12-18-2008

A Trust-based Secure Service Discovery (TSSD) Model for Pervasive Computing

Sheikh Iqbal Ahamed
Marquette University, sheikh.ahamed@marquette.edu

Moushumi Sharmin
University of Illinois at Urbana-Champaign

Recommended Citation

Ahamed, Sheikh Iqbal and Sharmin, Moushumi, "A Trust-based Secure Service Discovery (TSSD) Model for Pervasive Computing" (2008). *Mathematics, Statistics and Computer Science Faculty Research and Publications*. 307.
https://epublications.marquette.edu/mscs_fac/307

Computer Communications, Vol. 31, No. 18 (December 18, 2008): 4281-4293. 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.

A Trust-based Secure Service Discovery (TSSD) Model for Pervasive Computing

Sheikh I. Ahamed
Marquette University, 1313 W Wisconsin Avenue, Milwaukee, WI
Moushumi Sharmin
University of Illinois, Urbana-Champaign, IL

Abstract

To cope with the challenges posed by device capacity and capability, and also the nature of ad hoc networks, a Service discovery model is needed that can resolve security and privacy issues with simple solutions. The use of complex algorithms and powerful fixed infrastructure is infeasible due to the volatile nature of pervasive environment and tiny pervasive devices. In this paper, we present a trust-based secure Service discovery model, TSSD (trust-based secure service discovery) for a truly pervasive environment. Our model is a hybrid one that allows both secure and non-secure discovery of services. This model allows Service discovery and sharing based on mutual trust. The security model handles the communication and service sharing security issues. TSSD also incorporates a trust mode for sharing Services with unknown devices.

Keywords

MARKS, Secure service discovery, Pervasive computing

1. Introduction

Pervasive computing [1], [2], [3] has evolved over the last few years due to recent developments in portable low-cost lightweight devices and the emergence of short range, and low-power wireless communication networks. Pervasive computing environments focus on integrating computing and communications with the surrounding physical environment of day to day lives for making computing and communication transparent to the users. In a broad sense, pervasive computing includes four major areas: mobile computing, wireless networks, embedded computing, and context-aware sensor networks [4]. The pervasive computing environment is comprised of numerous devices that include PDAs, cell phones, smart phones, laptops, sensors, etc. Nowadays, these devices are truly everywhere making Weiser's vision a reality [5].

In a pervasive computing environment, there are different kinds of networks. On one end, there are smart spaces, or intelligent environments that provide devices with complex computational support, while at the other end there are networks without any such support. In between there are networks that take some kind of support from fixed access points or central servers, but are not parts of a smart space.

Fig. 1 depicts two ad hoc networks in a pervasive computing environment. In Fig. 1(a), the tiny devices communicate among themselves with the support of fixed, powerful devices. These devices act as servers or proxies and handle complex computations on behalf of the tiny devices. In Fig. 1(b), an ad hoc network is formed by mobile devices. There is no fixed infrastructure support. The devices communicate with each other directly or via another mobile device, and are responsible for performing computations by themselves. Our focus is infrastructure-less mobile computing.

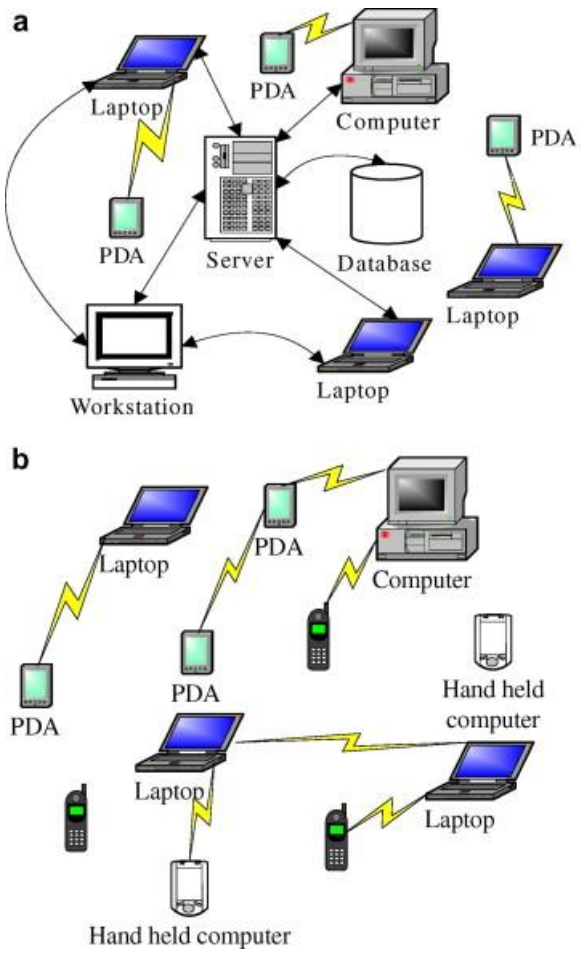


Fig. 1. Different types of networks in pervasive computing environment. (a) Ad hoc network in pervasive environment with powerful device support. (b) Ad hoc network in pervasive environment without powerful device support.

Service discovery is an integral part of every system running in a pervasive computing environment [6]. This process explores a device capable of offering a specific Service optimally. Despite the exponential growth of the exploitation of handheld devices (e.g. PDAs, laptops, smart phones, etc.), these devices themselves are suffering from a number of limitations [7], [8], which include but is not limited to, inadequate processing capability, restricted battery life, limited memory space, slwepensive connections, frequent line disconnection, and confined host bandwidth. Lack of fixed infrastructure support is a natural phenomenon in this environment, which leads to the dependency on other devices for services. These devices interact with other devices in an ad hoc manner. The nature of devices, communication pattern, and dependency on others in turn causes security threats. Also, due to the ad hoc and ephemeral nature of the network, one can't expect to get service from a particular device for a long span of time. Moreover, multiple devices may concurrently request one specific service. These aspects demand a scalable, efficient, quickly responsive, and dependable service discovery model.

The significance of security during service discovery in pervasive computing environments is an established truth [9], [10], [11]. Privacy, security, and trust issues in service discovery in pervasive computing area are of utmost importance [4]. Many users may be happy to share the services of their handheld devices, provided this sharing will not cause any security threat to them. Thus, the service discovery process demands models that ensure the privacy and security of the user. At the same time, it is also important to keep in mind that most of the devices operating in this environment are service poor. These devices do not support large-scale cryptographic protocols. The traditional security mechanism does not work in this environment, because the devices are computationally poor and the notion of physical security is not applicable [12]. Adding complex protocols will lower the performance of these devices so much that many embedded and mobile systems tend to ignore privacy, security, and trust issues. These factors force us to think about what type of model we should use. Is it logical to sacrifice total security? Or, is it absolutely necessary to impose a full-fledged security structure on these devices similar to desktops and laptops? Is there any middle ground, that will protect these devices without having a heavyweight security mechanism? Ensuring varying levels of security for various services is another research challenge. Lack of availability of information about users is another primary concern in designing a discovery model, which necessitates the introduction of risk assessment [13].

Existing service discovery models can be divided into three broad categories. First are the service discovery models that do not address security issues [14], [15], [16], [17], [18]. Second, there are models that consider a full-fledged security mechanism with the help of some fixed infrastructure support (powerful servers, proxies, etc.) [19], [20], [21]. There are also models that support security with the assistance of additional hardware [22], mutual authentication [23], and trust [24], [25]. These models either completely trust or completely distrust one device. Each of these models has their own strength and weaknesses as they attempt to solve the problem using different approaches.

In this paper, we present a trust-based secure Service discovery model, TSSD (trust-based secure service discovery). Our model is designed for a truly pervasive environment, where we assume that the mobile devices would be able to handle necessary computations and communications by themselves, without any fixed infrastructure support. Our model is a hybrid, that allows both secure and non-secure discovery of services. This model allows Service discovery and sharing based on mutual trust. However, for unknown devices building trust relationships is complicated and sometimes impossible. To handle situations like this, we also include a risk model. The security model handles the communication and service sharing security issues.

1.1. Contribution of this paper

The contributions of this paper are as follows:

1.1.1. Service based security mechanism

Different services need different levels of security. Security is not device specific, it is rather service specific. All services in a device are categorized based on their security needs.

1.1.2. Context dependent mode of operation

Our model operates in both secure and non-secure mode, depending on the context. In trusted environments it works in a different manner than in a distrusted environment. Group communication is facilitated in our model as members of the same group can set high trust level values for one another and operate without calculating trust and risk parameters each time they communicate. At the same time they can communicate with devices outside the group with the full trust and security enabled mode.

1.1.3. Trust model

A modified PGP-based trust model is designed and implemented. This allows devices to build trust relationships and share services without compromising their security and privacy.

1.1.4. Risk model

A risk assessment model is included that permits unknown devices to receive services that they need. Because the service providers can estimate the potential risk of sharing a service, they can either accept or reject requests efficiently. As most of the devices interacting in a pervasive ad-hoc network have no or very few information about each other, our model works efficiently in this environment.

