10-1-2009

Design and Implementation of S-MARKS: A Secure Middleware for Pervasive Computing Applications

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

Haifeng Li
Marquette University

Nilothpal Talukder
Marquette University

Mehrab Monjur
Marquette University

Chowdhury Sharif Hasan
Marquette University

Accepted version. *Journal of Systems and Software*, Vol. 82, No. 10 (October 2009): 1657-1677. DOI.© 2009 Elsevier. Used with permission.

Design and Implementation of S-MARKS: A Secure Middleware for Pervasive Computing Applications

Sheikh Iqbal Ahamed

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

Haifeng Li

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

Nilothpal Talukder

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

Mehrab Monjur

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

Chowdhury Sharif Hasan
Department of Mathematics, Statistics and Computer Science, Marquette University
Milwaukee WI

Abstract: As portable devices have become a part of our everyday life, more people are unknowingly participating in a pervasive computing environment. People engage with not a single device for a specific purpose but many devices interacting with each other in the course of ordinary activity. With such prevalence of pervasive technology, the interaction between portable devices needs to be continuous and imperceptible to device users. Pervasive computing requires a small, scalable and robust network which relies heavily on the middleware to resolve communication and security issues. In this paper, we present the design and implementation of S-MARKS which incorporates device validation, resource discovery and a privacy module.

Keywords: Pervasive computing, Secure middleware, Device validation and resource discovery

1. Introduction

As computer technology advances exponentially, human–computer interaction has stepped into a new era. People might engage in many computational devices simultaneously without even the awareness of their existence. The idea of pervasive computing is that almost every device we see today will be capable of communication and function in collaboration with one another in the near future. Integrated with wireless technology, voice recognition and image processing, the goal of pervasive computing is to create an unobtrusive and always available network for all embedded devices.

The feasibility of pervasive computing (Weiser, 1993) has been established in education, healthcare, industry, and elsewhere. Like other systems, security and privacy are big concerns for the pervasive computing system. Due to lack of a fixed infrastructure for authentication and authorization, devices in pervasive computing are more susceptible to malicious snoopers. Middleware can provide a solution for this problem. A middleware can handle the security, privacy, and communication issues among devices while programmers can focus on the business logic. S-MARKS is an effort to address some of these critical issues while devices interact with one another. Some of the well known middleware for mobile devices include RCSM+ (Yau et al., 2002), GAIA (Cerqueira et al., 2001), MIT's Oxygen (Dertouzos, 1999), MARKS (Sharmin et al., 2006a,b), Mobiware (Campbell, 1997), TSpaces (Wyckoff et al., 1998), LIME (Murphy et al., 2001), XMIDDLE (Mascolo et al., 2002), PICO (Kumar et al., 2003) and ALICE (Eichberg and Mezini, 2004). Most of the middleware need fixed infrastructure support and are not suitable for ad hoc network formation required in purely pervasive environment. *Among all of the above middleware, only*Yau et al. (2002) *RCSM+ (Reconfigurable Context-Sensitive Middleware)* (Yau et al., 2002) *is designed to enableapplications that require context-awareness and uninterrupted ad hoc communication among pervasive devices. The third party applications that require more complex operations based on contextual information are facilitated by the support that RCSM+ provides. RCSM+'s focus was different. It did not address security and privacy of the devices and applications. Inspired by RCSM+ our middleware S-MARKS fills in for those critical issues.* S-MARKS is specially designed to support the secured device, service discovery, and privacy aspects of the pervasive devices. The idea of RCSM+ centers on a situation-aware Object Request Broker (ORB), and related object communication frameworks (Yau et al., 2004). Other aspects of RCSM are supporting an ephemeral and situation-triggered group communication service that facilitates ad hoc formation of communities of devices with group keys (Yau and Zhang, 2003). It also provides support for a middleware for real-time system (Yau and Karim, 2004). The Impregnable Lightweight Device Discovery Model (ILDD) (Haque and Ahamed, 2008) supports efficient handling of discovered devices with ad hoc network formation which can also be used with real-time systems. Finally, S-MARKS also incorporates support for privacy-aware application (Langheinrich, 2001) modeling not found in existing middleware.

Device discovery is an integral part in pervasive computing. Device discovery is used to identify valid neighbors before any communication among the devices occurs. Mistakenly enrolling a malicious device as a valid neighbor could lead to the collapse of the whole network. In a pervasive computing environment, devices can join and leave the network arbitrarily since the device users are very mobile. Because of the mobility, the valid neighbor list has to be updated frequently. The relatively small memory capacity and less computing power of pervasive devices also set resource limitations which require that device discovery algorithms should be simple and efficient. As a result, having a light weight protocol to validate devices before they communicate is very important.

The other important concept in a pervasive computing environment is resource discovery; it occurs after the valid neighbor list has been obtained. Resource discovery is another integral part of every device present in this environment (Kindberg and Fox, 2002). It explores the devices in the valid neighbor list for the resources available in them. Mutual dependency between devices and the ad hoc nature of the network distinguish the resource discovery in a pervasive environment from the one in a network with fixed infrastructure. Three issues need to be resolved in this type of environment. First, the connection between devices may not remain for a long span of time. After service is invoked, the device might leave the network without notice. The resource list of valid neighbors needs to be updated frequently. Second, more than one device may simultaneously request the same service of a particular device. Service requester has to decide how to choose the ultimate service provider among available candidates. Similarly more than one device may grant the same service request from the same device. How to choose the ultimate service provider among available candidates has to be decided by the service requester. A selection algorithm should be invented in order to deal with the above scenarios. Finally, security is a big issue in resource discovery (Matsumiya et al., 2004; Stajano and Anderson, 2002; Stajano, 2002). Sometimes devices have private information which they don't want to reveal to others. Sometimes devices allow others to access private services only after the positive confirmation from the device user. In the service sharing environment, trust is related to security concerns (Kagal et al., 2001; Quercia et al., 2006; Satyanarayanan, 2001). A trust model helps devices to determine whether or not to share services with other devices. However, exposure of the high security resource should seriously be considered, even for trusted devices.

The privacy of the information exchanged (Bellotti and Sellen, 1993; Beresford and Stajano, 2003; Campbell et al., 2002) among the devices has become a critical issue with the introduction of contexts (Dey, 2001). Keeping that in mind, we incorporate in S-MARKS a privacy module that provides the applications an added advantage of controlling the amount and extent of information that goes out of the device. Since, in a pervasive environment we can't leverage the third party or anonymizer (Sweeney, 2002; Gedik and Liu, 2005; Ghinita et al., 2007; Mokbel et al., 2006) to measure the effective anonymity while disclosing the information, we made some trivial assumptions while measuring the anonymity based on the number of devices in the network (Talukder and Ahamed, 2008). Only valid neighbors can share services among one another with the protection of individual privacy.

The above concerns demand a middleware which is secure by design in a pervasive computing environment. Although a good number of middleware is present (Capra et al., 2001; Sharmin et al., 2006a,b; Campbell et al., 1997; Dertouzos et al,. 1999; Wyckoff et al., 1998; Sousa et al., 2002; Murphy et al., 2001; Mascolo et al., 2002; Cerqueira et al., 2001; Yau et al., 2002; Kumar et al., 2003; Eichberg et al., 2004; Ahamed et al., 2006), none of them provide secure solutions for device validation or trust oriented resource discovery. To address the above security issue, a middleware called S-MARKS is proposed in this paper. We present the details of S-MARKS middleware from the perspective of both design and implementation, which successfully addresses the security issue already mentioned. S-MARKS has been implemented using modern software engineering techniques. Hence, we summarize the contribution of our work here:

1. In S-MARKS, we have implemented and tested (with reconfigurable parameters) a valid device discovery technique with a robust authentication mechanism, Inpregneble Lightweight Device Discovery (Haque and Ahamed, 2008) or in short ILDD, that provides support for secured group formation with the detection of malicious users.
2. We also provide support for an independent recommendation-based distributed trust management module and incorporate secured service discovery and sharing through this module in S-MARKS.
3. Our modularized design keeps device, service discovery and privacy modules independent of one another and the ORB architecture helps establish communication among the modules and the devices.

