Marquette University

e-Publications@Marquette

Computer Science Faculty Research and
Publications

Computer Science, Department of

2-2010

Design, Analysis, And Deployment of Omnipresent Formal Trust Model (FTM) With Trust Bootstrapping for Pervasive Environments

Sheikh Iqbal Ahamed

Marquette University, sheikh.ahamed@marquette.edu

Munirul Haque

Marquette University, md.haque@marquette.edu

Md. Endadul Hoque

Marquette University

Farzana Rahman

Marquette University

Nilothpal Talukder

Marquette University

Follow this and additional works at: https://epublications.marquette.edu/comp_fac

Recommended Citation

Ahamed, Sheikh Iqbal; Haque, Munirul; Hoque, Md. Endadul; Rahman, Farzana; and Talukder, Nilothpal, "Design, Analysis, And Deployment of Omnipresent Formal Trust Model (FTM) With Trust Bootstrapping for Pervasive Environments" (2010). *Computer Science Faculty Research and Publications*. 42.

https://epublications.marquette.edu/comp_fac/42

Marquette University

e-Publications@Marquette

Computer Science Faculty Research and Publications/College of Arts and Sciences

This paper is NOT THE PUBLISHED VERSION; but the author's final, peer-reviewed manuscript. The published version may be accessed by following the link in the citation below.

Journal of Systems and Software, Vol. 83, No. 2 (February 2010): 253-270. [DOI](#). This article is © Elsevier and permission has been granted for this version to appear in [e-Publications@Marquette](#). Elsevier does not grant permission for this article to be further copied/distributed or hosted elsewhere without the express permission from Elsevier.

Design, Analysis, And Deployment of Omnipresent Formal Trust Model (FTM) With Trust Bootstrapping for Pervasive Environments

Sheikh I. Ahamed

Mathematics, Statistics and Computer Science, Marquette University, Milwaukee, WI

Munirul M. Haque

Marquette University, Milwaukee, WI

Md. Endadul Hoque

Marquette University, Milwaukee, WI

Farzana Rahman

Marquette University, Milwaukee, WI

Nilothpal Talukder

Marquette University, Milwaukee, WI

Abstract

The rapid decrease in the size of mobile devices, coupled with an increase in capability, has enabled a swift proliferation of small and very capable devices into our daily lives. With such a prevalence of pervasive computing, the interaction among portable devices needs to be continuous and invisible to device users. As these devices become better connected, collaboration among them will play a vital role in sharing resources in an ad-hoc manner. The sharing of resources works as a facilitator for pervasive devices. However, this ad hoc interaction among devices provides the potential for security breaches. Trust can fight against such security violations by restricting malicious nodes from participating in interactions. Therefore, we need a unified trust relationship model between entities, which captures both the needs of the traditional computing world and the world of pervasive computing where the continuum of trust is based on identity, physical context or a combination of both. Here, we present a context specific and reputation-based trust model along with a brief survey of trust models suitable for peer-to-peer and ad-hoc environments. This paper presents a multi-hop recommendation protocol and a flexible behavioral model to handle interactions. One other contribution of this paper is the integration of an initial trust model; this model categorizes services or contexts in different security levels based on their security needs, and these security needs are considered in trust bootstrapping. The other major contribution of this paper is a simple method of handling malicious recommendations. This paper also illustrates the implementation and evaluation of our proposed formal trust model.

Keywords

Pervasive computing, Trust, Initial trust, Malicious recommendation

1. Introduction

As people interact more in a world of pervasive computing, there is an increased interaction between people and smart devices. In such a world, computing power will move from big desktops to very small and miniature devices. Because of this increased efficiency of technology, seamless integration of computing power and human lives will become very important. And only advances in the pervasive computing era can help achieve this goal. Significant numbers of handheld devices may frequently join and leave this pervasive environment. These devices need to communicate and share content in an ad-hoc manner. The increasing availability of small computing devices (PDAs, mobile phones, etc.) and the deployment of next-generation wireless networks (UMTS, Wi-Fi, Bluetooth) are helping to make the vision of Weiser (1993) a reality. As mutual collaboration is unavoidable in a pervasive environment, it is very important to limit the interaction among trustworthy devices. For that, we need to find the answer to the question “With which node(s) should I interact and with which I should not?” Trust models have been developed to resolve this issue. These models are also responsible for establishing and managing trust relationships among the interacting devices. We believe that there is a need for a continuum of trust which mimics models the real world as closely as possible. In other words, we need to capture the real world model of trust where entities (people and devices) trust one another to varying degrees and extents.

Trust is an integral part of security and one of the prime concerns in pervasive computing because of its decision-making role in resource sharing. Trust models have been proposed by researchers considering the diverse scenarios of distributed systems, P2P networks, agent based systems, pervasive computing, etc. Several well-known trust models fail to be suitable in pervasive computing due to the use of different cryptographic and secret key sharing techniques like (Zhou and Haas, 1999) threshold cryptography (Josang, 1996, Josang, 1999, Josang and Knapskog, 1998), certificate, key binding, and trust relationship. These are considered too heavy weight for pervasive devices. Due to the presence of a fixed infrastructure in which a fixed device has to act as security agent in each domain, or a central server which is a requirement of central key server, trust

models proposed by Garfinkel (1995) are inappropriate here since devices can leave or join at any moment in this environment. Trust models requiring pre-configuration like the Distributed Public Key Model (Zhou and Haas, 1999) are also infeasible in a pure ad-hoc scenario where participating devices are not known to one another ahead of time. Some trust models are not viable in this environment due to their overly complex structure. This category includes Dynamic Trust Management (Blaze et al., 1996) (trust actions and policies have been specified through complex language), Virtual Community Trust (Rahman and Hailes, 2000) (extremely long recommendation chain), and Resurrecting Duckling model (Stajano and Anderson, 2002) (hierarchical chain of security).

These models suffer from one or more major shortcoming. However, the majority of these models considered a constant initial trust value for service-requesting devices, which is totally unknown to the service provider. Yet an effective and nearly accurate initial trust calculation is necessary since it is a precondition for successful trust evolution. Therefore, trust models must incorporate an adaptive initial trust while deciding whether to accept or reject a service request. Some preliminary work on an initial trust model is presented in Ahamed et al., (2008a, Ahamed et al., 2008b). Furthermore, most of the trust models have avoided the issue of malicious recommendations by passing this responsibility on to an authentication mechanism. But real life scenarios and facts prove that malicious nodes may exist even after adopting authentication techniques. So a trust model should also have the feature of handling malicious recommendations. Moreover, most of the trust models are node/device specific; that means they either trust or distrust a device as a whole. But consider a real life scenario where a device may be trusted for a context C_i and not trusted for some other context C_j . Thus, we need a context specific trust model and initial work on such a trust model is presented in Haque and Ahamed (2007). This paper describes the details of design, implementation and evaluation of the context specific trust model that incorporates dynamic initial trust and an efficient technique to deal with a malicious recommendation.

The contributions we present in our paper can be summarized as follows: (1) a survey of all trust models of the pervasive computing environment. (2) Formal definitions and classification of desired properties of 'trust', along with a recommendation protocol and an omnipresent behavioral model. (3) Architecture and conceptual diagram of our omnipresent trust model. (4) An initial trust protocol for determining a dynamic initial trust for an unknown device based on the security level of the requested context. (5) A simple statistical model for isolating malicious recommendations and gradually weeding out malicious nodes. (6) Simulation, performance measurement, and prototype implementation of our proposed trust model.

The organization of our paper is as follows: Section 2 presents the motivation behind our proposal. A survey of all trust models in a pervasive environment are presented in Section 3. The definition and important characteristics of trust are presented in Section 4. The classification of trust is described briefly in Section 5. Section 6 demonstrates the architecture and state diagram of our proposed model. The formation of different types of trust is discussed in Section 7. Sections 8 Recommended trust protocol, 9 Direct trust protocol, 10 Initial trust protocol describe the recommended, direct, and initial trust protocol, respectively. Overall trust calculation is explained in Section 11. Section 12 presents a malicious recommendation handling process. Our evaluation methodologies and some results are described in Section 13. Along with the conclusion, some open issues are stated in Section 14.

2. Motivation

Pervasive computing is susceptible and vulnerable to malicious active and passive snoopers. This is due to the unavoidable inter-device dependency, as well as a common shared medium, very transitory connectivity, and the absence of a fixed trust infrastructure. Pervasive devices depend on each other for resources since the small devices do not have the luxury of containing all required resources due to several limitations including poor

battery power and small memory storage. Hence, the issue of resource sharing comes up. In the volatile scenario of pervasive computing, where each device can join and leave arbitrarily, the knowledge of a fairly trusted device is very important for other devices. If a malicious device is considered to be a trusted neighbor in such a network, then the security and privacy of the whole system might collapse. Consider a hospital scenario as an example, where doctors, nurses, and other medical staffs can communicate with one another through a wireless network. A new person can enter the hospital premise as a visitor. He could possibly communicate with others posing as a doctor and access sophisticated and classified patient records. Hence, a trust model is very important for pervasive computing applications in order to determine trusted neighbors. This trust model will also be responsible for detecting malicious users within the network. Since mutual collaboration is unavoidable in this scenario, it is very important for one device to find out the trustworthiness of the service-requesting device before sharing service. For that, we need to find the answer to the question "To which node should I provide service and to which node should I not?" Trust models have been developed to resolve this issue, and be responsible for establishing and managing trust relationships among the interacting devices. Thus, the trust becomes an integral part of security and one of the prime concerns in pervasive computing because of its decision-making role in resource sharing (Robinson et al., 2004).

It is a common scenario for a device (service provider) to receive a service-sharing request from a device (service requester) unknown to it. In these cases, the service provider (SP) device makes a decision based on the recommendation of other devices who know the service-requesting (SR) device. This general idea of reputation-based trust models works fine with the assumption that there are no malicious recommendations. But that is not a real life scenario. Some trust models have bypassed this issue by assigning this responsibility to the authentication mechanism. However, the fact is that, despite the presence of an authentication mechanism, we can find active malicious devices in the network. We need a model to identify malicious recommendations to ensure the accuracy of the trust model.

Moreover, a majority of trust models did not take into account security concerns of the context while bootstrapping trust. In some models, security is considered with the evolution of trust. Security consideration is necessary for bootstrapping trust because the need for security changes with contexts. Furthermore, in a pervasive environment, some service-sharing contexts may need high security, and it is not sufficient to assign initial trust using only past experience or context nature. In such situations, context security should make a major contribution to the calculation of the initial trust value. In some real life applications of handheld devices, a device (SP) needs to consider the security requirement of the service to be shared with another device (SR). Such context security concerns need to be incorporated in the overall trust assignment including the initial trust calculation. How a device will be trusted initially requires consideration of the security need of contexts, since it can enhance the effectiveness of an entire trust model. Therefore, we also need a method in the trust model for computing initial trust based on the security level of different contexts in order to incorporate a security factor in the initial trust calculation.

Consider these scenarios that show the importance of a trust model as well as initial trust awareness of such models.

Scenario 1.