1.1.5. A simple and lightweight message passing

We have minimized the overhead of encrypting messages each time a device requests or provides services. We examined different security policies and tried to find one that possesses secure communication without overly burdening the system.

1.1.6. Classification of service discovery

This paper also presents a good survey of service discovery. It has also put the service discovery into four categories along with the state of art.

The outline of this paper is as follows: Motivations behind designing the system are presented in Section 2 which is followed by characteristics of our model in Section 3. Section 4 contains the current state of the art. An overview of our proposed approach is illustrated in Section 5. The details of the models have been described in Section 6 Trust model, 7 Risk and security model. The evaluation of our proposed model is presented in Section 8. Our future research direction and concluding remarks are described in Section 9.

2. Motivation

With the increased use of handheld and wearable devices, the pervasive computing area is stronger and more powerful than ever. Despite the physical constraint of devices running in such environment, most of the facilities that service-rich devices enjoy are incorporated in these devices. Devices depend on each other for Services because they do not have the luxury of containing all needed services due to physical constraints. Thus there is a need for a service sharing and discovery mechanism. Next, we consider some scenarios to show the importance and applicability of a simple service discovery model.

2.1. Scenario 1

To enjoy music, while on a bus ride, David needs any music playing software that is not available in his smart phone. David thinks that any of the passengers can have this software and decides to request all neighboring devices for the software.

2.2. Scenario 2

A group of marine biology students are collecting data from Lake Michigan. Their supervisor has provided a common formatted file for storing collected data. One of the students forgot to bring that file. He decides to request it from other students within the same group.

2.3. Scenario 3

Bob, a graduate student is going to attend a conference on ubiquitous applications. Due to heavy traffic he was late and when he reached the conference center, the presentation had already started. The presentation slides were interesting and the presenter was referring to some slides that he missed. Bob requests the slides from the presenter.

2.4. Scenario 4

Alice is visiting a botanical garden with her kids. It is 4:45 p.m. and suddenly she remembers that 5:00 p.m. is the deadline to pay her electricity bill. In her PDA she has her account numbers and all the necessary information, but her PDA has no Internet connection. She decides to send a request to nearby devices for a service that allows her to pay the bill.

The above scenarios present situations where service discovery and sharing are needed. But, these scenarios evoke the question of whether all services need the same level of security? Do all of these circumstances require the use of trust model or encrypted message communication?

The *first scenario* represents a situation in every day life. Depending on the preference of the device owners, the music playing software can be classified as a low or a moderate security service.

The *second scenario* presents a situation where each member has information about every other member. The access rights and trustworthiness is already known. The only question that arises is whether the requested service is available in the network or not?

In *the third scenario*, verification of Bob's identity is needed. If Bob is a registered conference attendee or there are other conference attendees who know Bob, then the presenter can provide the slides to him without much concern. Trust calculation is needed in this case.

The *fourth scenario* is a critical one, as the payment requires transfer of confidential information such as an account number and a debit/credit card number. Both Alice and the service provider need a secure communication mechanism. Also the presence of risk assessment is needed, as both the requester and provider have no prior information about one another.

A service discovery model can solve the issues arising from the above situations. We also notice that the needed security level is service-specific rather than device specific. This necessitates a service discovery model that can act in different ways depending on the security need of the related service. Most of the service discovery models proposed so far are not adaptive. Either they impose the same secure communication level for all services, which hampers the performance of tiny devices, or they totally ignore the security issues.

3. Characteristics

The required characteristics of a successful Service discovery model is summarized below:

Adaptive. The trust value and security level should change depending on the service itself, the service provider, and the service requester. The model should be able to work with and without security calculation, based on the situation.

Trust reliant. The model should consider trust relationships among devices. Where no prior information is available, mutual recommendation can be used as a tool for calculating trust. If no recommendation information is available, risk assessment can be used.

Infrastructure less. No infrastructure support (powerful servers, proxies, etc.) should be required. If the focus is on truly pervasive environments, then the model should work independently without any external support. This is important because in this environment infrastructure support is not always available.

Lightweight. The model should be lightweight in terms of executable file size.

Service oriented. As different services demand different levels of security, Service discovery models should be service oriented rather than device specific.

Non-degradable performance. The model should not put much overhead on the performance of the device.

Energy efficient. Service discovery models should be energy efficient. It should not require much battery power for computation or communication purposes.

4. Related work

Service discovery is a fundamental component of systems running in a pervasive computing environment. There are many schemes that attempt to solve this problem from various standpoints. Some products or protocols address the security and privacy issues of service discovery while others concentrate on providing the whole list of Services in a timely manner. A detailed survey on service discovery models and their classification can be found in [31]. In this paper, we categorize Service discovery models based on security and privacy features:

A. Service discovery models without security.
B. Secure service discovery models.
    1. Infrastructure-based.
    2. Infrastructure-less.
    3. Smart environment based.
    4. Hardware oriented.

Section 4.1 describes some service discovery models that do not address security issues and Section 4.2 discusses some models that offer a secure service discovery mechanism.

4.1. Service discovery models without security

The models that do not address security issues are Bluetooth, DEAPspace, International Naming System, etc. These models are some of the initial Service discovery models, which concentrate mainly on efficient Service lookup.

Bluetooth [32], [33] adopts a query-based approach. Device information, services, and the characteristics of the services are queried and connections between two or more bluetooth devices are established. This facilitates user selection, scope-awareness, and both unicast and broadcast communication. It also returns all matched Service information. However, Bluetooth does not provide effective security in Service sharing issue and fault-tolerant features.

DEAPspace [14] follows announcements for its Service discovery mechanism. Like Bluetooth, it also allows user selection. However, it doesn't take context information into consideration during Service discovery. Here, all the devices in an ad hoc network periodically broadcast the list of all known services (their own services and the services of other devices that they know). When a new device joins the network, it can get a picture of the existing services from these messages. As every device knows about the available Services, they generate a Service request query whenever necessary. The algorithm used for DEAPspace service discovery is very simple. However, this approach is not desirable because it uses bandwidth and also increases power consumption. Also, security and privacy issues are ignored.

International Naming System (INS) [16] supports both the query and announcement approach. It also supports unicast, anycast, and broadcast methods. It offers the best-matchService information and also provides facilities for limited support of context information. It assumes that every device will request a central name resolver for the type of services it requires, and the resolver will reply with the best matched device address. In a truly pervasive environment, the presence of a central resolver is not common. Even if a resolver is present, which stores all the Service information centrally, the entire system is vulnerable to single-point-failure.

4.2. Secure service discovery models

Most of the service discovery models designed now-a-days fall into this category. There are some models that include full fledged security mechanisms, while others rely on simple algorithms for limited security. This category can be sub divided into infrastructure based, infrastructure less, hardware based, and smart space oriented security mechanisms. In the following subsections we discuss each of these categories.

4.2.1. Infrastructure-based security

Secure service discovery service (SSDS) [19] concentrates mostly on security issues for communication purposes. It not only supports unicast, multicast, and broadcast communication, but it also provides facility for scope-awareness. Both query and announcement systems are supported. It allows capability based access control where pre-specified agents are allowed to discover services. SSDS follows the client-service-server model. To make the discovery process secure, all information passed between clients and servers, and servers and services are encrypted. This puts a lot of overhead on the mobile devices. In this approach, a single copy of the Service information is stored and accessed, which makes the system vulnerable to single point failure.

Jini [34] is based on Java and its main focus is on the enterprise level. Jini's architecture creates a federation to offer and share services. Devices join the federation to get required services. This requires prior knowledge about devices. They also do not consider separate security features because, devices have assigned access privileges.

Kazuhiro Minami et al. proposed a secure context-sensitive authorization scheme (CSAS) [35]. This is a proof-based decentralized system. This system distributes the proofs in different devices and eliminates the need of a central server. To authorize access, it uses previously stored information about the device user and checks its validity. This type of information is often impossible to collect for users in an ad hoc network.

Splendor [20] is a secure, private, and location-aware service discovery protocol. This approach considers environmental variables. Depending on the environment, a client-service model or client-service-directory model is used. Proxies are used to offload computational work on mobile services. Mobile services authenticate with proxies and proxies handle registration. In these situations, proxies are considered to be trusted servers. However, if no trusted server is available in an environment, who will handle this registration process? This model depends heavily on proxies that, compared to mobile devices, are considered powerful machines. But, what will happen if the particular mobile environment is lacking any such powerful device or that device does not want to communicate with other devices? What does happen? Splendor emphasizes security and privacy issues in Service discovery. It also supports location-awareness, and uses a combination of public key and symmetric key for security purposes. A client-service-directory-proxy based model is used for service discovery, which can be burdensome for devices like PDAs and smart phones. It also does not provide efficient support for directory faults. Its security model is based on mutual authentication. In our approach we use a combination of mutual authentication and a trust model that follows from this approach. This trust model will help devices in the authentication and communication processes.