4. We incorporated an independent privacy module to support context-aware applications. It considers only available parameters in the purely pervasive environment like the number of devices and provides a privacy measure to the user and service provider in order to balance the trade-off between two parties.

The rest of the paper is structured in the following manner. Section 2 provides an illustrative example to help understand the necessity of a secured middleware. In Section 3, we discuss the requirements needed for a middleware to be secure by design. In Section 4, we present our approach and the architecture of S-MARKS. In Sections 5–7, we provide the detailed design and implementation of device discovery, resource discovery, and privacy module using a UML diagram, Design Pattern and Data Flow Chart. Section 8 demonstrates screenshots and user interactions with the developed components of S-MARKS. In Section 9, the illustrated example is reiterated to demonstrate the indispensability of S-MARKS. In Section 10, we discuss existing problems of S-MARKS and our future work. In Section 11, we address related middleware developments.

2. An illustrative example – campus ad hoc network

Alice is moving around the campus with her handheld device that shares music with people. She is already connected to an ad hoc network with some buddies in her vicinity. Bob was passing through; out of curiosity he just wanted to sneak in and know what she and the others were doing. He intends to join the ad hoc network that Alice is already in. Bob gets authenticated in the system. He sees everybody is sharing some popular music. He tries to pick out some and achieves success. He is also eager to share some of his music.

Then along came Carl during his break, wanting to be with his buddies. He is authenticated in the system. But he is not allowed to use the music service from Alice. In the past he tried to download too much music from Alice that resulted in slow response for her device and eventually a lower trust level for Carl for that particular service. Now he can download only the weather service from Alice which requires a lower level of trust.

Then, poor David, he can't even get authenticated, as he is not welcome in the group because of his bad-mouthing outside the group. Even worse, he is also good at stealing keys. He tries to sneak in through different combinations of responses during the authentication phase but still can't get through.

Earl, a new guy, opens up a service called campus book, which is intended for sharing books, and maintaining a list of potential things for exchange. The service requires profile information and some current information to join and use the service. But not everybody is interested in divulging private information. So, a user is allowed to join through some negotiation, where his or her privacy is not fully compromised, yet he or she can use the service.

3. Required features of S-MARKS

For any middleware which is secure by design, certain features and functionalities must be required. They are briefly presented below.

3.1. Valid device discovery

A device needs to dynamically discover its valid neighbors while excluding malicious neighbors in a pervasive computing environment. Certain authentication must be made before any device is accepted as a valid neighbor. A valid neighbor list should be maintained for further interaction among devices. A device will not be involved in any interaction with another device that is not present in the valid neighbor list. Since a device could leave the network at any time, the valid neighbor list should be updated after short time intervals.

3.2. Trust based resource discovery

After the device validation phase, a device faces the problem of whether or not to share the resource with a device in the trusted neighbor list. Although all the devices in the neighbor list are valid, they have different trust levels. The trust level is built upon the interaction behavior and requires periodic update. The same interaction behavior might be interpreted differently by different device users. Besides the trust level of a valid neighbor device, each service has its own security level. A service with a high security level requires a high trust value. When a device requests different resources all from the same device, some requests are granted and some are not.

3.3. Malicious recommendation handling

The trust level in resource discovery is generated based on the recommendation from others directly and indirectly. Due to malicious intention, a false recommendation might occur which requires a mechanism to handle the incident. The malicious recommendation should be managed so that the overall trust related to a service requester is not undermined.

3.4. Privacy handling

The pervasive applications may need the users' static or dynamic credentials to provide access to the services. Static credentials may include age, group, education level, etc. Dynamic credentials refer to contextual information like location, activity state, etc. (Dey, 2001). The static profile and contextual information are exchanged often among devices in the dynamic environment. The owner of the information desires control of what goes out of the system. On the other hand, the service provider requires a certain level of quality of the information disclosed in order to provide the service. The greater the amount of information disclosed, the higher the chance of re-identification of the user even if the identity of the user is not disclosed (Talukder and Ahamed, 2008). The balance hinges between the user's desire to control the *anonymity level* of the information disclosed and the provider's requirement of meeting a *quality level* of that information. The privacy module incorporated in S-MARKS handles the trade-off between these two that varies among applicative contexts. The approach to measure the quality of information and level of anonymity of disclosed information can be found at Section 7.

4. Our approach

Our middleware S-MARKS, as shown in Fig. 1, consists of both core components and general components (or services). Core components include ILDD (**I**mpregnable **L**ightweight **D**evice **D**iscovery) (Haque and Ahamed, 2008), SSRD (**S**imple and **S**ecure **R**esource **D**iscovery) with Trust Management (Sharmin et al., 2005, 2006a,b) and Security Management, Privacy module (Talukder and Ahamed, 2008) and ORB (**O**bject **R**equest **B**roker). Communication refers to message or file transfer between devices. It is an open framework so that other services or modules can be embedded easily. Right now the core component has been developed. Later it would support context-service, and MaRcHer (**Ma**licious **Rec**ommendation **H**and**ler)** which we are currently working on.

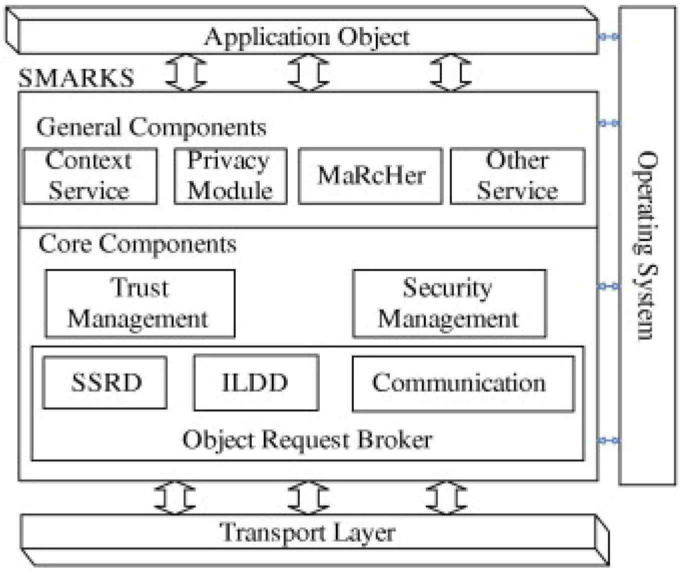


Fig. 1. Architecture of S-MARKS.

4.1. Class diagram of S-MARKS architecture

The class diagram, as shown in Fig. 1, gives the classes of core components in S-MARKS, their interrelationship, attributes, and methods of the classes.

DeviceDiscovery, ResourceDiscovery, Privacy Manager – these three classes realize ILDD, SSRD and Privacy Module, respectively. Resource Discovery uses Trust Manager and Security Manager to process a request for a certain resource. ILDDType is a request type for both a newcomer joining and group update requests used by DeviceDiscovery. The Group boolean variable (Bryant, 1986; Gossett, 1908; Zultner, 1999) determines whether the device is making a join or a group update request. Privacy Module requires support from the context-service and static profile service to collect information and can be used by the applications.

The S-MARKS class design follows observer pattern. Observer pattern defines a one-to-many dependency between objects so that when an observed object changes its state, all of its observers would automatically be notified and updated with the new data from the observed object. The essence of this pattern is that one or more objects (Observers) are registered with the target object (Subject) to observe the event which might be raised by the subject. Fig. 2 shows the Observer Pattern in ILDD. On the other hand, the application relies on the Privacy Manager to disseminate contextual and static information to external entities to avail services from them. The application provides the service requirement from the external entity, whereas the privacy module consults with specific parameters of the service, the privacy preference of the user, and measures quality of service and anonymity measure. Fig. 3 shows the class diagram of the privacy module.