Mr. Simon works as the front desk assistant in Aurora Hospital. He requests the duty roster file for the following week from Dr. Mason, Co-ordination Manager of the hospital, and receives the file. Later he tries to get a file containing sophisticated patient information, the same person but is denied. On the other hand when Dr. Morrice, Director of Ophthalmology, requests those two files, he receives both.

Scenario 2.

Bob is a registered attendee at a conference. Unfortunately, Bob fails to reach the conference in time and misses the key note speech. So he requests the voice files and receives them. Later he visits some presenters' (Dr. Iqbal, Dr. Haifeng and Dr. Sam) web pages and requests detailed information about some of their ongoing works. However, they deny Bob's request. Being persistent, Bob sends continual requests for those files. As a consequence Bob was discarded from the network.

Scenario 3.

Sunny was walking through Mayfair shopping mall. Suddenly he saw the song 'Hips don't lie' in the big screen. He was interested to get that song in his mp3 player. So he requested that song through the wireless network connection of the mall. Lubna had received that request but unfortunately she did not know Sunny. So she requested a recommendation regarding Sunny. Most of the recommendations were positive, but two unknown persons intentionally gave a negative recommendation about Sunny. Finally Lubna decided not to share the song.

Scenario 4.

Rachel is an executive at a multinational company located in Washington and she has to travel by subway every day. She is stranded in the subway because of some technical problems with the train on the day of an important meeting, in which she is a keynote speaker. Some major problems will occur if she fails to reach the office in time. She thinks of an alternative idea and calls one of her colleagues to do the presentation in the meeting. Since there is little time left before the meeting, her colleague asks her to send the presentation slides through e-mail. Rachel wants to instantly access the Internet, but does not have Internet service on her cell phone or PDA; she wants to request Internet service from any nearby devices.

Scenario 5.

Susan is a student intern in a hospital's pediatric department. She has answered the call of a seriously ill child, and wants the detailed medical information about that patient. The information is stored in the PDA of her senior doctor, who is currently busy in emergency surgery. Susan wants to access that information from her senior doctor's PDA by requesting the specific healthcare assistant.

The first scenario implies that the trust value of Mr. Simon is not enough to get the sophisticated patient information file. This scenario implies there should be various security levels or required trust values for different services. In Scenario 2 Bob has been discarded due to his malicious behavior even though he was a valid user. Scenario 3 implies that the recommended trust regarding Sunny was not sufficient enough to let him download the music files. Scenario 4 indicates that internet service requires high security, and security factors should be incorporated at the time of the initial trust calculation. Scenario 5 also points out that the requested service requires security. Such security concerns should be considered in initial trust assessment because those services require access to critical and confidential information. The above scenarios point out that a complete trust model should be able to do the following tasks: (1) compute trust based on the interaction between devices, (2) detect malicious nodes and (3) assign initial trust based on the security requirement of the context.

3. Related work

Trust models have been proposed by researchers considering the diverse scenarios of distributed systems, P2P networks, agent based systems, pervasive computing, etc. Several well-known trust models (Zhou and Haas, 1999, Josang, 1996, Josang and Knapskog, 1998, Josang, 1999, Kagal et al., 2001a, Kagal et al., 2001b, Garfinkel, 1995) fail to be suitable in pervasive computing due to the unique characteristics of the environment. Due to the

presence of a fixed infrastructure (a fixed device has to act as security agent in each domain (Garfinkel, 1995)) or central server (requirement of central key server (Kagal et al., 2001b)), several trust models are inappropriate here since devices can leave or join at any moment. Trust models requiring pre-configuration like the Distributed Public Key Model (Zhou and Haas, 1999) are also infeasible in a purely ad hoc scenario where participating devices are not known ahead of time. Some trust models are not viable in this environment due to their overly complex structure. This category includes Dynamic Trust Management (Blaze et al., 1996) (trust actions and policies have been specified through complex language), Virtual Community Trust (Rahman and Hailes, 2000) (extremely long recommendation chain), and Resurrecting Duckling model (Stajano and Anderson, 2002) (hierarchical chain of security).

Some models (Guha and Kumar, 2004, Kamvar and Schlosser, 2003, Matthew and Agrawal, 2003, Theodorakopoulos and Baras, 2004) have little focus on direct and overall trust formation and mainly deal with the composition and communication mechanism of trust data. A probabilistic approach has been adopted in the Beta Reputation System (Ismail, 2002), B-trust (Quercia et al., 2006), and Learning Model (Huebscher and McCann, 2005). Along with these models, a large number of recent trust models fall into the distributed and dynamic category (Sharmin et al., 2006a, Almenarez et al., 2004, Shand et al., 2003, Jameel et al., 2005, Basu and Callaghan, 2005, Wolfe et al., 2006, He et al., 2004). But these models suffer from one or more of the following shortcomings: (1) failure to handle situations when a node is depending on old, valid recommendations. (2) No methodology for a multi-hop recommendation. (3) Absence of a weighted recommendation – we need this because all recommendations should not be treated equally. (4) Failure to incorporate the security level of the context/service in trust or risk calculations. Moreover, most of the trust models are node/device specific; that means they either trust or distrust a device as a whole. In real life scenarios where a device may be trusted for a context C_i and not trusted for some other context C_j , the need for a context specific trust model is evident.

One of the pioneer models is PGP (Pretty Good Privacy) (Garfinkel, 1995) where each node can act as certification authority. But this model fails due to the requirement of a central key server. Rahman and Hailes (1998) introduced the notion of ‘distributed trust model’. A Distributed Public Key Model (Zhou and Haas, 1999) uses threshold cryptography and requires the pre-configuration of nodes. These models are expensive due to battery power requirements. The reputed game theory and distributed algorithm are pillars of the trust model proposed by Sun and Song (2004). Here we summarize some of the most recent approaches towards modeling trust.

A generic Bayesian trust framework has been proposed in (Quercia et al., 2006). Here, each peer (p_x) stores four data sets for each context for all the nodes with which it has interacted. The data set are: Direct Trust Set (DTS), Direct Experience Set (DES), Recommended Trust Set (RTS), and Sent Recommendation Set (SRS). Overall trust is generated from a weighted calculation of both direct trust and recommended trust as a distributed trust vector. A single trust value is extracted from the overall trust and sent to the decision module along with the confidence value. From the output of the decision module, the peer p_x takes the best decision to perform. When a peer (p_x) interacts with a new peer, the new peer is assigned a constant value for direct experience (Δ_d) and recommendation (Δ_r). But not mentioned anywhere is what the optimal values for these constants should be. Based on peer-to-peer interaction, a tuple $(d_1, d_2, \dots, d_n)$ is generated in this model indicating the trust distribution. But no criteria or behavior model has been defined for generating elements in the trust vector. It has been mentioned that, with the passage of time, the values for the elements in the tuple will be converged to bootstrapping values, yet the time information has not been specified. This information plays a vital role since the tuple entry is deleted when all the values converge to bootstrapping values. If the defined time is too long, then a peer has to save a lot of entries. At the same time, if the amount of time is too

small, it might have to go through repeated trust calculations. This model is also device specific, not context specific.

Sharmin et al. (2006a) proposed a trust model that works as a supporting module for SSRD (Simple and Secure Resource Discovery) and MARKS (Sharmin et al., 2006b). Complete trust and complete distrust are represented by 1.0 and 0.0, respectively. This context specific model supports reflexivity and partial transitivity. Each node maintains a table of trust values for its neighbors. Unicast, multicast, or broadcast strategies are followed based on the level of security of the offered service. Though this model indicates it is about the dynamic update of trust values, that issue has not been further addressed; overall trust calculation has not been a focus of this model. It does not consider a priority level of service when updating trust values.

Markus C. Huebscher and Julie A. McCann proposed a learning model for determining the trustworthiness of CP (context providers) (Huebscher and McCann, 2005). Positive/negative feedbacks of other nodes (those that have already considered service from the CP) and other CPs are input into a Bayesian parameter model that uses a parameterized probability density function (here beta distribution has been used). In order to make the system dynamic, a window of the last n feedbacks are counted in the beta distribution. The output of the Bayesian model is a probability density function showing the trustworthiness of the context provider. A probabilistic majority approach is used to nullify the effect of intentional negative feedback. A big limitation here is that is the model assumes that every device is reachable by every other device. For that reason, it does not provide any multi-hop recommendation mechanism.

Pervasive Trust Management (PTM) Model (Almenarez et al., 2004) is a decentralized trust model for ad hoc environments. This dynamic model implements a recommendation-based trust protocol for transferring data. Here a trust value of 0 represents complete distrust whereas 1 represents complete trust. First a belief space is formed from prior information and later, based on interaction, an evidence space is developed. This model was further enhanced to be a probabilistic trust model (Almenarez et al., 2006) that can be used as a beginning step in risk assessment. Here, the trust value is the same for all contexts (services) of a specific device. Another problem is that although a passive recommendation protocol has been incorporated, it is not capable of properly handling multi-hop passive recommendations.

Shand et al. (2003) proposed a trust scheme along with a risk assessment model for sharing resources with an effort to minimize user intervention. The trust assessment has been unified with access control. Trust on a node has been represented by the pair (belief, disbelief) where $(\text{belief} + \text{disbelief}) \leq 1$. Nodes are divided into several categories such as super user (owner), privileged, group, and public, which actually correspond to roles in RBAC (Role Based Access Control) system. An overall recommendation is calculated through a transitive combination of recommendation values. Not mentioned is how to avoid long chains of recommendations, and not addressed is how to handle a new node with no prior interaction records.

Jameel et al. (2005) proposed a trust model based on the vectors of trust values formed by different nodes in the network. The trust assessment depends on suggestions from neighboring entities that are mutually known to both entities taking part in the interaction. A method for handling a false recommendation has also been added. Xiu and Liu (2005) presented a generalized trust model with the intention to include all trust evaluation and establishment strategies in a single model. They describe a high level idea with no details of how the whole model will be implemented. Also absent are the verification or calculation methodologies for trust evidence and overall trust.

Taking uncertainty into account, He et al. (2004) proposed a model of trust that is based on cloud theory (Li et al., 1995). In order to elaborate the trust relationship involving two entities, a trust cloud has been introduced, denoted by three digital attributes (Expected Value, Entropy, and Hyper-Entropy). They also effectively

differentiated between distrust and unknown trust. A trust instance value is calculated from the trust cloud to make the ultimate decision regarding trust. A big limitation of this model is that all the entities are assumed to be honest. Though a relatively new term 'uncertainty' has been introduced, major factors that have not been considered include behavioral model and recommendation.

Lin et al. proposed a trust management model 'MobileTrust' (Lin and Varadharajan, 2005) to enhance security with the help of trust. The model is intended for mobile agents in distributed computing systems. They provide an approach for trust evaluation and updating the trust relationships of mobile agents. Risk was incorporated later in their trust model (Lin and Varadharajan, 2006). A threshold parameter is used which is directly related to risk on one side and system utility on the other. The performance of the model is highly dependant on this parameter but the choice of value for this parameter has been left as future work. Peer Trust (Xiong and Liu, 2004) and Power Trust (Zhou and Hwang, 2007a) are special trust models designed for peer-to-peer overlay networks with a Distributed Hash Tables (DHT) technique in order to achieve scalability in aggregating and managing reputation data. A scalable, robust and fault-tolerant reputation aggregation technique called Gossip Trust (Zhou and Hwang, 2007b) is designed for diverse and unstructured P2P networks (Gnutella, Kaaza, etc.). It adopts the gossip protocol to aggregate global reputation scores. But it does not incorporate distance-based aging of the nodes. The robust Bayesian reputation system (Buechegger and Boudec, 2004) provides a well-designed model of punishment and redemption of misbehaving nodes in the network. But distance and time-based aging is also not considered for the mobile ad-hoc network.