Progressive exposure [21], [36] is a secure service discovery approach. It addresses privacy issues using a mutual matching technique. Progressive exposure addresses security and fairness by not exposing too much information. In each round of message exchange between communicating parties, it tries to find whether any mismatch occurs. In case of a mismatch, the communication stops. It uses one-time code words and a hash-based message authentication code. It considers the presence of one user and one service provider, but it does not address situations in which many users and many service providers are present. Nor does it address what happens when a service provider leaves the ad hoc network, which is a very common scenario in a pervasive computing environment. When the service provider leaves the network, the user has to start from the beginning with the provider lookup and authentication phase. How the user knows about who is providing what services and keeps track of these are not addressed. From a security perspective this model presents a symmetric encryption key based approach that can be computationally costly if the service provider leaves the network soon after the communication is established. It provides privacy for service information, requests, domain identity, user credentials etc., and is based on client-service-directory model. Like Splendor it does not provide facility for fault tolerance.

4.2.2. Infrastructure less security

Patrick Brezillion et al. present a context-based security model (CSM) in [37]. They discuss the need for adaptive security based on the particular situation. In our approach we also used the concept of adaptive security but the mode of security depends not only on the surrounding environment (date, time, etc.) but also the preference of the user. Our policy file chooses the appropriate security model based on user preference.

Roshan K. Thomas et al. [38] present the challenges and research issues for secure pervasive computing. They express the need for a dynamic trust model as the pervasive computing environment poses new kinds of security challenges due to its diverse nature. They present a socio-technical view. We completely agree with their idea that to address security problems in this environment, we have to think of a model that provides adequate security yet also considers the service limitations of an ad hoc network and the devices operating in it.

SPDP [24] is a secure service discovery protocol based on PTM [39], [40] model. This model addresses the ad hoc networking issues of mobility and Service constraints. The need for a centralized server is abolished here by means of a user agent and service agent. Each device acts as its own certificate authority. For a service request, this model uses broadcast messaging. The requesting device updates its cache after getting a reply from the devices (if any reply). It then stores the device identities that it believes trustworthy. The devices' user agents continually listen for messages, which in turn means continual energy consumption. Also, the service request reply message requires fields such as how long the device will stay in the network; these are the kind of messages that some users may not wish to share. Another drawback is there are no categories for services according to security needs, yet they do define trust and its properties. This model uses a simple approach for service discovery and our model is inspired by their model. In our approach we follow a trust model that has properties derived from this model.

Narendar Sarkar et al. in [41] proposes an attribute vector calculus (AVCM) based approach for modeling trust. Their model describes both identity based trust and context based trust, and is one of the first models that discusses the importance of trust in a ubiquitous environment. Their model is a very simple one and does not address complex situations where assigning trust is not sufficient for ensuring security.

4.2.3. Smart space dependent security

Universal plug and play (UPnP) [15] defines protocols for communicating between UPnP control points and devices. The architecture is based on six phases of interaction. Control points handle discovery, description, and control phases, while addressing, eventing, and presentation are done by devices. It uses protocols like DHCP (dynamic host configuration protocol), ARP (address resolution protocol), and DNS (domain name service) for addressing purposes. Multicast messaging is used for device discovery. To make the model secure, it uses firewalls, medium access control layer encryption, and physical access control. This model is particularly suited for home, office or similar environments. It requires information about devices that will form the network, which is impossible to get in a pervasive network.

Anand Tripathi et al. in their context-based secure service access (CSRA) [42] paper present policies for service access. They concentrate on creating an environmental model that will provide transparent and secure support for services that a user needs. The main difference between our approach and theirs is that they require infrastructure for providing such services while our goal is to make the entire discovery process free from any infrastructure. To ensure secure access, they combine the users' specified policy, Service/service manager specified policy, and collaborative activity specified policy. Our approach is different in the sense that the entire security enforcement is dependent on the user and the service manager to cope with the limited computing power and space of tiny mobile devices.

Basu et al. presents a trust-based architecture (TRAC) for increasing security and user confidence in pervasive computing systems [43]. They use trust and role based access control for ensuring security and privacy. However their model is aimed at an intelligent environment (IE) only. This policy-based model allows users to define policies for themselves and thus gives users control to define their own security level. This model works in an IE because each user is known beforehand However, in a truly pervasive environment it is not possible to have prior information about every user and thus, this model is not applicable.

4.2.4. Hardware supported security

Heiko Kopp et al. provide a security architecture for service-based mobile environments (SME) [22]. Their central idea is to provide security with the help of hardware support. They consider different mobility levels that a device may go through and tries to provide hardware support for maintaining security. Though we also consider different levels of security, based on context and user preference, our main focus is providing a lightweight software solution.

In [44] Siani Pearson proposes a hardware-centered approach (HCA) for securing privacy. This mechanism works with the software system to maintain the owners privacy. The main drawback of this system is that it will not work on existing mobile and handheld devices.

A comparison table of the secure service discovery models discussed above is presented in Table 1.

Table 1. Comparison of secure service discovery models

Model	**Adaptive**	**Infrastructure support needed**	**Lightweight**	**Service oriented**	**Trust aware**	**Privacy aware**	**Context aware**	**Smart space needed**
SSDS [19]	No	Yes	No	No	N/A	N/A	N/A	No
UPnP [15]	No	N/A	No	No	Yes	Yes	Yes	Limited
SPDP [24]	No	No	Yes	No	Yes	N/A	No	No
Progressive exposure [21], [36]	No	Yes	No	No	No	Yes	Limited	No
Splendor [20]	No	Yes	No	No	Yes	Yes	N/A	No
Jini [34]	No	N/A	No	No	N/A	Yes	N/A	Limited
CSAS [35]	No	No	Yes	No	N/A	N/A	Yes	No
CSM [37]	Yes	No	Yes	No	N/A	N/A	Yes	No
AVCM [41]	Limited	No	Yes	No	Yes	Yes	Yes	No
CSRA [42]	No	Yes	No	No	N/A	N/A	Yes	Yes
TRAC [43]	No	N/A	No	No	Yes	Yes	N/A	Yes
SME [22]	Yes	N/A	N/A	Yes	N/A	Yes	No	N/A
HCA [44]	No	N/A	Yes	No	No	Yes	No	N/A
TSSD	Yes	No	Yes	Yes	Yes	Yes	Limited	No

5. Overview

To address the challenges presented in the motivation section and to attain the characteristics described, we propose a secure Service discovery model, TSSD. The TSSD model contains a discovery model and a trust, risk, and security management unit. The TSSD model consists of SAFE-RD (secure, adaptive, fault tolerant, and efficient service discovery) and SSRD (simple and secure service discovery) sub units. The SAFE-RD contains the device discovery unit and the Service discovery agent while the SSRD unit contains the trust, risk, and security unit. We describe the details of the SAFE-RD model in Section 5.1. The description of the SSRD unit is presented in Section 5.2. The architectural detail of TSSD is described in Section 5.3.

5.1. SAFE-RD model

SAFE-RD is the discovery unit of TSSD. After getting requests from devices through an application object, the discovery agent will pass it to the SSRD unit for processing. Subsequently, the processed request will reach the requesting device in the reverse direction. The SAFE-RD unit is responsible for cluster formation, Service lookup, and Service matching.

Devices residing in close proximity form clusters. Each cluster is considered a *Tree,* where devices are the nodes of the tree. Nearby clusters form a *Forest*. The Service manager of a device is responsible for Service lookup. To search a Service, a device, investigates its own cluster through Service manager, which is treated as the root of the tree. Fig. 2 represents one such tree structure, corresponding to any cluster where RM denotes the Service manager. It also reflects its "*Proximity Awareness"* feature, which means the RM prefers local Services first. Unavailability of such a Service will lead to exploration of the other clusters of the forest.

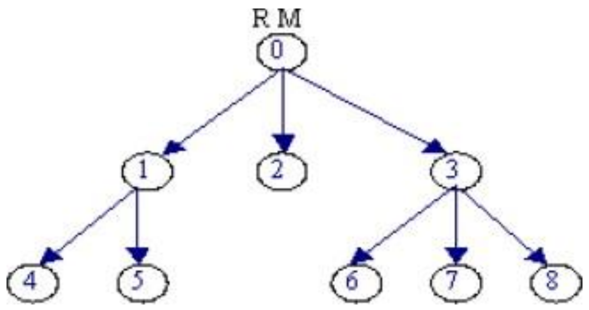


Fig. 2. Tree data structure.

The service manager, for service lookup, will use a match all technique. This allows a device to know all the potential Service providers. However, for service selection, the SAFE-RD communicates with the SSRD unit. The best provider is selected in terms of availability and trustworthiness. To maintain both QoS and scalability, SAFE-RD also stores information about the next best match.

5.2. SSRD model

The SSRD unit handles security related issues and consists of *trust management, risk assessment,* and *security management* sub units. The SSRD unit is directly linked to the Service discovery agent. The functionalities of all these units are maintained and controlled by the Service manager. All these units provide users with privacy and security without explicit user interaction. The model requires initial user input to set security levels for different services provided by the device. After this point, it needs user permission only in case of a highly secure service sharing time. This is necessary to maintain users' privacy. In this paper, we describe the features of our trust model, our risk assessment model, and our security model.