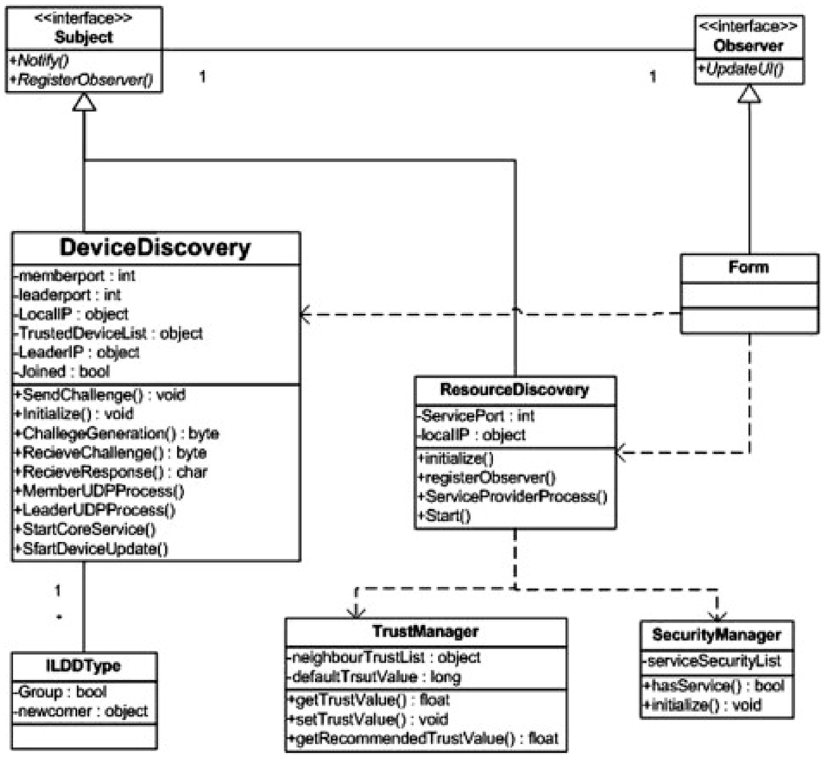


Fig. 2. Class diagram of device and resource discovery in S-MARKS.

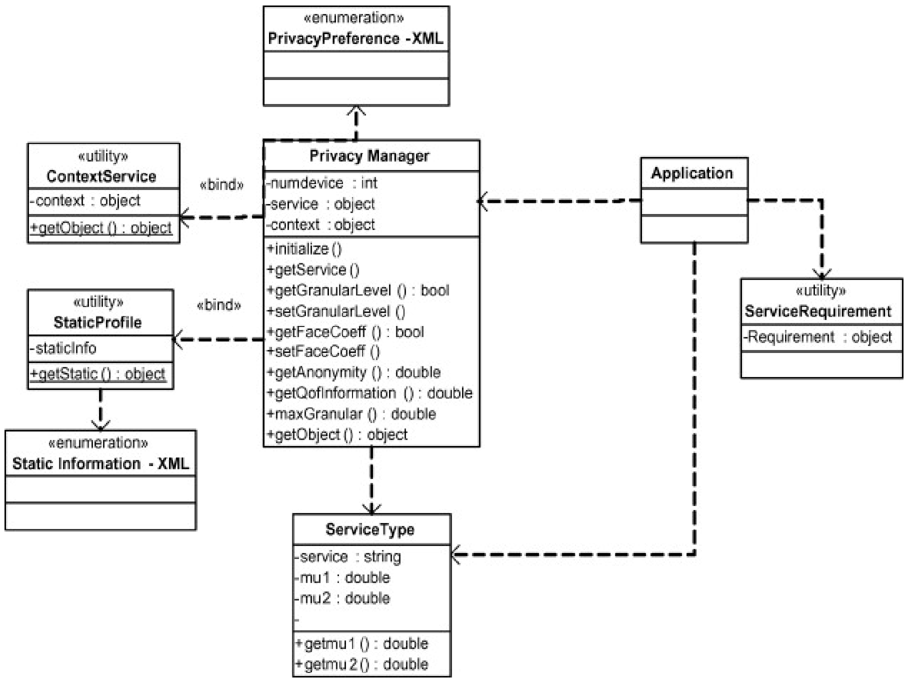


Fig. 3. Class diagram of privacy module interacting with application.

The user needs to know what is going on when they use S-MARKS. For example, when a new device requests to join the established group, the user should be notified that a device is requesting to join. In the user interface, a message bar shows S-MARKS activities. When the state of S-MARKS changes, S-MARKS should be able to update the form's message bar to reflect the change of the current status. In S-MARKS, a user form is the observer while the modules 'DeviceDiscovery' and 'ResourceDiscovery' are the subjects. These subjects register the user form as the observer. When these two modules receive messages from group members or newcomers, they notify the registered observer to update correspondingly.

The application can directly use Privacy Manager when attempting to exchange information across devices. If the application wants to use a service from the neighborhood, the device needs to know about the service requirements (the quality of information required to allow access), the service pattern parameters (μ_1, μ_2 – refer to Section 7 for details) and the disclosure levels of the information. Privacy Manager binds the context-service and static profile class for contextual and static information, respectively.

4.2. Communication stack and message format

The S-MARKS layer sits on top of the Transport layer (Fig. 4). S-MARKS has three different services running in its process: ILDD, SSRD and Communication service. The architecture of ILDD is shown in Fig. 5. The Application layer is on top of the S-MARKS layer. Each application service uses the Communication service of S-MARKS and the Communication service knows the application to which it should pass data. For this, each application tells which port the Communication service should listen or connect to.

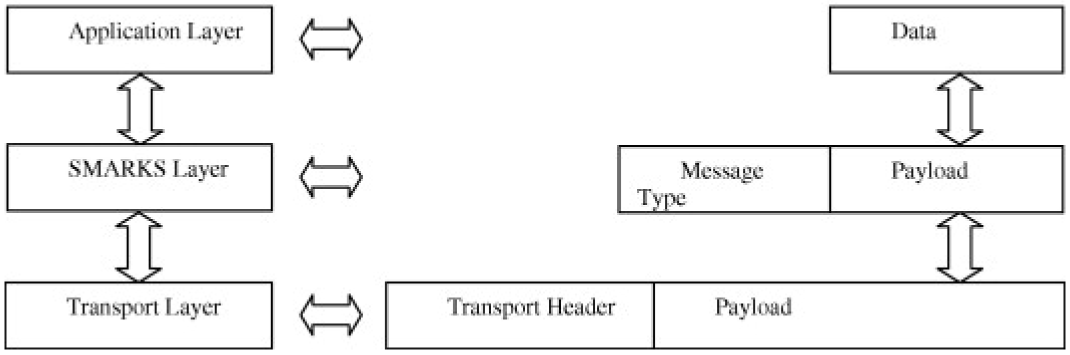


Fig. 4. Communication stack and message format.

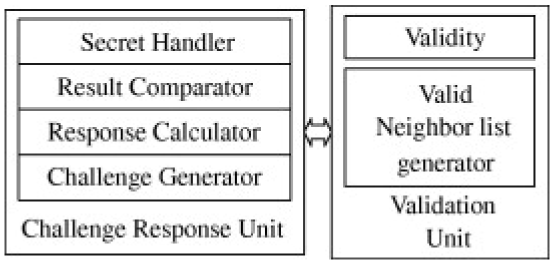


Fig. 5. Architecture of ILDD.

