weisstein E. W. "Arc Length." From MathWorld--A Wolfram Web Resource, http://mathworld.wolfram.com/ArcLength.html, accessed May 2013.

[Wenfei, 04] Wenfei F., Chan C.Y., Garofalakis M. “Secure XML Querying with Security Views.” Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 587-598, June 2004.

[Wierzbicki, 05] Wierzbicki A., Zwierko A., Kotulski Z. “A New Authentication Protocol for Revocable Anonymity in Ad-Hoc Networks.” The IASTED International Conference on Communication, Network and Information Security, CNIS 2005, November 2005.

[Witten, 11] Witten I., Frank E., Hall M. “Data Mining: Practical Machine Learning Tools and Techniques, Third Edition.” ISBN 0123748569, January 2011.

[Wu, 04] Wu, Y., Chang, E.Y. "Distance-function design and fusion for sequence data." Proceedings of the 13th ACM Conference on Information and Knowledge Management, pp. 324-333, 2004.

[Wullems, 04] Wullems, C., Looi, M., Clark, A. "Towards Context-Aware Security: Authorization Architecture for Intranet Environments." Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshop, PERCOMWY04, pp. 132-137, March 2004.

[Xi, 06b] Xi X., Keogh E., Shelton C., Wei L. "SAXually Explict Images: Finding Unusual Shapes." Proceedings of the 23nd International Conference on Machine Learning, 2006.

[Xiong, 02] Xiong L., Liu L. "Building Trust in Decentralized Peer-to-Peer Electronic Communities." Proceeding of the Fifth International Conference on Electronic Commerce Research, ICECR-5, pp. 1-15, October 2002.

[Xiong, 03] Xiong L., Liu L. "A Reputation-Based Trust Model for Peer-to-Peer Ecommerce Communities." IEEE Conference on Ecommerce, CEC'03, pp. 275-284, June 2003.

[Ye, 09] Ye L., Keogh E. "Time Series Shapelets: A New Primitive for Data Mining." KDD'09, June 29–July 1, 2009.

[Ye, 11] Ye L., Keogh E. "Time series Shapelets: a novel technique that allows accurate, interpretable and fast classification." Data Mining and Knowledge Discovery. Volume 22, Issue 1-2, pp. 149-182, January 2011.

[You, 04] You E.G., Lee K.S. "A Mobile Agent Security Management." Proceedings of the 18th International Conference on Advanced Information Networking and Application, AINA04, Volume 2, pp. 360-365, March 2004.

[Yu, 06] Yu W., Sun Y., Hany Z., Liu R. "Attacks on Trust Evaluation in Distributed Networks. Vulnerability Analysis and Defense against Attacks" Proceedings of the 25th IEEE International Conference on Computer Communications, INFOCOM 2006, 2006

[Yuan, 06] Yuan W., Donghai G., Sungyoung L., Youngkoo L. “A Dynamic Trust Model Based On Naive Bayes Classifier For Ubiquitous Environments.” Proceedings of the Second international conference on High Performance Computing and Communications, HPCC'06, pp. 562-571, 2006

[Zanten, 07] Zanten, H.V. "An Introduction to Stochastic Processes in Continuous Time." Lecture notes, Vrije Universiteit, Amsterdam. Available at http://www.math.vu.nl/sto/onderwijs/sp/sp_2007.pdf, 2007.

[Zakaria, 12] Zakaria J., Rakthanmanon T., Campana B., Mueen A., Batista G., Westover B., Zhu Q., Eamonn K. “Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping.” The 18th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 2012.

[Zadrozny, 01] Zadrozny B., Elkan C. “Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers.” Proceedings of the Eighteenth International Conference on Machine Learning, pp. 609-616, 2001.

[Zeckhauser, 00] Zeckhauser E., Resnic P., Kuwabara K. “Reputation Systems,” Communications of the ACM, Volume 43, Number. 12, pp. 45–48, 2000.

[Zhang, 08] Zheng Y, Quannan L., Yukun C., Xing X., Ma W-Y. “Understanding Mobility Based on GPS Data.” Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008), pp.312-321, 2008.

[Zhang, 09] Zheng Y, Zhang L., Xie X., Ma W-Y. “Mining interesting locations and travel sequences from GPS trajectories.” Proceedings of International conference on World Wild Web (WWW 2009), pp.791-800, 2009.

[Zhang, 10] Zheng Y, Xie X., Ma W-Y. “GeoLife: A Collaborative Social Networking Service among User, location and trajectory.” Invited paper, in IEEE Data Engineering Bulletin, Number 33, Volume 2, pp. 32-40, 2010.

[Zhang, 03] Zhang G. “Dynamic Context Aware Access Control for Grid Applications.” Proceedings of Fourth International Workshop on Grid Computing, pp.101-108, November 2003.

[Zhang, 04] Zhang G., Parashar M. “Context-aware Dynamic Access Control for Pervasive Applications.” Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference, CNDS 2004, pp. 21-30, January 2004.

[Zhang, 06] Zhang G., Parashar M. “Environment Sensitive Access Management for Pervasive Grid Applications.” Journal of Cluster Computing, Springer Netherlands Publishers, Volume 9, Number 2, pp. 19-27, January 2006.

[Zheng, 12] Zheng Y., Xiangye X., Xie X. “Inferring Social Ties Between Users With Human Location History.” Journal of Ambient Intelligence and Humanized Computing, December 2012.

[Zwierko, 05] Zwierko A., Kotulski Z. “Security of Mobile Agents: A New Concept of the Integrity Protection.” Cornel University Library, Volume abs-cs-0506103, 2005.