The *trust management unit* is responsible for maintaining the trust relationship with other devices. This unit calculates trust values for all devices and also updates the trust values depending on the behavior of the service provider or requester. It maintains a list of service-specific and average trust values and communicates with the risk assessment and security management unit whenever necessary.

For secure discovery and communication among devices, we are using a trust model influenced by PGP (Pretty Good Privacy) [45]. However, Direct PGP is not used since it requires huge computation. We are considering not only average trust but also service specific trust that will make the model more robust. For trust calculation, we are considering previous interaction information and also recommendation from other neighboring devices.

Lack of historical information is a natural phenomenon in a truly pervasive environment. Even if there is no information available, we still want to receive and provide services. To handle situations like this, we are proposing a risk assessment model that allows us to share Services without compromising the security of the device.

The *security management unit* selects the mode of communication (broadcast, multicast or unicast) depending on situation and security needs for a specific service. It also facilitates secure sharing of services. This mode is selected depending on the predefined functions and does not require explicit user intervention each time a service is requested. The security model also determines the mode of communication. It works with the trust management unit to request service and to provide service as securely as possible without compromising the performance of the device. Fig. 3 shows the conceptual diagram of the SSRD model.

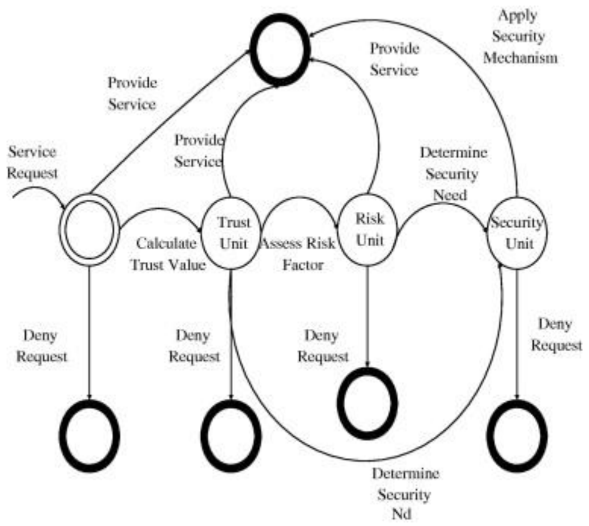


Fig. 3. Conceptual diagram of TSSD model.

5.3. TSSD architecture

TSSD is the Service discovery unit of MARKS (middleware adaptability for service discovery, knowledge usability, and self-healing) [26]. MARKS supports the core and supplementary services like Knowledge Usability [27], device discovery, Self-healing [28], PerAd service [29], etc. Fig. 4 presents MARKS architecture with TSSD unit.

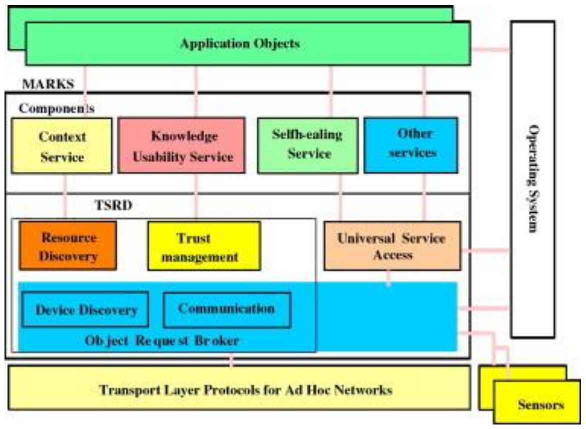


Fig. 4. MARKS architecture.

The architecture of the TSSD unit is presented in Fig. 5.

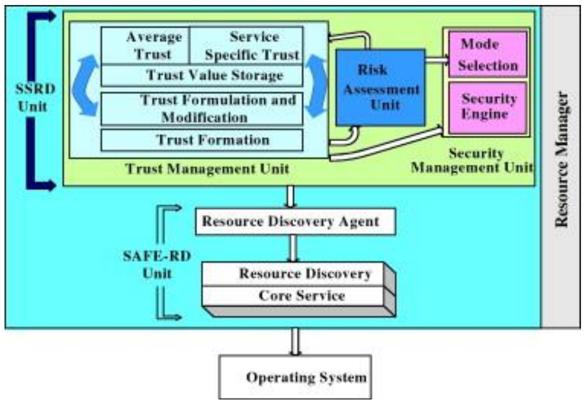


Fig. 5. Service discovery unit.

The TSSD unit consists of SAFE-RD and SSRD unit. The SAFE-RD unit consists of the device discovery unit, service discovery agent, and the SSRD unit contains trust management, risk assessment, and security management sub units.

The details of the SAFE-RD model and the MARKS architecture have been published in [46 and 26], respectively. Some preliminary results of SSRD are in [46], [57]. We have discussed only trust-based service discovery in [46] and risk based trust is introduced in [57]. In this paper, we present the details of trust, risk and privacy issues of service discovery for pervasive computing environment.

In this section, we have presented the overview of our TSSD model, Sections 6 Trust model, 7 Risk and security model contain the details of the trust, risk, and security model included in TSSD.

6. Trust model

Trust model plays important role in trust-based Service sharing decisions along with managing the trust relationships with other neighbors. We chose a set of services for testing the applicability of our model. The sample services include date and time service, music software, unzipped software, language translation software, etc.

6.1. Trust model properties

By trust we mean the level of confidence that a device has in another device. In this model, we are using both average trust level and service specific trust levels. Average trust levels are used when a device does not have any prior trust relationship with a particular device. The terms trust level and trust value denote the same thing in our paper.

The highest value of trust is 1.0 (complete trust) in our model, while the lowest is 0.0 (complete distrust). For each service, each new device is initially assigned a trust value of 0.5. We chose 0.5 because we believe that if no information about a device is available there is no reason to trust it or distrust it. This initial assignment does not threaten other devices as this value changes with the devices behavior and context. A device reserves the right to trust or distrust any device regardless of its trust level. $\tau(A,\ B, 1)$ means device A trusts device B completely for all services while $\tau(A_s,\ B,\ z)$ defines the trust level from A to B for service s (A owns this service and B is requesting it), which depends on the value of z $0.0 \leftarrow z \leftarrow 1.0$. $o(s,\ A)$ means A is the owner of service/device s.

1. ***Reflexive:*** Each device trusts itself completely.
   $\forall A \tau(A,A,1)$.
2. ***Mutual trust:*** If a person X has multiple devices $(s1,\ s2,\ s3,\ \ldots\ ,\ sn)$, all those devices have complete trust on each other.
   $\forall s1,s2(\mathrm{o}(s1,X) \wedge \mathrm{o}(s2,X)) \Rightarrow \tau(s1,s2,1)$.
3. ***Partially transitive:*** If A trusts B and B trusts C, then A trusts C. But, the trust level between A and C may differ.
   $\tau(A,B,1) \wedge \tau(B,C,1) \Rightarrow \tau(A,C,x)$,
   $\tau(A,B,x) \wedge \tau(B,C,y) \Rightarrow \tau(A,C,z)$,
   $z = \psi(x,y)$,
   $0.0 \leftarrow x,y,z \leftarrow 1.0$.

The first equation depicts the complete trust relationship between device A and B and device B and C. Based on the rule of transitivity, A trusts C but with a value x. This is in accordance with the fact that if A trusts B completely and B trusts C completely then it does not mean that A has to trust C completely. The second equation depicts a scenario where A trusts B with a value x and B trusts C with a value y. Using this trust relationship, we can calculate a trust value z between A and C. Now the value of z is the minimum of $(0.5,\ x\ *\ y)$. This value of z is chosen this way as it simplifies the trust value computation and in situations where trust relationship is building for the first time, it is better not to assign a higher trust level value.

4. ***Service dependent:*** For a particular service, A may trust B completely, while for another service it may not. The level of trust may vary depending on the associated security level of a particular service.
5. ***Context awareness:*** The level of trust depends on the context. It is a value at an instant of time. Examples of context are: time, location, and communication load of network, etc.

6.2. Hybrid model

It is obvious that not all services residing in a device need the same level of security. A device owner may offer weather service to anyone, but may offer Internet services to a limited group of people. This means there must be a security level value with each service. This allows the service manager to decide which services can be shared with whom, without explicit user input.

To reduce the computation load of our target devices, we categorize services based on their need for security. The service manager of each device is responsible for maintaining a list of services that the device owns and wishes to share. This list (Shown in Table 2) contains not only the service name but also a numerical value that denotes the security level of the associated service.