Depending on the application type, common service can use UDP or TCP. For streaming video or audio, or IP-telephony, it will use UDP and for internet service, mail service etc, it will use TCP. For ILDD and SSRD we use UDP. Each device in the network takes part to validate a device and to recommend trust for a device. To make such challenge, response and recommendation process fast, we decided to use UDP. We know that UDP is not reliable but these communications can be re-initiated in case of failure. Table 1 shows the port number each service uses.

Table 1. Port number assigned to services.

Port	Service
7890	ILDD service used by common device processes
7895	ILDD service used by the leader device process
7900	SSRD service
Others	Common service

In the ILDD service, port number 7890 is used by the common member process, which deals with the general messages between devices. 7895 is used by the leader process, which only runs in the leader device, for the purpose of updating the valid neighbor list. In the SSRD service, port number 7900, which deals with the resource request, is used by the resource discovery service. Table 2 shows the common message types used in the S-MARKS layer.

Table 2. Common message types in the S-MARKS layer.

Message code	Message type
11	Service request
12	Service grant
13	Service discovery done
20	Leader initialization
21	Challenge
22	Response to a challenge
23	Recommend
24	End of challenge and response
25	Update group list
26	Newcomer acceptance or rejection
27	Newcomer requests for joining
28	Add or remove IP
∗4∗	Other application specific message type used by communication service

5. Valid device discovery

In order to restrict interactions to only valid devices, we include in S-MARKS a unit to discover and authenticate/identify valid devices using a model named ILDD (**I**mpregnable **L**ightweight **D**evice **D**iscovery) (Haque and Ahamed, 2008). In the following section, we present the motivation for implementing ILDD in S-MARKS and give an overview of the approach in Sections 5.1 and 5.2. Later, from Section 5.3–5.7, we give design and implementation details of how ILDD is incorporated into S-MARKS. Finally, in Section 5.8 we present the evaluation of this unit.

5.1. Motivation

Pervasive devices have power, memory and computational constraints and because of that, implementation of standard cryptographic authentication protocols – symmetric (e.g. DES, AES) or asymmetric (e.g. RSA), is not feasible. With that in mind, we adopted in S-MARKS ILDD (Haque and Ahamed, 2008), a novel variant of Hopper and Blum's secure human authentication protocol (HB protocol in short) (Hopper and Blum, 2000, 2001). ILDD is a lightweight, symmetric-key authentication protocol using only AND, XOR and rotation operation. Although there are other variants of HB (Juels et al., 2005; Munilla and Peinado, 2007), we adapted ILDD as it is suited for ad hoc networks. Actually, HB protocol is based on a single server–client scenario but an ad hoc network is formed by a handful of devices joining and leaving arbitrarily. Instead of providing a challenge from a single server, ILDD modified the communication such that each valid device sends only one challenge to every other devices in the network.

5.2. Overview of ILDD

In this section, we discuss HB protocol at the beginning. Then, we shed light into the LPN (Learning Parity in the presence of Noise) problem, which is intended to render the adversary's job (compromising secret key) harder. Finally, we provide a summary to the ILDD protocol adopted in SMARKS.

Suppose Alice (A) wants to authenticate herself to Bob (B) and they both share an n bit secret x. B sends a random nonce $a \in \{0, 1\}^n$ as a challenge to A. A computes binary inner product $a\psi \leqslant \leftarrow x\psi$ and ψ sends ψ the ψ response $\psi \leftarrow a\psi \leqslant \leftarrow x$, i.e., the parity bit to B. B also computes the same and accepts A, if the parity is correct. This challenge and response round occurs q number of times.

Attack: The probability that an imitator can correctly guess the parity bits for all q rounds, is 2^{-q} (Abramowitz and Stegun, 1972). However, an eavesdropper can calculate the secret through Gaussian elimination method, if he can capture q valid challenge and response pairs, when $q \geqslant n$.

A can introduce noise in the response to thwart the attack. Now, A will send back the response as $(a.x)\oplus v$ where $v = \{0, 1|\text{Prob}(v = 1) = \eta\}$ and η is the noise. B will now accept A, if fewer than ηq responses are incorrect. With the introduction of noise, the adversary now needs to solve an instance of LPN problem at each round, which is computationally intractable (Berlekamp et al., 1978). The best known solution for a random LPN instance requires computational complexity of $2^{O(\frac{n}{\log n})}$ (Blum et al., 2003).

In ILDD (Haque and Ahamed, 2008), all the authenticated devices possess two common secrets: a key x and a function f: $\{0, 1\}^n \rightarrow \{0, 1\}^n$. After each authentication phase, a new x is generated from $f(x)$. If there are malicious devices present in the network, ILDD will prevent them from bypassing the authentication phase and hence, they cannot know x and $f(x)$. We store both x and f in a Trusted Store so that they cannot be compromised.

Suppose μ devices are authenticated and appear as valid in the network. A leader node (Haque and Ahamed, 2008), chosen based on battery power and trust level, sends a challenge to all the listed valid devices. Upon receiving the challenge each device calculates new x from $f(x)$. Each device now sends challenges a_1, a_2, a_3, ... to $\mu - 1$ other devices, where $a_i \in \{0, n\}^n$. Each device calculates $(ai{\cdot}x)\oplus v$ where $v = \{0, 1|Prob(v = 1)\eta\}$ and η is the maximum allowable percentage of noise (intentional incorrect answer). Then it sends the response back to the device generating the challenge. If a response is accepted by a device, it sends true recommendation for that responder to the leader node. A leader node accepts a device if the number of valid recommendation is $V \geqslant ceil((1 - \eta) * (\Delta - 1))$, where Δ denotes the number of valid devices in the network.

Attack: Let us assume, there are 101 valid devices in the network and each device can give 10 response out of 100 challenges with incorrect answers as η = 0.1. Suppose, among 101 valid devices, two devices are malicious. That is, upon receiving a valid response from a device they may give false recommendation about that device. Because of their presence, leader node will receive 88 true recommendations for a non malicious user and the user will be discarded from the valid device list. The reason is that at least 90 true recommendations are required for a device to be valid.

To prevent such unwanted scenario, ILDD considers that a device would be listed as valid if the number of true recommendations, $\geqslant ceil((1-\eta) * (\Delta - 1))$, where Δ is denoted as the expected number of malicious devices which are authenticated as valid.

So far, we have assumed that x and $f(x)$ are known by the devices prior to the authentication. In reality, SSL/TLS handshaking protocol (using public key cryptography) is used to negotiate secret x and $f(x)$ when a device registers itself for the first time. Once registered, it will be able to regenerate secret x from f and the costly SSL/TLS handshake will not be required for any future communication.

Small and large network: ILDD assumes separate models for small and large networks. The model discussed in the previous section is intended for large networks. For a small network, with a fewer number of devices, ILDD uses the following equation to consider a device to be a valid one:

$$V = k$$

In other words, for a small network, ILDD eliminates the need for noise η and Ω. This is because the noise is only required to prevent the attacker from compromising the key through Gaussian elimination method. When the key size is considerably greater than the network size and the periodic alteration of the key is performed using $f(x)$ after each authentication phase, the attack is not necessary to consider.

On the other hand, in a large network, the key size can be greater or less than the network size (see the Section 5.8 for the optimal length of the key size). In this case, ILDD considers the following equation:

$$V \geq ceil((1-n) * (\Delta - 1))\Omega$$

5.3. Design objective

ILDD is implemented as a service of S-MARKS. When S-MARKS is loaded into memory of devices, the service is started automatically. Any time a new version of S-MARKS is released, we check it in the repository and the user can check it out as a DLL file.

ILDD service has following method: Start method and Shutdown method. After ILDD service is started, the leader devices and member devices use it to update the valid neighbor list.