Discussed so far are some trust models that do not explicitly address initial trust. However, researchers have effectively devised some ways to estimate initial trust. Though our model calculates initial trust along the way of evolution of trust, we discuss some important initial trust models next. One way of assigning initial trust value is proposed in TRULLO (Quercia et al., 2007). This model uses "Single Value Decomposition" for trust bootstrapping. It sets initial trust values that are not arbitrary but are based on properties of its user's past experience. The initial trust model called CCTB (Ahamed et al., 2008a) takes advantage of the correlation existing between different contexts. It is also illustrated that CCTB offers a better initial trust value than some other known models. This trust model did not address the security level of the context in initial trust assessment. In another model (Ahamed et al., 2008b), the authors proposed an initial trust model. They categorized contexts in different security levels in order to incorporate them in trust bootstrapping. This allows the initial trust value to change proportionately with the context security level. A comparison of different famous trust models is presented in Table 1.

Table 1. Comparison of trust models.

Model	Adaptive	Need infrastructure support	Light weight	Dynamic	Distributed	Context specific	Behavioral model	Risk model	DBA	TBA	Initial trust model
B-trust Quercia06]	N/A	N	Y	Y	Y	N/A	Y	N	N	Y	N
SSRD [Sharmin06a]	Y	N	Y	Y	Y	Y	N	N	N	N	N
PTM [Almenarez04]	N/A	N	Y	Y	Y	N	L	N	N	Y	N
FDTM [Xiu05]	Y	Y	N	Y	Y	N	N	N	N	N	N
LMT [Huebscher05]	Y	N	Y	Y	Y	N	L	N	N	Y	N
TUTC [Shand03]	Y	N	Y	Y	Y	N	N	Y	N	Y	N
TMUS [Jameel05]	Y	N	Y	Y	Y	N	L	N	N	Y	N
TRAC [Basu05]	N	N/A	N	Y	Y	N	N	N	N	N/A	N
TPAN [Pirzada04]	Y	N	Y	Y	Y	N	N	N	N	N/A	N
TFPC [Wolfe06]	N/A	N	Y	Y	Y	N	N	N	N	N	N
CBTM [He04]	N/A	N	L	Y	Y	N	N	N	N	N	N
Mobile Trust [Lin05]	Y	N	Y	Y	Y	Y	L	N	N	Y	N
Our approach	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y

Y – yes, N – no, L – limited, N/A – not available, MHR – multi-hop recommendation, DBA – distance-based aging, TBA – time-based aging.

4. Trust definition and trust properties

In order to maintain and build trust relationships, first it is necessary to have knowledge of trust. In this section, we define the concept of trust and highlight some of its important characteristics.

4.1. Definition of trust

We define trust with a very popular commonly accepted definition (Gambetta, 1988): “[Trust] (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent will perform a particular action, both before [we] can monitor such action (or independently of his capacity of ever be able to monitor it) and in a context in which it affects [our] own action”. According to Gambetta, “trust is the probability that the trusted entity A does something, which is beneficial to the trusting entity B”. Trust plays an important role in the decision making about whether to reveal any information to the requesting service/device. Trust is determined by the confidence level; depending on that confidence level, a user decides whether to give out information or not and protect his/her privacy. Trust has been defined by several researchers (Almenarez et al., 2004, Gambetta, 1988) from different points of view. Terms such as risk, confidence, probability, etc. have been used in the definition of trust. Gambetta (1988) defined trust as the probability of doing something benevolent by a trusted node A for the trustor Almenarez et al. (2004) described trust as “the belief that an entity has about other entity, from past experiences, knowledge about the entity’s nature and/or recommendations from trusted entities. This belief expresses an expectation on the entity’s behavior, which implies a risk”. In pervasive computing, we formulate trust through previous and present interaction experiences of the associated nodes.

4.2. Characteristics of trust

In a pervasive environment, trust models and their properties can be referred through many formal mathematical model properties. Some of these properties evolved to address the unique characteristics of this environment. Thus, we define the properties of trust here from their scope in pervasive trust models. In forming trust relationships and properties, we used the human trust scheme as an ideal model to follow. In this paper, we used the term ‘context’ interchangeably with ‘service’. The term ‘node’ has been used to denote participating devices (PDAs, cell phones, etc.) in the network. Throughout our paper, the term ‘interaction’ denotes an action (for example, a request or sharing) regarding a context or service. Complete trust and complete distrust have been represented by the trust values 1 and 0, respectively. The binary operator T indicates the trust relationship. Each node has a list of available contexts or services ($C_1, C_2, C_3, \dots, C_i, \dots, C_k$). In our model we consider the notation mentioned in Table 2. Here, $T_D(A, B, C_i) = \phi$ indicates that there is no direct trust value of A on B for context C_i

$$T_D(A, B) = \frac{\sum_{i=1}^k T_D(A, B, C_i)}{m}$$

$$T_R(A, B) = \frac{\sum_{i=1}^k T_R(A, B, C_i)}{n}$$

Section 7 provides details of the equation for the overall trust value calculation. Trust has the following properties and relationships:

Table 2. Notations.

$T_D(A, B, C_i)$	Direct trust of A on B for context C_i
$T_D(A, B)$	Average direct trust of A on B
$T_R(A, B, C_i)$	Recommended trust of A on B for context C_i
$T_R(A, B)$	Average recommended trust of A on B

$T_I(A,B,C_i)$	Initial trust of A on B for context C_i
$T(A,B,C_i)$	Overall trust value of A on B for context C_i
$T(A,B)$	Average overall trust of A on B
k	Number of available contexts for A
m	Number of contexts for which A has direct trust value for B
n	Number of contexts for which A has recommended trust value for B

C1. Reflexive: If a device A has several contexts $(C_1, C_2, C_3, \dots, C_i, \dots, C_k)$ then the trust value of A on itself for any context C_i is 1

$$\forall C_i (T(A,A,C_i) = 1), \text{ where } i = 1, \dots, k$$

C2. Non-symmetric: For any context C_i , the trust of node A on node B does not necessarily indicate the same trust of B on A

$$T(A,B,C_i) \neq T(B,A,C_i)$$

Even if both A and B trust each other, that does not necessarily indicate the trust value of A on B (T_1) equals the trust value of B on A (T_2). If we delve a bit more and expand on this property about a specific context C_k where $T(A,B,C_k)$ indicates the trust of A on B for context C_k , then all the above statements also hold. If both A and B have the same context C_k (for example we can think of a service date and time sharing) to offer, then $T(A,B,C_k)$ indicates a high probability for $T(A,B,C_k)$.

C3. Partial transitive: If A trusts B and B trusts C for a particular context, then this chain relationship creates a partial trust value for A on C

$$\forall C_i (T(A,B,C_i) = \alpha \wedge T(B,C,C_i) = \beta \Rightarrow T(A,C,C_i) = f(\alpha, \beta)), \text{ where } i = 1, \dots, k$$

C4. Non-antisymmetric: If A trusts B and B trusts A, that does not indicate $A = B$

$$(T(A,B) = \alpha \wedge T(B,A) = \beta \neq A = B), \text{ where } \alpha \text{ or } \beta \text{ or may not be equal}$$

C5. Dynamic: The trust value of a node A on B for context C_i changes over time due to newer interactions

$$T_D(A,B,C_i)_{new} > T_D(A,B,C_i)_{old}, \mu > T_D(A,B,C_i)_{new}$$

$$T_D(A,B,C_i)_{new} < T_D(A,B,C_i)_{old}, \mu < T_D(A,B,C_i)_{new}$$

$$T_D(A,B,C_i)_{new} = T_D(A,B,C_i)_{old}, \mu = T_D(A,B,C_i)_{new}$$

Here, μ indicates the satisfaction level for the latest interaction which is described later.

C6. Context aware: Node A may have different trust values on B for different contexts or services. If node A has a trust value on B for a specific context C_i , this does not indicate that A will trust B for other available contexts

$$T_D(A,B,C_i) \neq T_D(A,B,C_j), \text{ where } i \neq j$$

C7. Time-based aging: The trust value of A on B for a specific context C_i decreases with the passage of time

$$\forall C_i (T_D(A,B,C_i))_t > T_D(A,B,C_i)_{t+\Delta t}, \text{ where } i = 1, \dots, k$$

C8. Distance-based aging: If node A collects trust values about B from other nodes in the network (recommendation), the trust values collected from closer nodes should be counted with more weight compared to the values collected from distant nodes. Distance-based aging ensures this feature.

5. Classification of trust

Trust in a pervasive environment mainly develops based on the interaction among devices. A device may establish a trusted relation with another device based on the satisfaction of a direct interaction with that device. This is known as *Direct Trust*. It is also possible to establish trust of other devices based on indirect interactions, which is known as *Recommended Trust*. Recommended trust can be further categorized into *active* and *passive*. Fig. 1 shows the classification of trust. We will discuss each type of trust and their evolution in Section 7.

Fig. 1. Classification of trust.

6. Architecture of our trust model

A trust model in a pervasive environment is responsible for assigning a direct or recommended trust value for each device and each context. The architecture of our trust model is shown in Fig. 2. This composite model contains three main components: (1) *trust model (FTM – Formalized Trust Model)*, (2) *Initial Trust Model (ITM)*, and (3) *Malicious Recommendation Handler (MRH)*.

Fig. 2. Architecture of our model.

6.1. Architectural diagram

Our trust model has a component-based architecture which is shown in Fig. 2. In this section we briefly discuss the architecture our model.

6.1.1. FTM

The *Formal Trust Model (FTM)* is comprised of Direct Trust Unit and Recommended Trust Unit. Direct trust is formed through direct interaction experience among the nodes. A behavior model is used to evaluate the satisfaction level of the direct interactions. A recommended trust protocol is used to evaluate the recommendations to form recommended trust.

6.1.2. ITM

The *Initial Trust Model (ITM)* is responsible for assigning an adaptive initial trust for unknown devices. ITM categorizes services in different security levels based on their security needs. A security factor is generated based on the user's disposition value and the security level of the requested service. Based on these values, an initial trust value is generated. ITM is a separate component and it enhances the functionality of FTM (Haque and Ahamed, 2007) by generating an adaptive initial trust value for unknown devices. However, as ITM mainly works with direct trust, it has been shown as a part of direct trust component in Fig. 2.

6.1.3. MRH

This *Malicious Recommendation Handler (MRH)* is used for weeding out a malicious recommendation and thus improving the accuracy of the trust model. It has two units. The α Unit operates when the number of recommendations is small. The operational methodology is based on control chart method. The β Unit works when the number of recommendations is large and this unit works on a probabilistic approach known as Student's t -Distribution.