Table 2. Service list with associated security level

Service name	Security level (1–10)	User permission (yes/no)
Date/Time	1	No
Weather	1	No
WordPad	2	No
Chat software	4	No
Unzip SW	7	No
Language translation service	8	No
Internet	9	Yes
Address book	10	Yes

Initially the user assigns these security levels. Values from 0 to 4 indicate that these services do not require secure communication with service requester and can be shared with almost any device even devices with a lesser trust value. Values from 5 to 7 means these services require secure communication and can be shared with devices having trust value 0.5 or higher. But services with security level 8 and higher means that these are highly sensitive services and can be shared only with highly trusted devices having a trust value $\geqslant 0.8$. We have assigned the security level values by considering the number of services that require these different levels of security. This assignment is not rigid and can be changed by the user without any complex procedure. If a requested service has a security level greater than 8, the user will be asked whether he wants to share it or not.

After getting service request, the Service manager consults the service list table first. If the security level is below 5, it shares the service with the requester (having trust value > 0.2). If the security level is between 5 and 7, it consults the trust value and depending on that value decides whether to provide the service or not. For services having security level > 7, it consults trust model and depending on trust value pre-approves or rejects the request. In case of pre-approval, it prompts to the user whether he wants to provide the service or not. This hybrid model is shown using algorithm PermitService.

For services with security level < 5, no trust calculation or secure communication is needed. But for services with higher security levels, at first trust is calculated and then secure communication is established between the provider and the requester. This lessens both the computation cost and the communication overhead. This in turn saves battery power and adheres to service providers' privacy.

If service request comes from a device for which there is no prior information and the recommendation data are also unavailable, then the trust management unit communicates to the risk assessment unit for risk calculation. Based on the feedback of the risk assessment unit, the trust unit calculates a trust value.

Procedure PermitService (s_R^P)

```
$P$, $R$ = Service Provider & Requester
$s_R^P$ = Service Requested by $R$ to $P$
$s^{SL}$ = Security level of $s_R^P$, $1 \leftarrow sSL \leftarrow 10$
$t^s$ = Trust Value of $s_R^P$ from $P$ to $R$, $0.0 \leftarrow ts \leftarrow 1.0$
$T_R^P$ = Trust Value from $P$ to $R$ for all services
$S^P$ = All services available at $P$
L1. if not ($s_R^P \in S^P$) then return "No such service"
L2. if not ($t^s \in T_R^P$) then Determine $t^s$ by (1)
L3. if $t^s > 0.2$ then
L4.  if $s^{SL} \leftarrow 4$ then provide $s_R^P$ to $R$; return
L5.  else return "not permitted"
```

```
L6. if $t^s \geqslant 0.5$ then
L7.  if $s^{SL} \leftarrow 7$ then provide $s_R^P$ to $R$; return
L8.  else return "not permitted"
L9. if $t^s \geqslant 0.8$ then
L10.  take input from the user and act accordingly
end procedure
```

6.3. Trust model

The motivation behind having a trust model is that we do not want to provide services to malicious devices but at the same time we also do not want to decline requests from legitimate devices. Our trust model is designed to associate each device with a trust value based on past behavior with the requesting device. Also when we calculate a trust value for an unknown device, we consider a PGP [45] based trust model. PGP is based on mutual certification of the validity of the keys. In this model, a user can obtain a key by building chain of trust starting from people they know. In this model "Global Trust Register" acts as a complementary introducer. In our model we took this concept and modified it to fit the ad hoc nature of the network and limited processing power of the devices.

In our model, the service manager of each device contains a list of nearby devices, which discloses their presence along with an associated trust value. Table 3 contains this list.

Table 3. Service-trust list

	Service 1	**Service 2**	**…**	**Service *n***
Device 1	0.6	0.8	…	0.75
Device 2		0.4	…	…
⋮	⋮	⋮	⋮	⋮
Device *m*	0.3	0.3	…	0.9

6.3.1. Trust value calculation

Eq. (1) is used for the calculation of trust value for devices that interacted with this device earlier. As the service manager of this device has information about the behavior of these devices, whenever a service request comes from any of these devices, it checks in the service-trust list whether there is any corresponding value for that particular service or not. It calculates the average trust value using (1) and assigns this value in the corresponding table position. Depending on this newly calculated value, it decides whether to accept the request or not.

(1)

$$\tau(SP,A) = \left(\sum_{i=1}^{n} Si * \tau(SPi,A,x)\right) / \sum_{i=1}^{n} Si\,.$$

Here, *SP* is the service provider, $\tau(SP, A)$ the average trust value of device *A* for device *SP*, S_i the security level of *i*th service, $\tau(SP_i, A, x)$ the trust value of *A* for *i*th service and *n* is the number of services that links *SP* and device *A*.

Eq. (2) is used by the service manager to calculate the trust value for any new device. If the new device requests a service, the service providing device generates a multicast message to all devices that it has involvement and asks for their recommendation about this device.

(2)

$$\tau(SP,D_{\text{new}}) = \left(\sum_{i=1}^{n} \tau(SP,i) \times \tau(i,D_{\text{new}})\right)/n$$

Here, *SP* is the service provider, D_{new} the new device requesting service, $\tau(SP, D_{\text{new}})$ the average trust value of D_{new} for device *SP*, $\tau(i, D_{\text{new}})$ the average trust value of device *i* for D_{new} and *n* is the number of devices that are link with both *SP* and D_{new}.

The above model allows a device to get services from a provider never before interacted with, and build a trust relationship gradually. If the service provider fails to get any recommendations from the devices, it communicates with the risk assessment unit for risk analysis. Based on the feedback from the risk unit if any device of the network does not have any information about the new device, then 0.5 is assigned as initial trust value. This value is chosen carefully to allow this device to get services that do not require high security and build a trust relationship with other devices. Another feature of our trust model is that the trust value is service specific. It is dynamic and changes depending on the behavior of the device. Devices with lower average trust value can request and get services that do not threaten the security of the service provider while at the same time are restricted from getting highly secure services.

6.3.2. Trust value modification

A human-like trust relationship is desirable in a pervasive computing environment, but it is impossible to impose robust security among devices in this environment. In human societies, trust relationships are built on mutually agreed upon behavior and recommendations. Unknown persons get an opportunity to become familiar with the recommendations of others and can enhance their trust level by displaying good behavior. Misbehavior also plays a role in determining trustworthiness of a person. We wanted to design a model that is similar to the human trust relationship model. However, the difficulty in doing so is determining what behaviors are rewarded and what are not acceptable. It is difficult to judge. In a smart space, when the users are known and have access rights associated with Services, we can calculate the degree of positive or negative behavior by observing whether or not they are trying to access non-permitted devices, or whether they are dominating the use of a Service and causing problems for others. However, in a truly pervasive environment, where the entities do not have any associated privilege level or are totally unknown, finding criteria that increases or decreases the level of trust of a device is extremely difficult. For simplicity we divided behaviors into three broad categories – good or rewarding behavior (this increases the average trust value), neutral behavior (which does not have any effect on the average trust value), and negative behavior (decreases the trust value). To evaluate these behaviors numerically, we have stored the average service time *(this includes request time + service offer time)*in the risk table. Whenever a device requests a service and is offered that service, the total time required to complete the service request is recorded and compared with an average time. Here, τ is used to update the trust values between devices. This value is introduced to mimic the trust building relationship in human networks in a simple manner. The average trust level value is updated using (3), (4), (5).

(3)

$$\theta i = (\sigma_a - \sigma_r)/\sigma_a,$$

(4)

$$\tau(SP,D) = \tau(SP,D) + \theta i,$$

(5)

$$\tau = \frac{\sum_{i=1}^{n} \theta i * \omega i}{n} \pm c.$$

Here, θ_i is the modification value for service *i*, τ(*SP*, *D*) the trust value of device *D* for service SP, τ(*D*) the average trust value of device *D*, ω_i the 0.1* security level of service *i*, *n* the number of services relating to provider and requester, σ_r the required time for a successful request completion, σ_a the average service time and *c* is the random behavioral parameter.

In (5), c is included to reflect behaviors such as sending too many requests in a small amount of time, repeatedly sending a request that has already been rejected, and so on. *c* is generated from a uniform random number generator [58] using parameters like number of requests (μ), number of accepts (α), number of rejects (β), number of same request (ω), etc. If $c > 0.5$, then we add in (5), otherwise we deduct *c*. The *c* value is calculated using (6).

(6)

$$c = (\alpha/(\mu + \beta)) - -(\omega/\mu).$$

Our trust model is privacy aware because the device asks the user in case of responding to a higher security level service request. Even if the trust or risk model indicates that it is safe to share a service; the user can deny any request. The model is designed in a way that it does not prompt the user for permission for every service sharing. However, when the security level of a particular service is above a threshold value, it asks for user permission. Thus, it maintains transparency of operation and protects users' privacy. Some preliminary results of this model are available in [47].

7. Risk and security model

Risk assessment and security enforcement are important steps of TSSD. The risk assessment unit and security management unit are part of the SSRD unit. Section 7.1 discusses the risk model, Section 7.2 contains the security model, and the message-processing model that we use in TSSD is described in Section 7.3.