5.4. New device requests to join

The sequence diagram in Fig. 6 describes how a newcomer enters an already established network. It is based on the assumption that all the devices use ILDD. Below we describe each step in the sequence diagram.

Step 1: A device requesting to join network broadcasts its request.
Step 2: A leader device responds to the newcomer's request.
Step 3: The leader device sends the newcomer request information to all the devices on the network. Then the leader device waits for the recommendation about the newcomer from all the network devices.
Step 4: Member devices receive the newcomer information from the leader device. Then the member devices use **Challenge Generator** to generate a challenge.
Step 5: Member devices use **Response Calculator**, which invokes **Secret Handler** to retrieve the secret, to calculate the response for the challenge that **Challenge Generator** has generated. Then they store the response. All the authenticated devices have the same secret *x*. If devices' secrets are the same and challenges for each device are the same too, the devices should calculate the same response. If the device user has configured the device with the correct secret, the device would be authenticated.
Step 6: Member devices send the challenge to the new comer.

Step 7: The newcomer receives the challenge from the member devices. Then the newcomer uses **Response Calculator** to calculate the result.

Step 8: The newcomer sends back the response to corresponding member devices.

Step 9: Member devices receive the response from the newcomer. Then they use the **Result Comparator** compare the received response with their stored responses. **Result Comparator** will get the recommendation for the newcomer.

Step 10: Member devices send the recommendations to the leader device.

Step 11: The leader device receives the recommendations from all the authenticated member devices. Then the leader device validates the newcomer by comparing the number of positive recommendations with the threshold value.

Step 12: If the newcomer is a trusted device, then the leader device would update the valid neighbor list to include the newcomer.

Step 13: If the newcomer is a trusted device, the leader device broadcasts the newcomer's IP address to all the existing authenticated member devices. By broadcasting only the newcomer information, we can avoid a malicious device getting all the other devices' information. The leader device also sends trusted neighbor list to the newcomer.

Step 14: If the newcomer is a trusted device, all the current authenticated devices add the newcomer to the existing trusted neighbor list. The newcomer gets the updated trusted neighbor list from the leader device

Step 15: The newcomer update is finished.

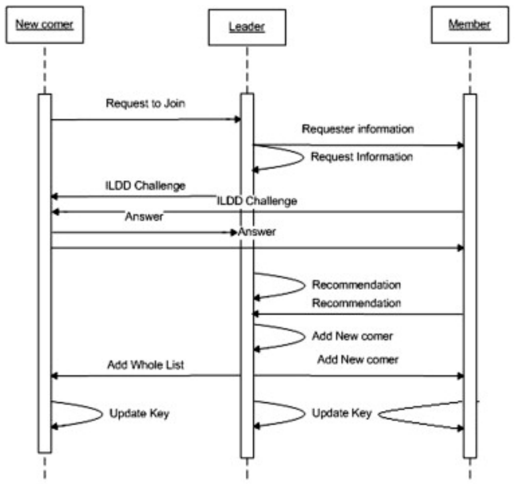


Fig. 6. New comer update interaction diagram.

5.5. A Member requests group list update

The sequence diagram in Fig. 7 describes how a leader device carries out a group neighbor list update. The process runs in a certain interval, and it can be started by a leader device manually (see Fig. 8)

Step 1: The leader node broadcasts a group update message to all its current authenticated devices.
Step 2: Each device generates challenges for all the other devices in the valid neighbor list using **Challenge Generator**.
Step 3: Each device sends a challenge to all the other devices in the valid neighbor list. Then it uses the **Response Generator** to calculate the corresponding response for each challenge it has sent out, and then stores the response.
Step 4: After receiving the challenges from all the other neighbors, each device uses its own **Response Generator** and the secret to generate a response for each challenge.
Step 5: Each device sends back the response to the challenge sender.
Step 6: Each device uses **Result Comparator** to compare the received response and stored correct response. Then it generates recommendations for all the other devices.
Step 7: Each device sends out recommendations for all the other devices to the leader device.
Step 8: The leader device receives all recommendations from all the other devices. Then the leader device compares the number of positive recommendation for each device with a predefined threshold to validate the device.
Step 9: The leader device updates the valid neighbor list, and sends the list to all the neighbors individually. Broadcasting is forbidden here since eaves-dropping may happen.
Step 10: The neighbor list of the authenticated devices get updated.
Step 11: The whole group update is finished.

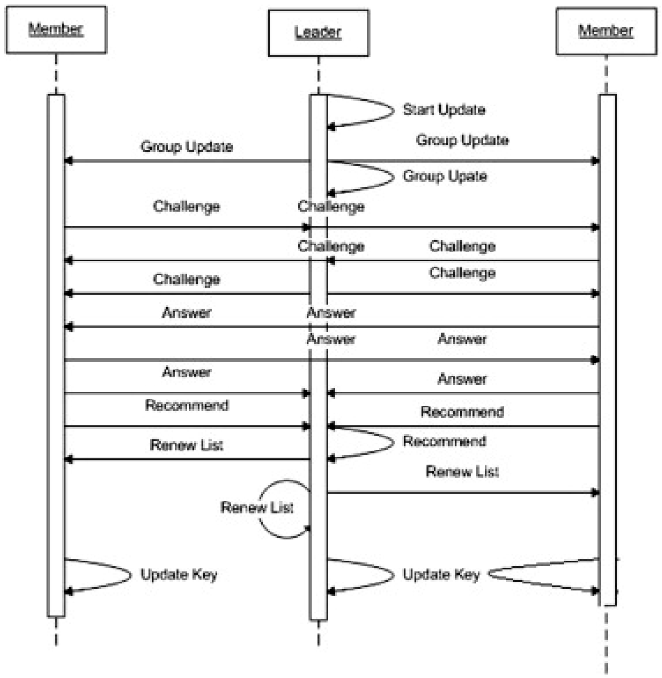


Fig. 7. Group update interaction diagram.