6.2. State diagram

Trust model (FTM), Initial Trust Model (ITM), and Malicious Recommendation Handler (MRH) work together to ensure trust establishment with minimal interaction with the user. As our trust model is service oriented, it requires user interaction at the beginning to set security levels for several offered services. It also requires the user to set the value of disposition based on the environment in which she/he belongs. Other than these, the whole model operates transparently to the user. Fig. 3 represents the conceptual diagram of the model.

Fig. 3. State diagram of our model.

7. Trust formation

When a node joins a pervasive network for the first time, it does not have any prior interaction record or history. A user's circumvention is necessary to decide whether to trust or distrust that node. We use a behavioral model to find a node's pattern of behavior and its satisfaction level with some other nodes after each interaction. This satisfaction level is used in updating direct trust values. Overall trust is formed through a weighted average of direct trust and recommended trust. A decision module is responsible for making a decision based on the trust value.

7.1. Direct trust formation

The evolution of direct trust depends on the direct interaction or experience with other devices. Based on the satisfaction level of an interaction between two devices, they modify their direct trust value. This direct interaction is shown by a direct link between A and B in the network topology in Fig. 4. This portion of trust is critically important since it does not depend on the recommendation of any trusted neighbor. So when two nodes, A and B, interact directly, they change their direct trust value based on the satisfaction level of that interaction.

Fig. 4. A node topology with trust relationships.

7.2. Recommended trust formation

This is the trust value that is suggested by other trusted nodes about a specific node. For example, A knows B and C but A does not know D. However, B and C know D. So B and C can send to A the trust values that they have on D, which will be counted as recommended trust values for D.

We devise a general equation for the calculation of recommended trust. Consider a scenario where node w_z requests a context C_i from w_1 . The node w_1 (SP) needs to know the recommended trust value on w_z (SR) to make the context sharing decision. Let us assume that there are n paths ($p_1, \dots, p_i, \dots, p_n$) with a hop length greater than 1 from w_1 to w_z . We do not consider paths of length 1 in calculating recommended trust since those paths indicate the direct interaction records. We use those paths in calculating direct trust only

$$(1) T_{p_i} = \left(\frac{T(w_1, w_2) + T(w_2, w_3) + \dots + T(w_{z-1}, w_z)}{\lambda} \right) \times \left(1 - \frac{(\lambda-1) \times \psi}{10} \right)$$

where $w_1, w_2, \dots, w_z$ are the nodes on the path p_i from SP (w_1) to SR (w_z) and

$$(2) T(w_x, w_y) = \begin{cases} T_D(w_x, w_y, C_i), & \text{where } T_D(w_x, w_y, C_i) \neq \emptyset \\ T_{avg}(w_x, w_y), & \text{otherwise} \end{cases}$$

here, λ = hop distance between w_1 (SP) and w_z (SR),

ψ = Distance-based aging factor

The recommended trust value of w_1 on w_z in C_i is calculated as

$$(3) T_R(w_1, w_z, C_i) = \frac{\sum_{i=1}^n T_{p_i}}{n}$$

The term $\left(1 - \frac{(\lambda-1) \times \psi}{10} \right)$ has been used as a weight factor to satisfy the *distance-based aging* property mentioned in Section 4.2.

Let us explain the situation with the help of Fig. 4. Assume that A wants a recommendation value for B. The recommendation values for B follow the paths $\{A, E, F, B\}$, $\{A, E, F, D, C, B\}$, $\{A, G, H, F, B\}$, and $\{A, G, H, F, D, C, B\}$ but not the path $\{A, B\}$ which actually shows the interaction link of direct trust. The path length of a

recommendation is at least 2, and this path gets the highest weight since it is the shortest recommendation path. In our model we used which means for a path of hop length 2, 95% of the recommendation value is counted. The weight for the recommendation is decreased 5% with a single increment in the path length.

Criteria for choosing the value of ψ : Here we look for appropriate values for ψ based on situations. In situations where the environment is very sophisticated or the probability of the presence of malevolent nodes is quite high, a node may not value a recommender who is quite a bit away and only trusts its close neighbors. In these cases, a higher value of ψ should be chosen. The higher the value of ψ , the higher the decrement rate for the weight factor used in the recommended trust calculation. For example, if we choose $\psi = 1$ instead of 0.5 in our above example, then for a path of 2 hops, $90\% \left(1 - \frac{(2-1) \times 1.0}{10}\right) = 0.90$ of the recommendation value is counted. The decrement rate of the weight factor is 10% with a single increment in the path length which was 5% with $\psi = 0.5$.

Let us assume a scenario where SP needs the recommendation of SR. Assume that there are 5 paths from SP to SR with hop lengths 2, 3, 4, 5, and 6, respectively, and all the links (connecting line between 2 nodes) on all the paths have a trust value of 0.9. The results of different recommendation values for different paths (using Eq. (1)) and the recommendation trust (using Eq. (3)) have been summarized in Table 3 and Fig. 5.

Table 3. Recommended trust values with varying hops and ψ values.

ψ	2 hop	3 hop	4 hop	5 hop	6 hop	Rec. trust, T_R
0	0.9	0.9	0.9	0.9	0.9	0.9
0.5	0.855	0.81	0.765	0.72	0.675	0.765
1	0.81	0.72	0.63	0.54	0.45	0.63
1.5	0.765	0.63	0.495	0.36	0.225	0.495

Fig. 5. Comparison of recommendation values for different path length with different ψ values.

Though we used quite a high trust value (0.9) in the links of all the paths, the average trust value quickly decreases as the value for ψ increases. Depending on the situation, a value between 0.5 and 1 works fine. The average trust value decreases in a reasonable manner – neither too quickly nor too slowly.

7.2.1. Active recommendation

Sometimes a situation may occur where a node only believes the nodes with which it has direct interaction and wants recommendations only from them. From this perspective, the idea of an active recommendation is introduced, which is the average active recommendation value for a specific node considering only the minimum length recommendation paths (2 hops). In Fig. 4, if A wants the active recommendation for F, then only the

paths {A, B, F} and {A, E, F} will be counted. Similarly, in Table 3, the recommended values for hop 2 are active recommendation.

7.2.2. Passive recommendation

This is the recommendation that we usually think of. It considers all the paths (hop length ≥ 2) from source to destination. In general, we consider all the active recommendations to also be passive. However the reverse is not true for all passive recommendations. Consider a passive recommendation of F for A, according to Fig. 4. All the paths {A, B, F}, {A, E, F}, {A, B, C, D, F}, and {A, G, H, F} are considered.

7.3. Initial trust formation

Initial trust comes into play when the requester is either unknown to the provider for any context or only unknown for the requested context. By the term “unknown”, we denote that the provider has no direct experience, i.e., direct trust with the requester for the specific context. Suppose *SR* requests a service from *SP* in the context C_i . ITM (see Section 6.1.2) can compute the initial trust value in three steps as follows, if the provider has past experience with the requester:

Step 1.

A trust value t is estimated from the past experience of *SP* with the *SR* in other contexts, to be precise, from $T_D(SP, SR, C_j), \forall j \neq i$

Step 2.

Security factor ξ , for the context to be bootstrapped, is computed. We describe this term later.

Step 3.

Finally, ITM bootstraps trust for *SR* in the requested context as $T_I(SP, SR, C_i) = t \times \xi$.

Otherwise, in the case of a completely unknown requester, ITM sets the provider’s disposition value θ as the initial trust. The detail of the initial trust is described in Section 10.

In summary, the initial trust value for *SP* on *SR* for the context C_i is calculated as

$$T_I(SP, SR, C_i) = f(x) = \begin{cases} t \times \xi, & \text{where } \sum_{all contexts} T_D(SP, SR, C_j) \neq 0, \forall j \neq i \\ \theta, & \text{otherwise} \end{cases}$$

8. Recommended trust protocol

In this section, we focus on the establishment method of recommended trust in our model. A provider broadcasts a request message to get a recommendation for a specific requester (*SR*). Every device that receives the message forwards this message unless it reaches any device that has direct interaction with the *SR* or the maximum hop value diminishes to 1, whichever occurs first.

8.1. Message format

In our model, the devices use a common form of message to communicate with each other. The message format is shown in Fig. 6. *Type* indicates whether it is a request or reply. *Type* = 0 for a request and *Type* = 1 for a reply message. ID_{Req} is the request id. The *TS* field contains a timestamp to thwart a replay attack. The next four fields are the *IDs* of service provider, service requester, sender and receiver of the message, respectively. The hop value, *Hval*, is followed by the total trust value over the path, *Rval*.

Type	ID _{Req}	TS
ID _{SP}	ID _{SR}	
ID _S	ID _R	
Hval	Rval	

Fig. 6. Message format.

8.2. Active and passive recommendation protocol

For active and passive recommendation, we use two algorithms. Algorithms 1 and 2 give the pseudo code for the *request originator* (denoted by *SP*), the *request receiver* (denoted by *P*), and the *service requester* (denoted by *SR*), respectively.

8.2.1. Algorithm 1: RequestRecommendation

Algorithm 1 (see Fig. 8) has two phases – *requesting phase* and *calculation phase*. When *SP* wants to get a recommendation for a specific requester *SR*, it starts from phase 1 and forms the request. The *max_hop_value* is the maximum length of the path that *SP* wants to consider in the calculation of the recommended trust value for *SR*. The value may be device specific or a system parameter. For our model, we consider. When *SP* wants to consider only an active recommendation, it sets *max_hop_value* to 2. The optimal value for *max_hop_value* is discussed later in this section. The hops beyond this value need not be considered while determining recommended trust value. The service provider uses a broadcast address in the field of receiver host address so that the packet reaches all its neighbors. The provider now enters phase 2 where it listens to the replied messages until the timer goes off. Whenever *SP* receives a reply from its neighbor, it adds the trust value of the link with the *Rval* of the reply message. Then it calculates the total weighted recommended trust. *SP* keeps on listening and doing the accounting until the timer goes off. Finally, the provider computes the average recommended trust value for the *SR*.

We know that when a new device joins a network, the trust value that the service provider has on it is θ (see disposition value of the service provider, in Section 10). Suppose the service provider is neutral to the new device, i.e., $\theta = 0.5$. The portion of trust from 0 to 0.5 can be considered the path of negative trust level while the rest (0.5–1) as the path of positive trust level. If we receive a recommended trust value along a path which is less than or equal to 0.5, this is not going to help the requester node in building positive trust. We devise the algorithm in such a way that it does not decrease the overall recommendation value when considering a long path that has high trust values in all the links but poor recommendation values due to the weight factor. At the same time, we do not want to discard a path that generates a poor recommendation value due to malevolent behavior of the *SR*. Consider the following Fig. 7 to understand the scenario.

Fig. 7. A topology of devices with trust relationships.