7.1. Risk model

A risk model is essential during the sharing services in a pervasive environment. Risk evaluation becomes significant when a service request comes from an unknown device or when there is not enough recommendation information. When a service request arrives, we calculate the trust value of the requesting device (if the providing device has information about the requester or by collecting recommendation from other devices). Then based on the security level of the requested service, we accept or deny the request. When the requester is unknown to all the neighboring devices (a very common scenario in pervasive computing), the device is assigned an initial trust value of 0.5 which would allow it to receive lower security-intensive services and build a trust relationship with others. However, if that device requires a higher security level service, it is denied. To address this issue, we have added the risk assessment along with our trust and security model.

The risk model that we are currently using is a lightweight one. Each device has a risk evaluator. This evaluator stores information about high security services and calculates the risk value when a request comes for one of these services. Each time a service request arrives along with an accepted or rejected event, it updates the risk value associated with that service. It collects information about the service that includes number of accepts (γ), total number of requests (ϕ), average trust values of the devices who request this service, service time (σ), etc.

To calculate, the risk factor (7) is used:

(7)

$$\rho = \frac{1}{(\gamma/\phi)} \times \tau.$$

Here, ρ is the risk factor, γ the number of accepts, ϕ the number of request and τ is the average trust value for this service.

The range of the risk factor, ρ is $0 \leftarrow \rho \leftarrow 1.0$. This is a weighted average with respect to average trust value. A value of 0.5 indicates around 50% acceptance rate for this particular service. If the risk factor value is high (> 0.5), then the request is rejected. In the case of a low risk factor, the service is provided. Based on this value, the device assigns a risk factor with the service. As this information is collected every time a service is requested or shared, a historical database is created for services of a particular device. Each device has its own database that allows it to decide the risk factor for its services. This allows a device to decide whether to accept a request or not when there is little or no information available about a requester. Table 4 shows some sample data stored in a device.

Table 4. Risk value table

Id	**Number of request (φ)**	**Number of accept (γ)**	**Average trust value (τ)**	**Average service time (σ) in ms**
5	3	1	0.75	21
9	7	6	0.6	15
13	17	13	0.83	40

Each time a service request is made, the risk value table is updated to include the modified number of requests, number of accepts, average trust value of devices for which the request is accepted, and average service time to offer. The updated data is used to calculate the risk factor for sharing a service with unknown devices. We are currently using statistical distributions to find out optimal percentage rate and trust value pair that lowers the risk of service sharing. The average service time is compared with the service-sharing time to evaluate the behavior of the requesting device. This value is used for dynamic modification of trust value. In Section 6.3 the details of trust value modification is presented.

7.2. Security model

There are four major aspects of security-confidentiality, integrity, availability, and authentication. In our model, we focus on these issues from a pervasive environment viewpoint. Each of these security issues has been widely studied and many standard solutions exist for each (e.g. confidentiality [49], [50], integrity [50], [51], [52], availability [53] and authentication [45], [54], [55]). Each of these security solutions has pros and cons, but the most crucial thing is that there is no solution designed with the limited computational power of mobile devices in mind. The ad hoc nature of the network also restricts us from using these systems in their original form. Therefore, we have proposed a trust model, influenced by PGP (pretty good privacy), for authentication purposes and to handle security issues. The MWT (maximum waiting time) concept [48] is used to attain availability. To maintain the privacy rights of the users, the device owners reserve the right to disclose availability time to others. Users can choose either the message formats that include availability time or do not include availability time. Depending on the environment of the user, any of the messages can be used. For integrity and confidentiality, we are using a signature based [45] hybrid model composed of both public key and symmetric key.

In Section 6, we have described our trust model that handles authentication issues. We are using a digital signature based security mechanism for integrity purposes, instead of MAC [51] or HMAC [52]. Once the trust relationship is established, for those services that do not require a high level of security, the provider offers the service without using any public key or symmetric key operations. This saves not only computation time but also battery power, which is a major concern for these tiny devices [9]. But for critical services (services that require high level of security), we are using a model composed of digital signatures and public key operations. In reality, a small number of services require this level of security. So the added computation for using a public key is not a burden.

7.3. Message processing model

For requesting a service, devices use one of the following message formats:

Service_Request (device_id, service_name)
Service_Request (device_id, service_name, MWT)
Service_Request (device_id, service_name, MWT, Requesters_Signed_Key)

Here, MWT is the maximum time a requester can wait to share/receive a service. MWT is added to cope with the ad hoc nature of devices operating in a pervasive computing environment. If the device owner chooses not to disclose the availability information, the first format is used. The second message is sent, if the requested service has a low security level. Otherwise, the third message is sent. Depending on the device, the security level of the service, and the relationship with other devices in the ad hoc network, this request is sent by using a broadcast, multicast or unicast method.

1. If the device is new or no information is available about where this service can be found, and the security level of the specific service is not high, it will broadcast the Service_Request message.
2. If no information is available about where this service can be found, and the security level of the specific service is high, it will multicast the Service_Request message to devices that it already communicated with and has an established trust relationship.
3. If the device has prior information about the service (from where it can be shared), it will send a multicast or unicast message depending on the security level associated with the service.

The service provider sends any of the above two messages depending on the security level of the requested service. The provider always uses unicast for this reply.

Service_Reply (device_id, service_name)
Service_Reply (device_id, service_name, Providers_Signed_Key)

Before sending a Service_Reply message with the signed_key, the provider sends Request_for_Trust_Value messages to devices that have a trust relationship with it. This message is sent using a multicast method.

Request_for_Trust_Value (requesting_device_id, providers_id)
Reply_for_Trust_Value (repliers_id, trust value, Repliers_signed_key)

The Reply_for_Trust_Value message is always sent as a unicast message. Because most of the service requests do not have a high level of associated security, there is little overhead for sending and verifying a signed key. This is also critical for services with high associated security levels.

The result of the TSSD model [56] is presented in the 30th Annual International Computer Software and Applications Conference (COMPSAC 2006).

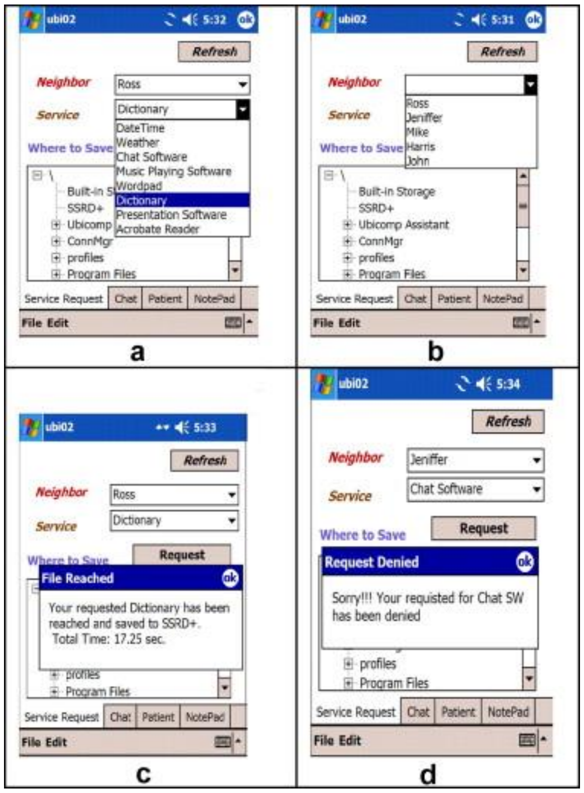
8. Evaluation

We have evaluated the performance and usability of the TSSD model by implementing prototype, designing applications, and using simulation tools. We have designed applications that use the TSSD model for device discovery and Service sharing. To estimate the overhead of using this model, we have measured the battery power consumption.

In Section 8.1, we present some screen shots taken from applications that use TSSD as a core service. Section 8.2 discusses the performance evaluation of our model.

8.1. Prototype implementation

We have implemented a prototype of our proposed model in the service discovery unit of MARKS. We have used a test bed consisting of a set of Dell Axim X30 pocket PCs (Processor type is Intel@PXA270, speed is 624 MHz). The underlying OS is WinCE and the implementation language is C# on. NET Compact framework. This prototype is also compatible to laptops, desktops, and smart phones. As the underlying wireless protocol, we have used the mobile ad hoc mode of IEEE 802.11b. Some screenshots of application using the Service discovery service are shown in Fig. 6.



Your requested Dictionary has been reached and saved to SSRD+.
Total Time: 17.25 sec.

Sorry!!! Your requsted for Chat SW has been denied

Fig. 6. Application that uses TSSD. (a) Available services. (b) Neighboring devices. (c) Successful service request. (d) Request denied.

8.2. Performance measurement

Our service discovery model is lightweight. To evaluate the performance of our model we have used battery power as a performance metric. We have constructed a test bed of seven PDAs, which are wirelessly connected in ad hoc mode. At first, we have measured the power without running our prototype. Later we have done the same thing after executing the prototype. Fig. 7 shows the remaining battery power for seven PDAs before and after running TSSD model. It shows that the battery power consumption is nominal for TSSD. The X axis denotes time in minutes and the Y axis denotes remaining battery power.