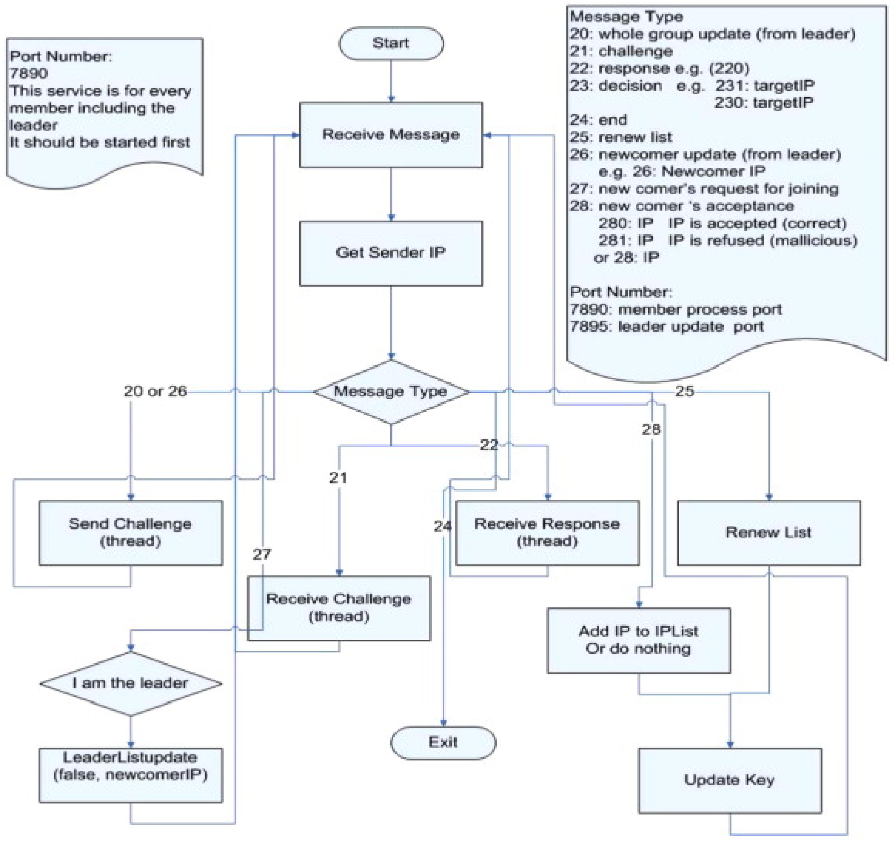

Port Number:
7890
This service is for every member including the leader
It should be started first
Start
Receive Message
Get Sender IP
Message Type
Message Type
20: whole group update (from leader)
21: challenge
22: response e.g. (220)
23: decision e.g. 231: targetIP
230: targetIP
24: end
25: renew list
26: newcomer update (from leader)
e.g. 26: Newcomer IP
27: new comer's request for joining
28: new comer 's acceptance
280: IP IP is accepted (correct)
281: IP IP is refused (mallicious)
or 28: IP
Port Number:
7890: member process port
7895: leader update port
20 or 26
25
28
22
21
24
27
Send Challenge (thread)
Receive Response (thread)
Renew List
Receive Challenge (thread)
Add IP to IPList Or do nothing
I am the leader
LeaderListupdate (false, newcomerIP)
Exit
Update Key

Fig. 8. Data flow diagram of the core device discovery process.

5.6. Data flow chart design

The data flow chart is helpful to analyze the event sequence and verify the program logic. ILDD architecture comprises two parts: the Challenge Response Unit and the Validation Unit. In this section, we show the data flow chart to analyze the event sequence and verify the program logic. In the implementation, MemberUDPProcess and NeighborListUpdate are two different processes corresponding to these two units.

MemberUDPProcess deals with nine types of messages. ILDD takes corresponding action based on the type of message it receives. Each message received would trigger a thread managed by the thread pool. For example, a message "21:challenge" could lead the device to generate a thread "Receive Challenge". Again, the thread "Receive Challenge" gets a challenge from the sender, calculates a result based on its secret, then sends the result back to the challenge sender. Another example is message "27" which represents a newcomer joining request. Only leader device responds to this message. Once a leader device receives such a message, it would deposit the request into a request buffer. Another thread named 'NeighborListUpdate' keeps scanning the request buffer for a newcomer's request to join. Once such a message is found in the buffer, a thread named 'DeviceUpdate' would be started.

MemberUDPProcess works as follows:

Step 1: Receive a message from the bonded port 7890.
Step 2: Retrieve the sender's IP address.
Step 3: Determine the message type.
Step 4: Take the corresponding action based on the message type. After receiving message '24', exit the thread.

"NeighborListUpdate" is a thread which only runs in the leader device. This thread corresponds to the Validation Unit which deals with recommendations for a newcomer from all existing authenticated devices. "NeighborListUpdate" works as follows:

Step 1: Scan the newcomer request buffer for any request message.
Step 2: Initialize an update message.

Step 3: If there is any request in the buffer, retrieve the request and set the update message type to be newcomer update.
Step 4: If there is no request in the buffer, set the update message type to be group update.
Step 5: Call "DeviceUpdate" method, in which the update message is the parameter.
Step 6: After a predefined amount of time, go to Step 2.

"DeviceUpdate" is the main method called by thread "NeighborListUpdate". Fig. 9 shows the data flow of the 'DeviceUpdate'. It works as follows:

Step 1: Specify the leader device's own IP as the Leader IP which would be sent to a newcomer.
Step 2: Start 'LeaderUDPProcess' thread, which processes recommendations from authenticated neighbors.
Step 3: Determine the valid neighbor list update type. Then send a corresponding update message to authenticated neighbors. Group update message type is 20. Newcomer update message type is 26.
Step 4: Wait for "LeaderUDPProcess", which stores recommendations into tables, to finish.
Step 5: Generate a new neighbor list based on the number of positive recommendations and the predefined threshold. The table 'table_all' contains the information about how many recommendations for a specific device are received. The table 'table_correct' contains the information about how many positive recommendations for a specific device are received. The ratio of the positive recommendation number to the whole recommendation number would be compared with the predefined threshold.
Step 6: Send the valid neighbor list to all the members.

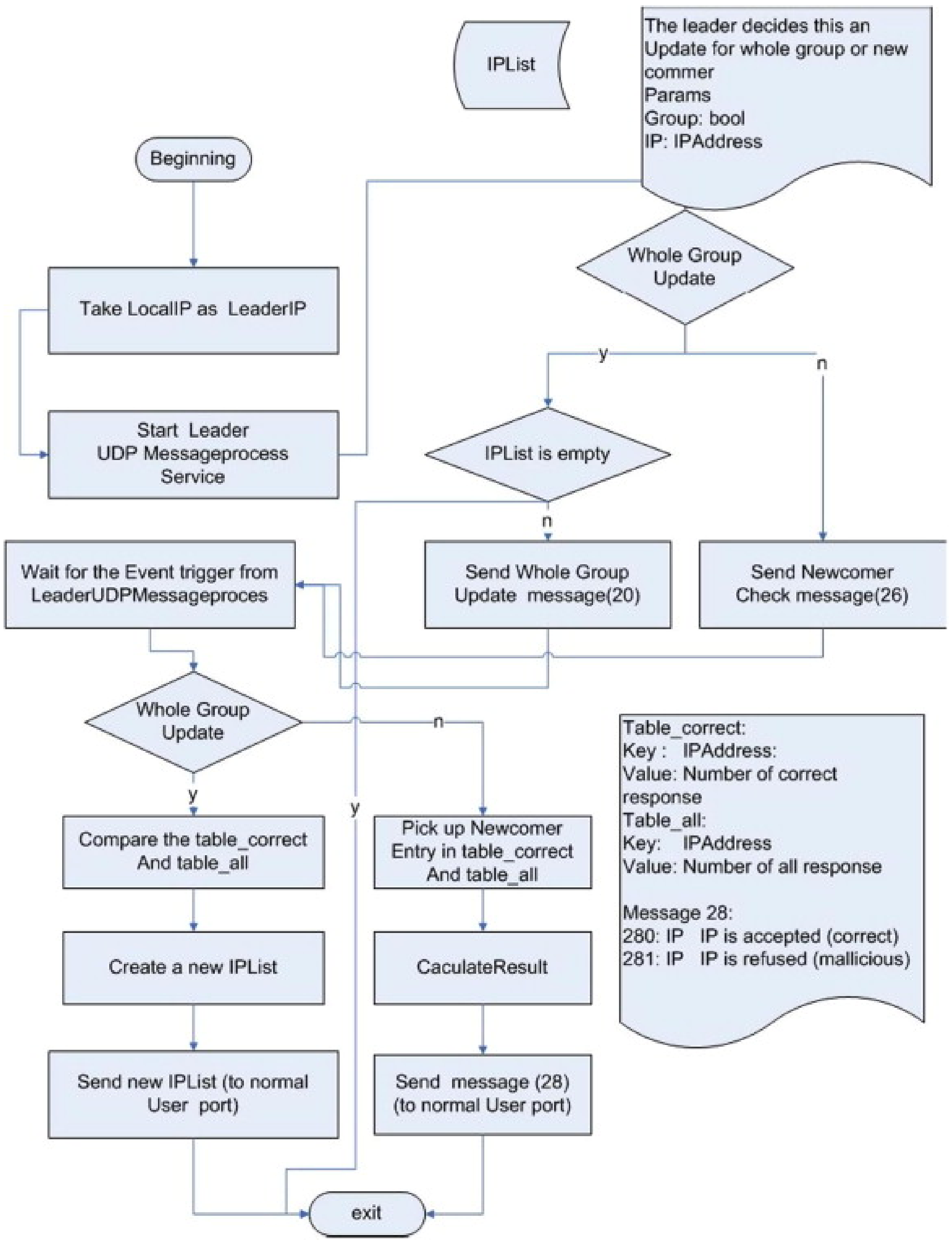

IPList
The leader decides this an Update for whole group or new commer
Params
Group: bool
IP: IPAddress
Beginning
Take LocalIP as LeaderIP
Start Leader UDP Messageprocess Service
Whole Group Update
y
n
IPList is empty
n
y
Send Whole Group Update message(20)
Send Newcomer Check message(26)
Wait for the Event trigger from LeaderUDPMessageproces
Whole Group Update
n
y
Compare the table_correct And table_all
Create a new IPList
Send new IPList (to normal User port)
Pick up Newcomer Entry in table_correct And table_all
CaculateResult
Send message (28) (to normal User port)
Table_correct:
Key : IPAddress:
Value: Number of correct response
Table_all:
Key: IPAddress
Value: Number of all response
Message 28:
280: IP IP is accepted (correct)
281: IP IP is refused (mallicious)
exit

Fig. 9. Data flow diagram of device update.