Algorithm 1: RequestRecommendation: Request Originator SP

```

1. Function REQUESTRECOMMENDATION ( $ID_{SR}, ID_{Req}$ )
2. // Let  $B\_Addr$ ,  $T_{path}$ ,  $\lambda$  be broadcast address, total trust over path, and hop distance, respectively.
3. // Phase 1: Requesting phase
4.  $Hval \leftarrow \max\_hop\_value$ 
5.  $Rval \leftarrow 0$ 
6.  $ID_R \leftarrow B\_Addr$ 
7.  $msg \leftarrow < 0, ID_{Req}, T_{SP}, ID_{SR}, ID_{SR}, ID_{SR}, ID_R, Hval, Rval >$ 
8. Broadcast  $msg$ 
9. // Phase 2: Calculation phase
10.  $t \leftarrow StartTimer()$ 
11. while  $t \neq 0$  do
    a.  $msg_i \leftarrow$  received message from the  $i^{th}$  neighbor as a reply to  $msg$ 
    b.  $T_{path} \leftarrow msg_i.Rval + T(SP, i)$  //  $T(SP, i)$  is same as  $T(w_x, w_y)$  mentioned in equation (2)
    c.  $\lambda \leftarrow \max\_hop\_value - msg_i.Hval$ 
    d. // Let  $U_{rec}$ ,  $W_{rec}$  be unweighted and weighted form of recommended trust value
    e.  $U_{rec} \leftarrow T_{path} / \lambda$ 
    f.  $W_{rec} \leftarrow U_{rec} \times \left(1 - \frac{(\lambda-1) \times \psi}{10}\right)$ 
    g. if  $U_{rec} \geq \theta$  then
        i. if  $W_{rec} > \theta$  then
            1.  $T_{rec} \leftarrow T_{rec} + W_{rec}$  //  $T_{rec}$  is total weighted recommended trust
            2.  $path \leftarrow path + 1$  //  $path$  denotes total no of path
        ii. end if
    h. else
        i.  $T_{rec} \leftarrow T_{rec} + W_{rec}$ 
        ii.  $path \leftarrow path + 1$ 
    i. end if
12. end while
13. if  $path = 0$  then
    a.  $T_R \leftarrow 0$  //  $T_R$  is average recommended trust value
14. else
    a.  $T_R \leftarrow T_{rec} / path$ 
15. end if
16. return  $T_R$ 

```

Fig. 8. Algorithm 1 RequestRecommendation.

Device G (SR) sends a request for a service to A (SP) which requires a trust value of 0.7. Since A does not know G (which means G has a trust value of 0.5), A requests a recommendation. Note there are three paths from A to G which are {A, B, C, D, E, F, G}, {A, H, G}, and {A, I, J, G} with path lengths of 6, 3, and 2, respectively. Here, the value of ψ is considered 0.5 so that the weight factor is 0.95 for a path of 2 hops, 0.90 for a path of 3 hops and so on (see Section 7.2). See (Table 4).

Table 4. Average recommended trust for G (SR) to A (SP) from Fig. 7.

Path	Unweighted recommended trust, U_{Rec}	Hop length, λ	Weight factor, $\left(1 - \frac{(\lambda - 1) \times \psi}{10}\right)$	Weighted recommended trust, W_{Rec}	Avg. recommended trust, T_R
{A, B, C, D, E, F, G}	$(0.7 + 0.7 + 0.7 + 0.6 + 0.6 + 0.6)/6 = 0.65$	6	0.65	0.49	0.55
{A, H, G}	$(0.6 + 0.3)/2 = 0.45$	2	0.95	0.43	
{A, I, J, G}	$(0.8 + 0.7 + 0.7)/3 = 0.73$	3	0.90	0.66	

For path {A, B, C, D, E, F, G}, the algorithm calculates the unweighted recommended trust value as 0.65 which is greater than 0.5. Thus, the unweighted recommended value calculated for this path indicates positive trust level for the node G. But when this trust value is multiplied by the weight factor, we get a value of 0.49 which is beyond the positive trust level. This path will have a negative impact on the overall trust value of A on G which is already 0.5. In that case, G will appear as a non-trusted node to A. So we discard this path from calculation of recommendation trust because G cannot be made non-trusted for being a distant node. Meanwhile, for path {A, H, G} the unweighted recommended trust value is 0.45 which indicates that the trust level is negative for G even without considering a weight factor. Multiplying this value (0.45) by the weight factor (0.95) generates a weighted recommended trust of 0.43 for this path. Though the trust value is less than 0.5 we consider this path to take into account the malevolent behavior of G. Finally path {A, I, J, G} is also counted because it generates a weighted recommendation value of 0.66 which is greater than 0.5. The average recommended trust value gets the value of 0.55.

In summary, two things happen in the case of a node with an already positive trust level. First, the paths which yield positive recommendations before and after applying the distance-based aging are considered. Second, the paths with negative recommendation even before applying the distance-based aging are counted towards determining the average recommendation. However, for an already malevolent node, every path is counted and no such considerations are necessary.

Optimal hop length. In our model, a device has the flexibility to define the maximum length of a recommendation path through the 'Hop value' (*Hval*) field of the request packet. Here, we provide some guidelines for choosing the value of *Hval*.

1. For a very small *Hval*, the response time required for calculating the recommended trust will be short. However it may not consider several possible recommendation paths with hop lengths greater than that specified in the *Hval*. So there is a tradeoff between response time and accuracy of the recommended trust for the specified *Hval*.
2. The upper bound of the *Hval* value is related to the weight factor and disposition value (see Section 10)

$$Hval \leq \frac{\text{Disposition Value}, \theta}{\text{Decrement rate of weighting factor per hop}}$$

According to our model, the equation becomes

$$Hval \leq \frac{\theta}{\left(1 - \frac{(\lambda - 1) \times \psi}{10}\right) - \left(1 - \frac{\lambda \times \psi}{10}\right)}$$

In our scenario, we consider $\theta = 0.5$ and $\psi = 0.5$. So the decrement rate of the weight factor is 5%. Therefore the maximum value for *Hval* becomes,

$$(4) Hval \leq \frac{0.5}{0.05} = 10$$

Our algorithm is not going to consider any recommendation path with the higher trust values in its intermediate links but generating a poor recommendation (less than or equal to 0.5) for this path when it is multiplied by a small weight factor (as the recommendation path has a long length). Therefore, it is not meaningful to use such a high *Hval*, because some recommendation paths with a length that is within the specified limit of *Hval* will generate an overall recommendation of less than or equal to 0.5.

Consider a scenario where device A requires a recommendation for device B. Consider a path of length 10 from A to B and assume that all the intermediate links have a trust value of 1.0. According to Eq. (1) the recommendation value for B through this path will be

$$T_{p_i} = \left(\frac{1 + \dots + 1}{10} \right) \times \left(1 - \frac{(10 - 1) \times 0.5}{10} \right) = 0.55$$

Now consider paths from A to B with length greater than 10. The recommendation value will be less than or equal to 0.5 irrespective of the trust values of the intermediate links, and this path will be discarded by our algorithm. This indicates that any *Hval* value greater than 10 will generate the same overall recommendation trust value which will be generated when *Hval* = 10. But the increased *Hval* value will certainly increase the response time. So, in this scenario, 10 is the upper limit thus proving Eq. (4).

8.2.2. Algorithm 2: ReplyRecommendation

Each receiver invokes the *ReplyRecommendation* function (see Fig. 9) whenever he receives any request. If it is a duplicate request, the receiver simply discards it. Otherwise the receiver proceeds to the next step. As the request already includes the hop between him and the request forwarder, he reduces the *Hval* by one. If the receiver discovers that the service requester is his peer, he just forms the as shown in Algorithm 2 and sends the reply message back to the request forwarder. In another case, he may simply forward the message to his neighbors to collect recommendations from them. Upon receiving a reply from any peer, he adds the trust value of that link with *Rval* of the packet and changes the sender and receiver addresses of the packet to send it back to the node that has forwarded the request to him. Eventually the request originator receives all the recommended trust values.

Algorithm 2: ReplyRecommendation: Request Receiver
1. Function REPLYRECOMMENDATION (<i>msg</i>)
2. // Let <i>B_Addr</i> be broadcast address.
3. // Let <i>Q</i> be the request forwarder
4. if <i>msg</i> is duplicate then
a. discard <i>msg</i>
b. return
5. end if
6. <i>Hval</i> ← <i>msg.Hval</i> - 1
7. if <i>SR</i> is neighbor of <i>P</i> then
a. <i>msg_p</i> ← < 1, <i>ID_{rec}</i> , <i>TS_p</i> , <i>ID_{SP}</i> , <i>ID_{SR}</i> , <i>ID_p</i> , <i>ID_Q</i> , <i>Hval</i> - 1, <i>T(P, SR)</i> >
// <i>T(P, SR)</i> same as equation (2)
b. send <i>msg_p</i> back to <i>Q</i>
8. else if <i>Hval</i> = 1 or no neighbor remains then
a. reply <i>Q</i> with an ACK message
9. else
a. <i>msg</i> ← < 0, <i>msg.ID_{Req}</i> , <i>TS_p</i> , <i>msg.ID_{SP}</i> , <i>msg.ID_{SR}</i> , <i>ID_p</i> , <i>B_Addr</i> , <i>Hval</i> , <i>msg.Rval</i> >
b. Broadcast <i>msg</i>
c. <i>MSG</i> is set of messages replied back to <i>P</i>
d. for all <i>msg_i</i> ∈ <i>MSG</i> , where <i>msg_i</i> is received from <i>ith</i> neighbor of <i>P</i> do
i. <i>msg_i.ID_R</i> ← <i>ID_Q</i>
ii. <i>msg_i.ID_S</i> ← <i>ID_p</i>
iii. <i>msg_i.Rval</i> ← <i>msg_i.Rval</i> + <i>T(P, i)</i> // <i>T(P, i)</i> same as equation (2)
iv. send <i>msg_i</i> back to <i>Q</i>
e. end for
10. end if

Fig. 9. Algorithm 2 ReplyRecommendation.

9. Direct trust protocol

In this section we describe the details of the behavioral model along with the process of direct trust evolution.

9.1. Categories of behavior

Devices in a pervasive environment may behave differently from one another. We, therefore, classified the behavior of devices in four different categories which is shown in Fig. 10.

Fig. 10. Classification of behavior.

9.1.1. Positive behavior

Positive behavior refers to some type of rewarding attitude. This type of behavior helps in increasing the direct trust value, so over the course of time, a positively behaving device may expect its direct trust value to be increased.

9.1.2. Negative behavior

The opposite of positive behavior is negative behavior, which results in punishment. One common example of negative trust can be delay in providing a service.

9.1.3. Neutral behavior

Some behaviors may indicate neither positive nor negative attitude and are characterized by neutral behavior (for example, a node requests a service and is denied). No action is taken in case of neutral behavior.

9.1.4. Malevolent behavior

The malicious attitude of devices is termed malevolent behavior, such as attacks, and is punished severely. Negative behavior is separated from malevolent behavior because, unlike negative behavior, malicious behavior is a clear indication that the device is malicious.

For malevolent behavior, an alert message is broadcasted. The alert message results in an immediate deposition of the malevolent node whereas negative behavior will just result in decrease of trust value. But all positive behaviors should not be treated equally since these behaviors may vary widely in terms of weight and impact. This statement is equally true for negative behaviors. Thus, we classified both positive and negative behaviors into three categories.