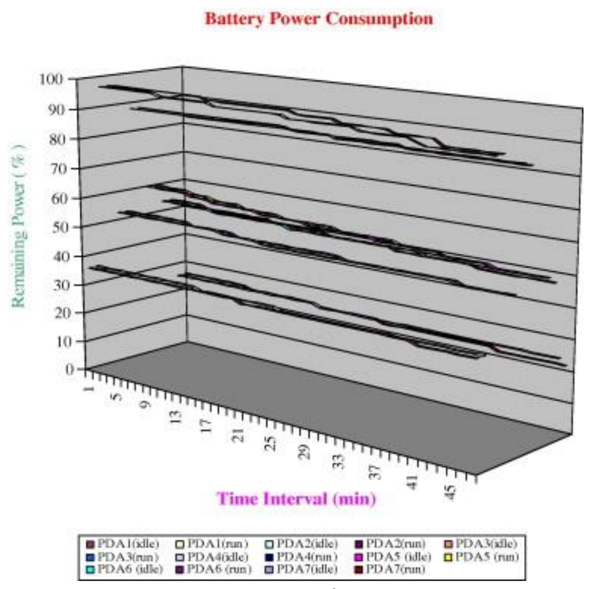


Fig. 7. Power consumption by TSSD.

To collect data for comparisons, we generated random service requests from seven devices. We measured the time required for service discovery and sharing using our model and without using our model (normal case). The services are chosen from Table 2. A portion of the collected data is shown in Table 5 (second column).

Table 5. Comparison of service time

Service name	**Time (s)**		
	Normal	**Trust**	**Trust, risk, security**
DateTime	0.1	0.105	0.105
WAV (148 KB)	0.7	0.72	0.75
Chat SW (262 KB)	0.9	0.925	0.975
Unzip SW (323 KB)	1.0	1.03	1.07
Address book (810 KB)	1.8	1.91	1.91
Dictionary (5.94 MB)	17.2	17.25	17.25
Music SW (7.96 MB)	23.6	23.66	23.66
Acrobat Reader (13.5 MB)	40	40.05	40.05

Here, we see that for normal services (e.g. DateTime, Chat & Music playing SW, etc.), encryption and user intervention is not needed. To calculate the trust value, it needs less than 60 ms. On the contrary, for the delicate address book sharing, both user intervention and encryption are needed. The encryption and trust calculation take only 110 ms, which is negligible.

We have collected data of our model using only the trust model and using trust, risk, and security model. We have compared both sets of data to estimate the overhead of using the risk and the security models. Table 5 (column 3) also lists this service time comparison.

Services with lower security levels do not experience any response time change since the services' risk model is never used. From our experiment we have found the services that require higher response time also are not affected since the risk evaluation time is negligible compared to overall service response time. From the results, it is proven that the additional calculation for risk features do not result in significant computation overhead.

9. Conclusion and future work

We have proposed the design of a service discovery model, TSSD. To facilitate service lookup and efficient information dissemination; the concept of a service manager has been introduced. To maintain the privacy of users and their willingness to share services, trust, risk, and security models have been implemented. Efficiency is obtained by adopting combinations of secure and non-secure modes of operation for service provider selection and service lookup, respectively.

We have implemented TSSD as a part of MARKS, dependable middleware designed for devices running on a pervasive computing environment. We have also implemented applications that will use the algorithms presented here. The suitability of the service discovery protocol has been evaluated using simulations [30] and designed applications. We have also studied the performance improvement of MARKS after incorporating the service discovery unit.

Our aim was to provide a security model that would work in everyday situations without compromising the performance of the device. We have designed a simple but efficient model that takes care of security related issues without causing much battery power consumption. Our model is a hybrid one in a sense that it operates both in secure and non-secure mode depending on the level of security needs for the service. By implementing a hybrid mode of operation, we have minimized the overhead of encrypting messages each time a device requests or provides services. However, when there is no prior information available, building a trust relationship is difficult. To address situations like this, we have also added a risk model that analyzes the risk of sharing a particular service and takes appropriate action. The vulnerability issues against several types of attack scenarios have been put as out of scope for this paper. This issue has been discussed in details along with solution in our previous paper [59].

Currently, our risk model uses historical data to calculate the risk factor. Inclusion of risk parameters is a new concept for ad-hoc mobile devices. This model can be made more robust using different distributions for finding out the threshold value for risk parameters that minimizes the risk of service sharing. The addition of appropriate risk parameters will make this model tremendously useful. Our existing model works for single-hop Service discovery and sharing. This model can be extended to facilitate multi-hop discovery and service sharing. Features like dynamic service integration can be included. Service integration may replace failed services or plug in more services depending on the situation without starting the entire process over. By considering the devices as cells, “Cellular automata” [57] concepts can be employed to adopt this feature. We will also work of privacy inference and preservation in future.

Acknowledgements

The authors appreciate the help of Steve Sass and Karl Stamm to improve the readability of the paper. We also thank Shameem Ahmed for implementing part of the test bed. We also appreciate the comments of Md. Munirul Haque. We also appreciate his help for formatting the paper.

References

[1] M. Weiser. **Some computer science problems in ubiquitous computing.** *Communications of the ACM*, 36 (7) (1993), pp. 75-84

[2] Pervasive Computing definition. Available from: <http://www.parliament.vic.gov.au/sarc/EDemocracy/Final_Report/Glossary.htm>.

[3] Pervasive Computing framework. Available from: <http://framework.v2.nl/archive/archive/node/text/default.xslt/nodenr-156647>.

[4] P. Robinson, H. Vogt, W. Wagealla, Some research challenges in pervasive computing, in: Post Workshop at the Second International Conference on Pervasive Computing, April 18–23, 2004, Vienna, Austria, pp. 1–16.

[5] M. Weiser. **The computer for the twenty-first century.** *Scientific American* (1991), pp. 94-104

[6] T. Kindberg, A. Fox, System software for ubiquitous computing, IEEE Pervasive Computing (Jan–Mar) (2002) 70–81. Available from: <http://www.champignon.net/TimKindberg/kindberg-fox-ieeepvc.pdf>.

[7] M. Satyanarayanan, Fundamental challenges in mobile computing, in: Fifteenth ACM Symposium on Principles of Distributed Computing, Philadelphia, PA, USA, May 1996, pp. 1–7.

[8] R. Want, T. Pering, System challenges for ubiquitous & pervasive computing, in: 27th International Conference on Software Engineering (ICSE 2005), St. Louis, MO, USA, May 15–21, pp. 9–14.

[9] F. Stajano. **Security for Ubiquitous Computing.** Wiley, New York (2002) pp. 110–111

[10] F. Stajano, R. Anderson. **The resurrecting duckling: security issues for ubiquitous computing.** *Computer*, 35 (4) (2002), pp. 22-26 Part Supplement

[11] K. Matsumiya, S. Tamaru, G. Suzuki, J. Nakazawa, K. Takashio, H. Tokuda, Improving security for ubiquitous campus applications, in: *Symposium on Applications and the Internet Workshops* (SAINT 2004), January 2004, pp. 417–422.

[12] L. Kagal, T. Finin, A. Joshi. **Moving from security to distributed trust in ubiquitous computing environments.** *IEEE Computer* (December) (2001)

[13] Y. Chen, C.D. Jensen, E. Gray, V. Cahill, J. Seigneur, A general risk assessment of security in pervasive computing. Available from: <https://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-45.pdf>.

[14] M. Nidd. **Service discovery in DEAPspace.** *IEEE Personal Communications* (2001), pp. 39-45

[15] B.A. Miller, T. Nixon, C. Tai, M.D. Wood. **Home networking with universal plug and play.** *IEEE Communications Magazine*, 39 (12) (2001), pp. 104-109

[16] W. Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, The design and implementation of an intentional naming system, in: 17th ACM Symposium on Operating Systems Principles (SOSP'99), Kiawah Island, Scotland, 1999, pp. 186–201. Availablr from: <http://citeseer.ist.psu.edu/cache/papers/cs/13731/http:zSzzSzwww.cs.cmu.eduzSzPeoplezSzbumbazSzfiling_cabinetzSz.zSzpaperszSzsosp99zSzadjie-winoto.pdf/adjie-winoto99design.pdf>.

[17] M. Balazinska, H. Balakrishnan, D. Karger, INS/Twine: A scalable peer-to-peer architecture for intentional discovery, in: *International Conference on Pervasive Computing*, Zurich, Switzerland, August 26–28, 2002, pp. 195–210.

[18] Microsoft Corporation, Universal Plug and Play Device Architecture, Version 1.0, Microsoft Co., 2000.

[19] S. Czerwinski, B.Y. Zhao, T. Hodes, A. Joseph, R. Katz, An architecture for a secure service discovery service, in: *Fifth Annual. International Conference on Mobile Computing and Networks* (MobiCom'99), Seattle, WA, 1999, pp. 24–35. Available from: <http://www.win.tue.nl/johanl/educ/2Q341/Papers/BerkeleySDS.pdf#search='An%20Architecture%20for%20a%20Secure%20Service%20Discovery%20Ser vice'>.