LeaderUDPProcess is a thread started by DeviceUpdate. Its main responsibility is to receive recommendations from authenticated neighbors. It stores positive and negative recommendations into different tables, which are used later by the 'DeviceUpdate' method for the further processing. LeaderUDPProcess would not exit until it receives a message '24'. When 'LeaderUDPProcess' exits, the blocked 'DeviceUpdate' thread would wake up and continue to process the recommendations.

MemberUDPProcess takes different actions based on the message type. Usually each action is a separate thread. There are several actions which need to be illustrated.

5.6.1. Send challenge

Each time one device receives a message of type 20, it is notified to send a challenge to the target device. First it would clear a response table. The response table is a Hash map table in which the key is the target device's IP address while the value is the response of applying LPN algorithm to the challenge. The result is 0 or 1. The response table is used later to generate a recommendation for the target device. Secondly, the device chooses the sending target based on the message information. If the update is a group update, all devices in the valid neighbor list are the targets. Otherwise only the newcomer is the target. Thirdly, a challenge string is generated, and then LPN algorithm is applied with a secret to produce the correct response for the challenge. The correct response then is stored into the response table. Finally, the message "21" which contains the challenge is sent to the target.

5.6.2. Receive challenge

Each time one device receives a message with type "21", it is notified to start a thread 'Receive Challenge'. In a newcomer update, only the newcomer receives the challenge. In a group update, all the authenticated member devices receive challenges. First the device would determine the update type. If it is a group update, the device has to know whether the challenge is coming from the valid neighbor list. The device would not respond to a challenge from an unknown device, which might be malicious. If it is a newcomer update, the newcomer answers all the challenges. Secondly, the device would apply the LPN algorithm to the challenge with the predefined secret, and then get the response for the challenge. Finally it would send back the response to the sender with the message type "22".

5.6.3. Receive answer

When MemberUDPProcess thread receives a type 22 message, the device starts "Receive Answer" thread. First the device finds the corresponding entry for the challenge it has sent out. Secondly, it retrieves the correct response for the challenge from this entry. Thirdly, it compares the correct response and received response, and then generates a recommendation. Finally, it sends out the recommendation to the leader device.

5.7. Implementation of the LPN secret

The LPN secret is a fixed byte array which is known by group members in advance. All the responses are generated based upon this secret. Theoretically, LPN secret's length should increase automatically if the group size is increased sharply. The length of the secret should guarantee that the secret can not be decrypted by a malicious user. The algorithm for calculating the response uses the bitwise operation is illustrated step by step:

Step 1: Get the result of an AND operation between a challenge and a secret.
Step 2: XOR operations are performed on the result of Step 1 in a sequential manner such that the result of an XOR operation with bit one and bit two is performed with bit 3. That result is calculated with bit 4, and so on.

We developed a simplified Challenge Response Unit for our S-MARKS. The **Secret of ILDD** is a fixed-length byte array which can be configured by the user. **Challenge Generator** generates a challenge which is a random byte array with the same length as a preconfigured secret. **Response Calculator** applies bitwise AND, XOR and Rotate operations to a challenge with a preconfigured secret, and then generates a one-bit response.

The following code illustrates how to apply bitwise operation to generate a response.

Algorithm *ResolveChallange*(challenge)

```
result ← Perform bitwise AND between challenge and secret
count ← 0
k ← result
while k > 0
 if (k modulo 2 = 1)
 count ← count + 1
endif
 k ← k/2;
end
if count modulo 2 = 0 return 0
else count modulo 2 = 1 return 1
endif
```

5.8. Evaluation

This section includes study on optimization of several parameters in ILDD with a network of different number of devices. The parameters such as length of secret x, η and Ω can be tuned to achieve this optimized performance of the network.

5.8.1. Optimal length of secret x

In Section 5.2, we presented the definitions and equations for ILDD considering small and large networks. We showed that in a small network, ILDD doesn't incorporate noise. In the case of a large network, the key size can be greater or less than the network size and passive snooper can guess the key. The advantage of incorporating noise in the responses is that the adversary cannot be sure whether he was able to obtain the correct key. If the number of devices is considerably greater than the key size n and if the adversary captures all the challenge and response pairs, he may become lucky in number of occasions and can validate himself.

Our preliminary result (Fig. 10) shows that for a network of 40 devices, the optimal value for key size n is 32 bits where the success rate for the adversary is almost 0%. Thus the leader node is required to anticipate the size of the network and determine the key length accordingly.

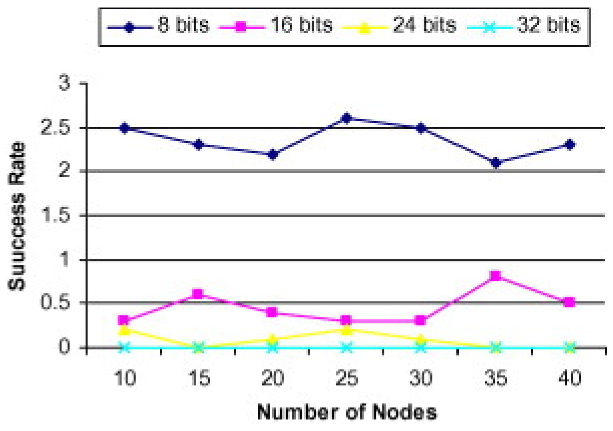


Fig. 10. Success rate in regenerating secret *x* through arbitrarily captured challenge-response pairs with increasing number of nodes.

5.8.2. Optimal value of *Ω*

In case of small network the key size is considerably greater than the network size, i.e. if the key length is 32 bits there is no more than 10 devices. With only 10 challenge-response pairs captured by the adversary and with key changed after every phase, the adversary cannot determine the actual key. For this, small networks do not use noise or *Ω* in their equation (see the equation in Section 5.2). In case of large network, the use of noise is enough to solve Gaussian elimination problem, if the adversaries were only passive snoopers. However, by considering an active adversary who maliciously gives false recommendations, ILDD introduces *Ω* (details in Section 5.2) in large network equation. When an adversary gives false recommendations some valid device gets left out in the authentication phase.

Our preliminary experiment (Fig. 11) shows that for large network of 50 valid devices, if *Ω* = 3, the active adversary will not be able to discard any valid device from the list of valid devices. It also shows that with the network getting larger, the malicious devices get less advantage. In essence, ILDD (Haque and Ahamed, 2008) shows that if the network has more than 27 devices with *Ω* = 3 i.e. 3 malicious devices present, they cannot affect the authentication of any valid devices.