As device behavior may vary depending on the application, it is always difficult to develop a general behavioral model. Our classification takes a step forward for a general purpose behavioral model.

9.2. Overview of behavioral model

Fig. 11 shows the positive and negative behaviors of devices with their distribution of trust among 'Low', 'Medium' and 'High' categories.

Fig. 11. Classification of trust based on behavior.

We devised equations to calculate the satisfaction level (μ) based on a weighted average technique. The notations used in this equation are summarized in Table 5. The equation for satisfaction level calculation is:

$$\mu = \frac{B(positive) + B(negative)}{x + y + z + o + p + q}$$

where

$$B(positive) = \left(\sum_{i=1}^x (Bh_{(+)i} \times k) + \sum_{j=1}^y (Bm_{(+)j} \times v) + \sum_{k=1}^z (Bl_{(+)i} \times \pi) \right)$$

and

$$B(negative) = \left(\sum_{m=1}^p (Bm_{(-)m} \times \varsigma) + \sum_{l=1}^o (Bh_{(-)l} \times \varpi) + \sum_{n=1}^q (Bl_{(-)n} \times \tau) \right) / \eta$$

Table 5. Notations used in the calculation of satisfaction level.

μ	Satisfaction level (the satisfaction level, actually indicates the direct trust value for this latest interaction)
$B(positive)$	Sum of all positive behaviors
$B(negative)$	Sum of all negative behaviors
$Bh_{(+)}, Bm_{(+)}, Bl_{(+)}$	High, medium, and low positive behaviors, respectively
$Bh_{(-)}, Bm_{(-)}, Bl_{(-)}$	High, medium, and low negative behaviors, respectively
η	Priority level of the context associated with this interaction
κ, v, π	Values associated with each high, medium, and low category of positive behavior
ϖ, ς, τ	Values associated with each high, medium, and low category of negative behavior
x, y, z, o, p, q	$\{x, y, z, o, p, \text{ and } q\}$ denote the number of high, medium, and low positive and corresponding negative behaviors occurring for this interaction

We divided the sum of negative behaviors by η , assuming that the priority of contexts is a positive integer value. This is because when a highly sophisticated context receives a negative behavior, that incident should receive more punishment compared to a situation where the same negative behavior is applied to a trivial or less prioritized context. In our model, we used $k = 1, v = 0.8, \pi = 0.6, \varpi = 0.4, \varsigma = 0.2, \tau = 0$, respectively. These variables are used to permit modifications of these values based on the application requirements. All these variables can be used to adjust the rate of change of a direct trust value.

9.3. Properties of behavioral model

Some of the important properties (P1, P2, P3, and P4) of our behavior model are as follows:

P1.

Our behavioral model ensures a small reward for positive behavior but increasing punishment for negative behavior. Our motivation behind such logic is proposed by Baier (1986) who states, “Trust is much easier to maintain than it is to get started and is never hard to destroy”.

P2.

The value of a positive behavior is not dependent on the priority level of context. As it is expected that all behaviors will be positive, the positive behavior value does not change with the priority level of associated contexts. For negative behavior, the level of punishment varies with the priority level of the associated context.

P3.

In our behavioral model, a single interaction may consist of any combination (0 or more) of positive, negative, neutral, and malevolent behaviors.

$$I(interaction) = \{(b_{+1}, b_{+2}, \dots, b_{+w}), (b_{-1}, b_{-2}, \dots, b_{-x}), \{ (b_{n1}, b_{n2}, \dots, b_{ny}), (b_{m1}, b_{m2}, \dots, b_{mz}) \}$$

Here, b_+ , b_- , b_n , and b_m represents positive, negative, neutral, and malevolent behaviors, respectively. To the best of our knowledge, this is the first model that considers an interaction from this vector point of view. To denote the usefulness of this strategy, consider a scenario where device A gets a context C_k from B. Suppose A drew attention in two behaviors: successful completion of the context and time required for the context sharing. The result from device A on these two criterions might be different, for example, highly positive for successful completion and low positive on time required. Thus, instead of denoting the overall satisfaction of A as positive, this strategy provides the opportunity to more accurately categorize separately each criteria of a context and their behaviors.

P4.

Special attention has been given to handling an event which may appear to be wrong but difficult to prove that it had been done deliberately, for example, repeatedly requesting the same service within a short duration. We put such events in the low negative category (with a trust value of 0.4) to ensure that these types of behaviors do not receive much punishment.

9.4. Evolution of direct trust

The value of direct trust changes when device A wants a context from C_i B and receives it. This evolution of direct trust depends on the satisfaction level of the interaction between the interacting devices. If A has a previous direct trust value with B for context and is the satisfaction level for the latest interaction, then the new trust value will be the average of these two. This process supports the characteristic 'time-based aging' (Section 4.2.) since we are giving 50% weight on the latest interaction outcome and the rest on the direct trust value incurred from previous interaction records. If there is no previous direct trust value then this becomes the current direct trust value. Therefore, the equation for direct trust evolution is:

$$(5) T_D(A, V, C_i)_{new} = \begin{cases} \frac{T_D(A, B, C_i)_{old} + \mu}{2}, & \text{where } T_D(A, B, C_i)_{old} \neq \phi \\ \mu, & \text{otherwise} \end{cases}$$

9.5. Simulation results of direct trust

We generated direct trust by simulating it based on the values of positive and negative behaviors of different categories. Next we present graphs that describe simulation results.

In Fig. 12a–d below, the X-axis shows direct trust. In Fig. 12a and b, the increment and decrement of trust values for positive and negative behaviors of different categories are shown.

Fig. 12. (a–d) Simulation results of direct trust.

We tested our model for negative behavior when applied to different priority levels. As seen in Fig. 12c, when the same medium negative behavior is applied to different priority levels, the higher the priority level (PL3) the faster the rate of decrement of the direct trust value. Here PL(1) means context with priority level 1, the lowest priority level.

According to our distribution of trust values in different categories, a low positive behavior converges to 0.6 and a low negative behavior converges to 0.4 (for a context with priority level 1). From Fig. 12d, we find, this convergence rate is the same for a context with priority level 1. Here, the low positive behavior curve converges from 0 to 0.6 at around the 8th consecutive low positive behavior and, at the same time, the low negative behavior curve converges from 1 to 0.4. But this convergence rate becomes much higher when low negative behavior is applied to a context of priority level 2 and 3, as shown in Fig. 12d.

The direct trust value becomes 0.4 only at the second consecutive low negative behavior when the priority level is 2. When two consecutive low negative behaviors are applied to a context of priority level 3, the direct trust value goes from 1 to nearly 0.35. This proves the desirable property 1 mentioned in Section 5.3 which states ‘Small reward for positive behavior but increasing punishment for negative behavior’. Here, $Bl(-)PL(1)$ means low negative behavior applied to context with priority level 1.

10. Initial trust protocol

Our initial trust scheme takes into account the security level of the services while establishing trust. The two major characteristics of the initial trust model are as follows:

Sensitive to context security: The initial trust value for a context should be inversely proportional to its security level. In other words, the initial trust value of a context with a high security level should be lower than that of a context with a low security level

$$T_I(SP, SR, C_i) < T_I(SP, SR, C_j), \text{ if } SL_j > SL_i$$

Able to adapt disposition of user: The disposition value indicates a user’s (service provider’s) personality in sharing or providing services. Each device that runs ITM can adjust the behavior of SP by changing the disposition value. The modified value reflects the user’s change in preference.

We categorize services based on their need for security. It means that each service is associated with an integer value representing its *security level*. We use this numerical value to bootstrap the initial trust for the corresponding service. It is obvious that not all services of a device need to have the same security level.

Similarly, the same service in different devices, whether within the same network or not, may have different security levels. Each *SP* maintains a table of its services along with the security level. It does not have to provide services for all security levels. In our model, we assume the range of security levels of contexts is $1 \sim 10$.

Security factor ξ is computed for bootstrapping an unknown trust according to Eq. (6). It is introduced to integrate the security issues of contexts in initial trust calculation. In other words, ξ is the weight factor that makes our model sensitive to context security levels. If a *SR* requests a service for C_i from *SP*, then ξ is calculated as follows:

$$(6) \xi = 1 - \frac{SL_i - 1 \times \theta}{SL_{max}}$$

where SL_i = security level of context C_i and θ = disposition value; we describe it later. SL_{max} = maximum security level and in this paper it is 10. For example, if $SL_i = 10$ for context C_i , then θ is 0.55, whereas for $SL_i = 0.5$, θ is 0.8. The higher the security level of a context, the more the security factor is reduced by Eq. (6). As a result, it adapts the bootstrapped trust value by Step 3 (see Section 7.3).

Disposition value θ represents a numerical value that interprets the *disposition* of a service provider to a new requester. The range of interpreted values of θ is $[0, 1]$. We label three positions on the θ scale as the nature of user: pessimist, optimist, and neutral (see Fig. 13).

Fig. 13. Disposition values of a service provider.

By assigning a value to θ , a service provider can control his own attitude towards new devices that are about to join the network or the devices that are new to a context. Depending on the *SP*'s mobility, a disposition value can be fixed or variable. The value assignment to θ , in fact depends on the environment or network to which the user belongs. A user can adaptively behave by changing his disposition value. Fig. 14 shows how the security factor ξ changes with the security level of contexts for different values of θ . With the increase in the security level, ξ remains constant for $\theta = 0$. It represents an optimistic behavior of a service provider, as estimated trust t becomes the bootstrapped trust. Given $\theta = 0.5$, ξ drops by 0.05 with a single level increase in the security of service. It reflects *SP*'s neutral behavior, as estimated t is moderately reduced with increasing service security levels. *SP* shows its pessimistic attitude when $\theta=1$ as ξ drops by 0.1 with a single level increase. As a result, the bootstrapped trust value is drastically reduced.

Fig. 14. Comparison of security factor for different security levels due to different disposition value of SP .

Suppose service provider A is in a trusted network such as a school or office. Here, A may want to share his services with other devices without any type of security constraint. A can easily set $\theta = 0$ to behave as an optimist. After leaving his office, A travels in a subway train to return home. Obviously, a subway train is not a fully trusted environment. Therefore he changes θ from 0 to 0.5 to reflect his neutral behavior. Over the weekend, when he goes to a marketplace or multiplex, he can assign $\theta = 1$, the pessimistic attitude, as surrounding people as well as devices are unknown. Finally, we assume that an interface is provided in the devices that can adaptively control the disposition of a user in case of sharing or providing services in a pervasive environment.

To demonstrate how ITM bootstraps trust, we consider device A provides services to $\{B, C, D, E\}$ in the contexts $\{C_1, C_2, C_3, C_4\}$. Suppose B sends a request to A for the context C_2 . SP finds that B is new to this context and therefore it needs to calculate the initial trust value for B in the context C_2 . ITM bootstraps trust for devices in contexts by following the steps we mentioned earlier (see Section 7.3).

Step 1.