[20] F. Zhu, M. Mutka, L. Ni, Splendor: a secure, private, and location-aware service discovery protocol supporting mobile. services, in: Pervasive Computing and Communications, 2003 (PerCom 2003), Proceedings of the First IEEE International Conference, 23–26 March 2003, pp. 235–242. Available from: <http://www.angoya.net/lni/papers/MyPapers/ZhMN03a.pdf>.

[21] F. Zhu, M. Mutka, L. Ni, PrudentExposure: a private and user-centric service discovery protocol, in: Proceedings of the 2004, IEEE Annual Conference on Pervasive Computing and Communications (PerCom 2004), March 2004, pp. 329–340. Available from: <http://www.cse.msu.edu/zhufeng prudent.pdf>.

[22] H. Kopp, U. Lucke, D. Tavangarian, Security architecture for service-based mobile environment, in: *Third IEEE International Conference on Pervasive Computing and Communications Workshops* (PERCOMW'05),Washington, DC, USA, March 2005, pp. 199–203.

[23] F. Zhu, M. Mutka, L. Ni, Expose or not? A progressive exposure approach for service discovery in pervasive computing environments, in: *Third IEEE International Conference on Pervasive Computing and Communications* (PerCom 2005), March 2005, pp. 225–234.

[24] F. Almenarez, C. Campo, SPDP: a secure service discovery protocol for ad-hoc networks, in: *9th Open European Summer School and IFIP Workshop on Next Generation Networks* (EUNICE 2003), Hungary, September 2003.

[25] R. He, J. Niu, M. Yuan, J. Hu, A novel cloud-based trust model for pervasive computing, in: *The Fourth International Conference on Computer and Information Technology* (CIT'04), September 2004, pp. 693–700.

[26] M. Sharmin, S. Ahmed, S.I. Ahamed, MARKS (middleware adaptability for discovery, knowledge usability, and self Healing) in pervasive computing environments, in: *Third International Conference on Information Technology: New Generations*, NV, USA, April 2006, pp. 306–313.

[27]

S. Ahmed, M. Sharmin, S.I. Ahamed, Knowledge usability and its characteristics for pervasive computing, in: *The 2005 International Conference on Pervasive Systems and Computing* (PSC-05), Las Vegas, USA, June 2005, pp. 206–209.

[28] S. Ahmed, M. Sharmin, S.I. Ahamed. **ETS (efficient, transparent, and secured) self-healing service for pervasive computing applications.** *International Journal of Network Security*, 4 (3) (2007), pp. 271-281

[29] S. Ahmed, M. Sharmin, S.I. Ahamed, PerAd-Service: a middleware service for pervasive advertisement in M-Business, in: *29th International Computer Software and Applications Conference*, Edinburgh, Scotland, July 2005, pp. 17–18.

[30] OMNeT++ Community Site. Available from: <http://www.omnetpp.org/>.

[31] F. Zhu, M. Mutka, L. Ni, Classification of service discovery in pervasive computing environments, MSU-CSE-02-24, MSU, 2002, pp. 1–17.

[32] Specification of the Bluetooth System – Core, Bluetooth SIG, Version 1.1, February 22, 2001. Available from: <http://www.bluetooth.org/docs/Bluetooth_V11_Core_22Feb01.pdf>.

[33] Bluetooth Special Interest Group, SDP Specification.

[34] Jini™ Technology Core Platform Specification, Sun Microsystems, Version 1.2, December 2001. Available feom: <http://wwws.sun.com/software/jini/specs>.

[35] K. Minami, D. Kotz, Secure context-sensitive authorization, in: *Third International Conference on Pervasive Computing and Communications Workshops* (PerCom 2005), Hawaii, March 2005, pp. 257–268.

[36] F. Zhu, M.W. Mutka, L.M. Ni. **A private, secure, and user-centric information exposure model for service discovery protocols.** *IEEE Transactions on Mobile Computing*, 5 (4) (2006), pp. 418-429

[37] P. Brezillon, G.K. Mostefaoui, Context-based security policies: a new modeling approach, in: *Second IEEE International Conference on Pervasive Computing and Communications Workshops*, FL, 2004, pp. 154–158.

[38] R.K. Thomas, R. Sandhu, Models, protocols, and architectures for secure pervasive computing: challenges and research directions, in: *Second IEEE International Conference on Pervasive Computing and Communications Workshops* (PerCom 2004), FL, 2004, pp. 164–168.

[39] F. Almenarez, A. Marin, C. Campo, C. Garcia, PTM: a pervasive trust management model for dynamic open environments, *Pervasive Security, Privacy, and Trust* (PSPT 2004), MA, 2004. Available from: <http://jerry.c-lab.de/ubisec/publications/PSPT04_PTM.pdf>, accessed May 2006.

[40] F. Almenarez, A. Marin, D. Dyaz, J. Sanchez, Developing a model for trust management in pervasive devices, in: *Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops* (PERCOMW'06), 2006, pp. 267–271.

[41] N. Shankar, W. Arbaugh, On trust for ubiquitous computing, in: *Workshop on Security in Ubiquitous Computing* (UBICOMP 2002), Gteborg, Sweden.

[42] A. Tripathi, T. Ahmed, D. Kulkarni, R. Kumar, K. Kashiramka, Context-based secure access in pervasive computing environments, in: *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, vol. 00, FL, USA, 2004, pp. 159.

[43] J. Basu, V. Callaghan, Towards a trust based approach to security and user confidence in pervasive computing systems, in: *The IEE International Workshop, Intelligent Environments 2005* (IE05), UK, June 2005. Available from: <http://cswww.essex.ac.uk/Research/iieg/papers/Jisnu%20Basu%20IE05.pdf>, accessed May 2006.

[44] S. Pearson, How trusted computers can enhance privacy preserving mobile applications, in: *Proceedings of the Sixth International IEEE Symposium on a World of Wireless Mobile and Multimedia Networks* (WoWMoM'05), Taormina, June 2005, pp. 609–613.

[45] P.R. Zimmermann. ***PGP Source Code and Internals.*** MIT Press, Cambridge, MA (1995)

[46] M. Sharmin, S. Ahmed, S.I. Ahamed, SAFE-RD (secure, adaptive, fault tolerant, and efficient service discovery) in pervasive computing environments, in: *IEEE international Conference on Information Technology* (ITCC 2005), Las Vegas, USA, April 2005, pp. 271–276.

[47] M. Sharmin, S. Ahmed, S.I. Ahamed, An adaptive lightweight trust reliant secure service discovery for pervasive computing environments, in: *Fourth Annual IEEE International Conference on Pervasive Computer and Communications* (PerCom 2006), Pisa, Italy, March 2006, pp. 258–263.

[48] Available from: <http://islab.oregonstate.edu/koc/ece478/project/dos1.pdf>.

[49] W. Diffie, M.E. Hellman. **New directions in cryptography.** *IEEE Transaction on Information Theory*, IT-22 (6) (1976), pp. 644-654

[50] R.L. Rivest, A. Shamir, L. Adleman. **A method for obtaining digital signatures and public key crypto systems.** *Communications of the ACM*, 21 (2) (1978), pp. 120-126

[51] MAC (Message Authentication Code). Available from: <http://www.rsasecurity.com/rsalabs/node.asp?id=2177>.

[52] H. Krawczyk, M. Bellare, R. Canetti, HMAC: keyed-hashing for message authentication, RFC 2104, IETF, Feb 1997.

[53] V.D. Gligor, A note on the denial-of-service-problem, in: *IEEE Symposium on Security and Privacy* (SSP'83), Oakland, CA, USA, April 1983, pp. 139–149.

[54] L. Lamport. **Password authentication with insecure communication.** *Communications of the ACM*, 24 (11) (1981), pp. 770-772

[55] N.M. Haller, The S/Key TM one-time password system, in: *ISOC Symposium on Network and Distributed System Security*, San Diego, CA, USA, February 1994, pp. 151–158.

[56] M. Sharmin, S. Ahmed, S.I. Ahamed, SSRD+: a privacy-aware trust and security model for discovery in pervasive computing environment, in: *Proceedings of the 30th Annual InternationalComputerSoftware and Applications Conference* (COMPSAC 2006), Chicago, September 17–21, 2006, pp. 67–70.

[57] Available from: <http://en.wikipedia.org/wiki/Cellular_automata>.

[58] Pierre L'Ecuyer, Unifor random number generators: a review. Available from: <http://www.informs-cs.org/wsc97papers/0127.PDF>.

[59] M. Haque, S.I. Ahamed, Haifeng Li, K.M. Asif. **An authentication based lightweight device discovery (ALDD) model for pervasive computing.** *Proceedings of the 31st Annual International Computer Software and Applications Conference* (COMPSAC 2007), IEEE CS Press, Beijing, China (2007) pp. 57–64