At this step, trust t is derived using the mechanism mentioned in Quercia et al. (2007). Here, the trust matrix is decomposed into three other matrices using *singular value* decomposition (Abdi, 2007). It determines the contexts in which B has already been served by A. Based on some predefined cases and by performing a sequence of steps, t is estimated. A detail analysis of trust estimation process can be found in Quercia et al. (2007).

Step 2.

The security level of the context now needs to be incorporated with the estimated trust. We do this by determining the security factor ξ which integrates the security issues in initial trust assignment. This security factor ξ works as a weight factor that makes our initial trust model sensitive to context security levels. ITM calculates ξ from Eq. (6) by using the security level of C_2 , i.e., SL_2 and θ , the disposition of A.

Step 3.

Having estimated the trust value t and security factor ξ for C_2 , ITM computes the initial trust value of A on B in the context C_2 using the process mentioned in Subsection 7.3. This is the initial trust value $T_I(A, B, C_2)$. In this way, an SP can derive an initial trust value of an SR for an unknown context. Therefore, the incorporation of a context security level in trust bootstrapping allows our initial trust model to be sensitive to the security requirements of different contexts.

11. Overall trust calculation

In our model, each device maintains a trust table. In the table for each context C_i a device (requester) has two fields, direct trust (T_D) and recommended trust (T_R). Suppose for some context C_i , device A may have T_D but no T_R for device B. This can occur when B has requested a trivial and very low priority context, and A has not asked for a recommendation for a very low priority service and instead provided that service directly. Conversely, B can request a context with high priority level. Since A had no prior interaction record with B, A asked for a recommendation about B for that context. After getting the recommendation value, A denied to provide B with that context since the trust value was too low. At this stage, A has T_R for B but no T_D . In calculating overall trust, if the T_D field for C_i is empty, then DT is assumed to be the $T_I(A, B, C_i)$ (see Eq. (5)). The overall trust value of A on B for context C_i is calculated as

$$T(A,B,C_i) = \begin{cases} \frac{\eta \times T_D(A,B,C_i) + T_R(A,B,C_i)}{\eta + 1}, & \text{where } T_D \neq \emptyset \text{ and } T_R \neq \emptyset \\ T_D(A,B,C_i), & \text{where } T_D \neq \emptyset \text{ and } T_R = \emptyset \\ \frac{\eta \times T_I(A,B,C_i) + T_R(A,B,C_i)}{\eta + 1}, & \text{where } T_D = \emptyset \text{ and } T_R \neq \emptyset \\ T_I(A,B,C_i), & \text{where } T_D = \emptyset \text{ and } T_R = \emptyset \end{cases}$$

Here, η denotes the priority level of C_i .

When the value $T_D(A,B,C_i)$ is available in the calculation, we multiplied this value by η because when a node requests a higher priority context than is normal, the SP will give more weight to its direct experience and less weight to the recommended value.

$$T(A,B) = \frac{\sum_{i=1}^k (T_D(A,B,C_i) + T_R(A,B,C_i))}{m + n}$$

Here k denotes the total number of contexts, and m and n denote the number of contexts for which A has T_D and T_R values, respectively, for B.

The overall trust information is provided to the decision module which makes a final decision based on the input and some defined policies. These policy definitions will depend on the application which will use the trust model.

12. Handling malicious recommendation

Circumventing a malicious recommendation can highly influence the performance of any trust model. However, a majority of trust models have avoided this by passing this responsibility on to an authentication mechanism. Since an authentication mechanism does not allow any malicious nodes to join the network, there is no concern about malicious recommendations. But real life scenarios and facts prove otherwise. Malicious nodes may exist even after adopting authentication techniques. We propose a simple protocol for restricting a malicious recommendation and gradually weeding out all malicious nodes. The MRH component of our trust model performs this protocol to deal with malicious recommendations. This protocol is based on a Student's t -Distribution (Gosset, 2008, Abramowitz and Stegun, 1972).

12.1. Issues regarding malevolent nodes

One important issue regarding a malicious recommendation is what happened when a node is detected as malevolent in the network. An alert message will be broadcast upon detecting a node's malevolent behavior. The alert message will be handled in the same way as interactions between peer nodes with positive trust levels. Even if a malicious node was previously assigned a positive trust value, once detected as malevolent, it is immediately deposited from the list and permanently treated as non-trusted. A node may consider more than one alert message from trusted entities to permanently mark a malevolent node as non-trusted. There is no way to readapt the node. The rehabilitation of malicious nodes remains for future research and is not addressed in our paper. Here we assume that the number of benevolent nodes is much higher than that of malicious nodes. Flooding of malicious recommendations cannot occur since a node can send only one recommendation about another node.

First, we consider the situation where the number of recommendations is few. We assume the standard deviation is known. To evaluate our proposal we have fitted the control chart method. This was first used by

Walter Shewart to determine whether the process metrics are stable. In the evaluation of metric data, Zultner proposed two types of control charts: (1) moving range control chart and (2) individual control chart (Zultner, 1999).

Let us take a data set of 10 recommendations which are as follows: {0.9, 0.88, 0.85, 0.75, 0.78, 0.8, 0.73, 0.2, 0.87, 0.9}. A moving range control chart has been generated by plotting the difference between successive observations. In Fig. 15a, mR bar denotes the average of moving range values while UCL indicates the Upper Control Limit. Here, UNPL and LNPL denote Upper Natural Process Limit and Lower Natural Process Limit, respectively. In the individual control chart, A_m = mean of the observed recommendation values, and σ = standard deviation.

(a) Moving range control chart for the observed recommendation values

(b) Individual control chart for the observed recommendation values

(c) Plot result for recommendation values using t -Estimate:Mean

Fig. 15. (a–c) Illustration of different types of chart.

Here we set the upper limit to 1 (maximum possible recommendation value) and the lower limit to 0 (maximum possible recommendation value). From Fig. 15b, we find that all the recommendation values lie within the region defined by $+1\sigma$ and -1σ deviation lines about the mean with the exception of 0.2 which lies even below the LNPL. We discard any recommendation value which is not within the interval (0.590637 (value of -1σ), 0.941363 (value of $+1\sigma$)). (See Table 6)

Table 6. Mean, standard deviation, and upper–lower limits for different confidence values.

Mean		0.766
Standard deviation		0.2082
Confidence level (%)	LCL	UCL

90	0.645381	0.886619
95	0.61715	0.91485
99	0.55216	0.97984

For a large number of recommendations, we refer to Fig. 15c. Here we use the approach of Student's t -Distribution. This approach is used when the standard deviation is not known. For example, let us take a data set of 10 recommendations which are as follows: {0.9, 0.88, 0.85, 0.75, 0.78, 0.8, 0.73, 0.2, 0.87, 0.9}. In this distribution, a confidence value of 90% indicates that the mean of the data set will be in the interval specified by UCL (Upper Confidence Limit) and LCL (Lower Confidence Limit). In Fig. 15c, we show the following results for different confidence levels.

12.2. Approaches of malicious recommendation handling process

The information from the interval region is used in two ways, and we name these *approach 1* and *approach 2*.

We discard any recommendation value which is outside this interval region; we take the mean of the rest of the values as the overall recommendation value. The optimization is readily observable. The old overall recommendation value of 0.766 is replaced by a much accurate overall recommendation value of 0.828889 $((0.9 + 0.88 + 0.85 + 0.75 + 0.78 + 0.8 + 0.73 + 0.87 + 0.9)/9)$.

If the recommendation value given by a specific device falls outside the interval region for more than a specified threshold value (Δ), the device will be discarded.

Approach 1:

In the first approach, a device will be discarded if

$$w_1\alpha + w_2\beta + w_3\lambda \geq \Delta,$$

where w_1, w_2 , and w_3 are weight factors with $w_1 < w_2 < w_3$

α = number of times a device sends recommendations which falls in the interval $(+1\sigma, +2\phi]$ or $(-1\sigma, -2\phi]$. This is computed by the unit shown in Fig. 2 in 6.1.

β = number of times a device sends recommendations which falls in the interval $(+2\sigma, UNPL]$ or $(-2\sigma, LNPL]$. This is computed by the unit shown in Fig. 2 in 6.1.

δ = number of times a device sends recommendations which has value greater than or equal to UNPL or lower than or equal to LNPL.

Approach 2:

In the second approach, a device will be discarded if

$$\eta \leq \Delta,$$

where η = number of times a device sends a recommendation which falls outside the interval defines by UCL and LCL.

The value of η is dependent on the level of importance of the environment. In a highly sophisticated environment, a very small number of out of range recommendations will put a device out of the network, whereas more tolerance can be shown in a less sophisticated environment.

13. Evaluation

We evaluated our proposed trust model in the following ways:

- (1) Implementation of the prototype.
- (2) Measurement of the performance with remaining battery power.
- (3) Performance measurement using simulation.

13.1. Simulation results

We simulated our model using OMNeT++ to evaluate several issues such as scalability, timing requirement, etc. One of our objectives was to find the required time when the recommendation path gets longer. We simulated a scenario where node B (SR) requests a service from A (SP). We assumed that there are five paths from A to B with path lengths of 2, 3, 4, 5, and 6, respectively. We measured the required time for getting recommendations along these paths. We repeated this computation five times and calculated the average recommendation time along different paths. The results are shown below (Fig. 16).

Fig. 16. Timing requirement for receiving recommendation packets for varying path lengths.

13.2. Implementation of the prototype

We implemented our trust model and a service-sharing application that uses the underlying trust model. 802.11b protocol has been used on Dell Axim X50v PDAs. We used VC++.NET Compact framework for the implementation. Some screen shots of our implementation are shown in Fig. 17.

Fig. 17. (a–h) Some screen shots from our implementation.

13.3. Performance measurement using battery power

Our model consumes a small amount of battery power. In order to prove this statement, we measured the remaining amount of battery power in two scenarios. (1) When our model is not running on a PDA. (2) When our model is running on a PDA. As we see from the graph, the decrement rate of remaining battery power is almost the same in both scenarios (Fig. 18). We did not begin from the same starting point but adjusted this because the two curves almost merge together, making the comparison difficult to observe.

Fig. 18. Rate of decrement of remaining battery power before and after running our model.

14. Conclusion and future works

Secure communications cannot occur between entities in a pervasive environment unless a trust relationship has been established. The vast majority of such relationships use identity as a basis for establishing trust. A flexible trust model, built upon a formal basis, is essential in providing secure communications in the future world of pervasive computing. In this paper, we present a context aware reputation-based trust model. To the best of our knowledge, this is the first model that provides a mechanism for handling multi-hop recommendations. At the same time, it integrates a dynamic initial trust for unknown devices which is based on the security level of contexts. As an attempt to simulate human trust, we augmented our model with both time and distance-based aging of trust values. The security level of a context has been considered to make our model more liable to prioritized contexts. The security level of context is also considered while computing initial trust. A model for separating malicious recommendations and gradually weeding out malicious nodes has been incorporated in our model. We have also introduced the disposition, used with initial trust establishment, in this paper. Furthermore, we introduce in this paper an upper limit on selection of hops, malicious recommendation handling, and simulated results for the behavioral model. For evaluation purpose, a simple resource sharing application operating on our trust model has been implemented. We tested the scalability issue of our model using OMNeT++ simulator. The discussion on vulnerability of the model in the face of different types of attacks has been kept out of the scope of this paper. We have discussed this issue in detail, with a solution, in our previous paper (Haque and Ahamed, 2007).

In the future, we plan to incorporate a risk assessment model with this trust model. The path discovery issues and the analysis of communications overhead in a hop-based recommendation protocol are considered our future research work. We are also in the process of incorporating this trust model in a healthcare and elderly care application on a handheld device. We will be able to perform a very realistic usability study with persons who are real patients and users of the application.

Acknowledgement

This project is partially supported by Wehr grant and Way Klinger award. Microsoft has donated Smart phones/PDAs for this project.

References

- Abramowitz and Stegun, 1972. Abramowitz, M., Stegun, I.A., 1972. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York.
- Abdi, 2007. H. Abdi. **Singular value decomposition (SVD) and generalized singular value decomposition (GSVD)**. N.J. Salkind (Ed.), *Encyclopedia of Measurement and Statistics*, Sage, Thousand Oaks, CA (2007), pp. 907-912
- Almenarez et al., 2004. Almenarez, F., Marin, A., Campo, C., Garcia, C., 2004. PTM: a pervasive trust management model for dynamic open environments. In: *Proceedings of the pervasive security, privacy, and trust (pspt 2004)*, Massachusetts. <http://jerry.clab.de/ubisec/publications/PSPT04_PTM.pdf> (accessed April 2007).
- Almenarez et al., 2006. Almenarez, F., Marin, A., Diaz, D., Sanchez, J., 2006. Developing a model for trust management in pervasive devices. In: *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, p. 267.

- Ahamed et al., 2008a. Ahamed, S.I., Monjur, M., Islam, M.S., 2008a. CCTB: context correlation for trust bootstrapping in pervasive environment. In: *Proceedings of the 6th Annual Conference on Privacy, Security, and Trust (PST 2008)*, Fredrickton, NB, Canada, pp. 167–174.
- Ahamed et al., 2008b. Ahamed, S.I., Hoque, E., Rahman, F., Zulkernine, M., 2008b. Towards secured trust bootstrapping in pervasive computing environment. In: *Proceedings of the 11th IEEE High Assurance Systems Engineering Symposium (HASE 2008)*, China, pp. 89–96.
- Baier, 1986. A. Baier. **Trust and antitrust**. *Ethics*, 96 (2) (1986), pp. 231-260.
<http://dougaskaralis.com/files/trust_ans_antitrust.pdf>
- Blaze et al., 1996. Blaze, M., Feigenbaum, J., Lacy, J., 1996. Decentralized trust management. In: *Proceedings of the IEEE Symposium on Research in Security and Privacy*, number, pp. 164–173.
- Buchegger and Boudec, 2004
- Buchegger, S., Boudec, J.Y., 2004. A robust reputation system for P2P and mobile ad-hoc networks. In: *Second Workshop on Economics of P2P Systems*, Boston.
- Basu and Callaghan, 2005. Basu, J., Callaghan, V., 2005. Towards a trust based approach to security and user confidence in pervasive computing systems. In: *IEE International Workshop, Intelligent Environments 2005 (IE05)*, UK, pp. 223–229.
- Gambetta, 1988. D. Gambetta. **Can we trust?** D. Gambetta (Ed.), *Trust: Making and breaking cooperative relations*, Basil Blackwell, Oxford (1988), pp. 213-237
- Garfinkel, 1995. S. Garfinkel. **PGP: Pretty Good Privacy**. O'Reilly & Associates Inc. (1995)
- Guha and Kumar, 2004. R. Guha, R. Kumar. **Propagation of trust and distrust**. *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*, ACM Press (2004), pp. 403-412
- Gosset, 2008. W.S. Gosset. **The probable error of a mean**. *Biometrika*, 6 (1) (2008), pp. 1-25
- He et al., 2004. He, R., Niu, J., Yuan, M., Hu, J., 2004. A novel cloud-based trust model for pervasive computing. In: *The Fourth International Conference on Computer and Information Technology (CIT'04)*, pp. 693–700.
- Huebscher and McCann, 2005. Huebscher, M.C., McCann, J.A., 2005. A learning model for trustworthiness of context-awareness services. In: *The Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05)*, pp. 120–124.
- Haque and Ahamed, 2007. Haque, M., Ahamed, S.I., 2007. An omnipresent formal trust model (FTM) for pervasive computing environment. In: *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, pp. 49–56.
- Ismail, 2002. Ismail, R., 2002. The beta reputation system. In: *Proceedings of the 15th Bled Conference on Electronic Commerce*. <<http://sky.fit.qut.edu.au/~josang/papers/JI2002-Bled.pdf>> (accessed April 2007).
- Josang, 1996. Josang, A., 1996. The right type of trust for distributed systems. In: *New Security Paradigms 96 Workshop*.
- Josang and Knapkog, 1998. Josang A., Knapkog, S.J., 1998. A metric for trusted systems. In: *Proceedings 21st NIST-NCSC National Information Systems Security Conference*, pp. 16–29.
- Josang, 1999. Josang, A., 1999. An algebra for assessing trust in certification chains. In: *Proceedings of the Network and Distributed Systems Security (NDSS'99)*.
- Jameel et al., 2005. Jameel, H., Hung, L.X., Kalim, U., Sajjad, A., Lee, S., Lee, Y.K., 2005. A trust model for ubiquitous systems based on vectors of trust values. In: *Seventh IEEE International Symposium on Multimedia*, pp. 674–679.
- Kagal et al., 2001a. Kagal, L., Finin, T., Joshi, A., 2001a. Trust-based security in pervasive computing environments. In: *IEEE Computer*, vol. 34, pp. 154–157.
- Kagal et al., 2001b. Kagal, L., Finin, T., Joshi, A., 2001b. Moving from security to distributed trust in ubiquitous computing environments. In: *IEEE Computer*.
<<http://www.cs.umbc.edu/~finin/papers/ieee01/ieee01.pdf>> (accessed April 2007).

- Kamvar and Schlosser, 2003. M. Kamvar, S.D. Schlosser. **The eigentrust algorithm for reputation management in p2p networks.** *WWW'03: Proceedings of the 12th international conference on World Wide Web*, ACM Press (2003), pp. 640-651
- Li et al., 1995. D. Li, H. Meng, X. Shi. **Membership clouds and membership clouds generator.** *Journal of Computer Research and Development*, 42 (8) (1995), pp. 32-41
- Lin and Varadharajan, 2005. Lin, C., Varadharajan, V., 2005. Trust enhanced security for mobile agents. In: *The Seventh IEEE International Conference on E-Commerce Technology CEC 2005*, pp. 231-238.
- Lin and Varadharajan, 2006. Lin, C., Varadharajan, V., 2006. Trust based risk management for distributed system security – a new approach. In: *The First International Conference on Availability, Reliability and Security, ARES 2006*, p. 8.
- Matthew and Agrawal, 2003. Matthew, R., Agrawal, R., 2003. Trust management for the semantic web. In: *Proceedings of the Second International Semantic Web Conference*, pp. 351-368.
- Quercia et al., 2006. Quercia, D., Hailes, S., Capra, L., 2006. TATA: towards anonymous trusted authentication. In: *Proceedings of the Fourth International Conference on Trust Management*, Pisa, Italy, pp. 313-323.
- Quercia et al., 2007. Quercia, D., Hailes, S., Capra, L., 2007. TRULLO-local trust bootstrapping for ubiquitous devices. In: *Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous*, pp. 1-9.
- Robinson et al., 2004. Robinson, P., Vogt, H., Wagealla, W., 2004. Some Research challenges in pervasive computing. In: *Post Workshop at the Second International Conference on Pervasive Computing*, Vienna, Austria, pp. 1-16.
- Rahman and Hailes, 1998. Rahman, A.A., Hailes, S., 1998. A distributed trust model. In: *Proceedings of the 1997 Workshop on New Security Paradigms*, pp. 48-60.
- Rahman and Hailes, 2000. A.A. Rahman, S. Hailes. **Supporting trust in virtual communities.** *Proceedings 33th Hawaii International Conference on System Sciences*, vol. 1, IEEE Press (2000), p. 9 (s)
- Stajano and Anderson, 2002. Stajano, F., Anderson, R., 2002. The resurrecting duckling: security issues for ubiquitous computing. In: *Computer*, vol. 35(4), Part Supplement, April 2002, pp. 22-26.
- Shand et al., 2003. Shand, B., Dimmock, N., Bacon, J., 2003. Trust for ubiquitous, transparent collaboration. In: *First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pp. 153-160.
- Sun and Song, 2004. Sun, H., Song, J., 2004. Strategyproof trust management in wireless ad hoc network. In: *Proceedings of the IEEE Canadian Conference on Computer and Electrical Engineering*, vol. 3, pp. 1593-1596.
- Sharmin et al., 2006a. Sharmin, M., Ahmed, S., Ahamed, S.I., 2006a. An adaptive lightweight trust reliant secure resource discovery for pervasive computing environments. In: *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2006)*, pp. 13-17.
- Sharmin et al., 2006b. Sharmin, M., Ahmed, S., Ahamed, S.I., 2006b. MARKS (middleware adaptability for resource discovery, knowledge usability and self-healing) for mobile devices of pervasive computing environments. In: *Third International Conference on Information Technology: New Generations (ITNG'06)*, pp. 306-313.
- Theodorakopoulos and Baras, 2004. J. Theodorakopoulos, G. Baras. **Trust evaluation in ad-hoc networks.** *Proceedings of the 2004 ACM Workshop on Wireless security*, ACM Press (2004), pp. 1-10
- Weiser, 1993. M. Weiser. **Some computer science issues in ubiquitous computing.** *Communications of the ACM*, 36 (7) (1993), pp. 75-84
- Wolfe et al., 2006. S.T. Wolfe, S.I. Ahamed, M. Zulkernine. **Trust Framework for Pervasive Computing Environments.** The 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06), *IEEE CS Press, Dubai, UAE* (2006), pp. 312-319
- Xiong and Liu, 2004. L. Xiong, L. Liu. **PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities.** *IEEE Transactions on Knowledge and Data Engineering*, 16 (7) (2004), pp. 843-857

- Xiu and Liu, 2005. Xiu, D., Liu, Z., 2005. A dynamic trust model for pervasive computing environments. In: *The Fourth Annual Security Conference*, Las Vegas, NV.
- Zhou and Haas, 1999. L. Zhou, Z.J. Haas. **Securing ad hoc networks**. *IEEE Network Magazine*, 13 (6) (1999)
- Zhou and Hwang, 2007a. R. Zhou, K. Hwang. **PowerTrust: a robust and scalable reputation system for trusted P2P computing**. *IEEE Transactions on Parallel and Distributed Systems*, 18 (4) (2007), pp. 460-473
- Zhou and Hwang, 2007b. Zhou, R., Hwang, K., 2007b. Gossip-based reputation aggregation in unstructured P2P networks. In: *IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1–10.
- Zultner, 1999. R.E. Zultner. **What do our metrics mean?** *Cutter IT Journal*, 12 (4) (1999), pp. 11-19