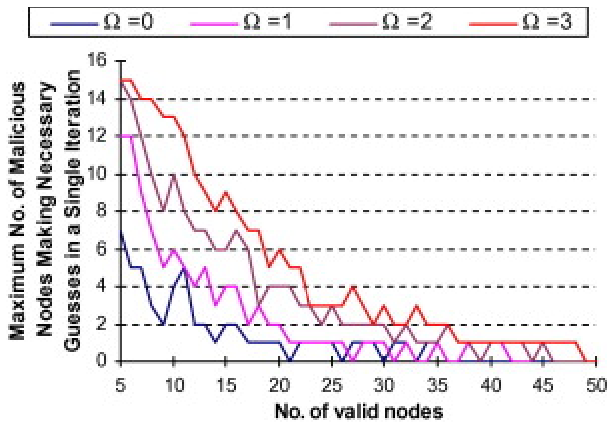


Fig. 11. Maximum number of malicious devices in a single iteration that made necessary number of correct guesses to join the network.

5.8.3. Switching between small and large network

Previous two evaluations show how variations occur in terms of key lengths and equations (especially the value of *Ω*) for small and large networks. Therefore, decision has to be made as to when we should switch from small network to large network equation. In the small network equation (in Section 5.2) there is no use of noise or *Ω*, and for that reason, to eliminate Gaussian elimination problem, ILDD has to considerably increase the key length with the increase of the number of devices. For a large network, the equation for authentication is robust and can handle increasing number of devices with almost constant key size *n*. Response time, therefore, in small network increases exponentially, whereas, in large network it remains almost constant or increases linearly (see Fig. 12).

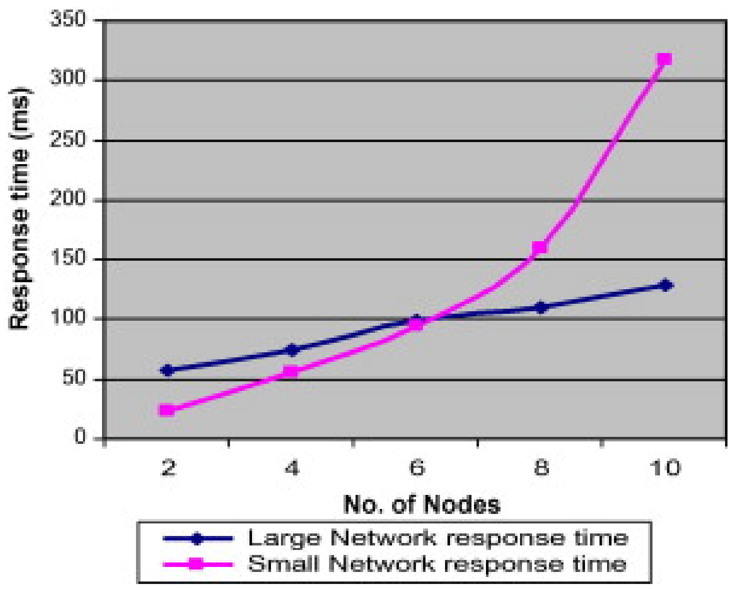


Fig. 12. Timing comparison between large and small network.

For the small network model the key length is such that malicious eavesdropper cannot guess by capturing transactions. But because its response time increases exponentially we need to change the mode to large network. Experiment shows (Fig. 13) that the number of malicious devices who can guess in a single iteration is either 1 or 0 for number of devices more than 21.

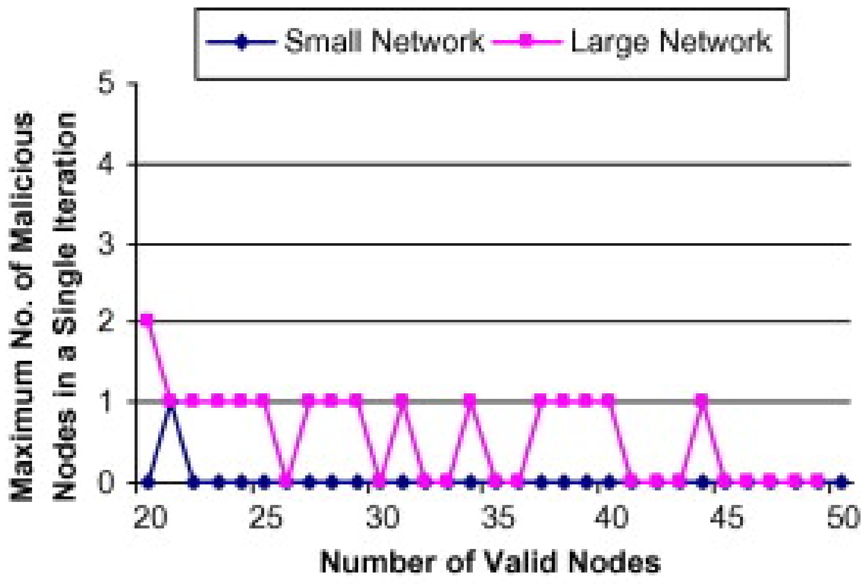


Fig. 13. Maximum number of malicious devices in the network by making necessary number of correct guesses in small and large network models.

6. Trust based resource discovery

Resource discovery is a core component in S-MARKS. We developed a trust-based resource discovery model named SSRD (**S**imple and **S**ecure **R**esource **D**iscovery) (Sharmin et al., 2006a,b). This model has two functional subunits: a trust management unit and a security management unit. Fig. 14 provides a quick overview of the units.

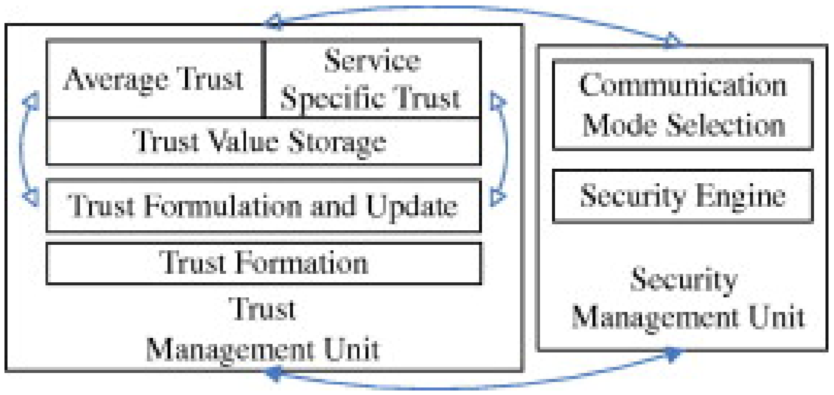


Fig. 14. Architecture of SSRD.

Following is the trust management unit in brief:

6.1. Trust management unit

The trust management unit maintains a trust level list for all the authenticated neighbors in which 0 represent complete distrust and 1 represents complete trust. A device that has just passed the validation phase (ILDD) and has no prior interaction records will have a trust value of 0.5. This trust model is both reflexive and transitive, which means the trust value of a device relies heavily on suggestions from other devices (if γ denotes the trust value of A on B and δ denotes that of B on C, then the trust value of A on C is a function of γ and δ). The trust level thus maintained of different contexts for the devices is used later on for secured service discovery. We consider the range of trust value as [0, 1]. Although there are approaches on trust bootstrapping (Quercia et al., 2007) in the literature, we consider the initial trust value as 0.5 for simplicity reasons. Since this node does not have any prior interaction records or known history, they can be neither trusted nor distrusted. Let us look at a brief classification of trusts managed by the trust management unit.

6.1.1. Direct trust

Direct trust evolves from a node's direct experience with other nodes. As a node interacts with other nodes in the network, its direct trust value for each of the other nodes changes based on the satisfaction level of the interactions. It is the most reliable portion of overall trust. This direct interaction in Fig. 15 is shown by a direct link between A and B in the topology of interaction records. The binary operator *T* indicates the trust relationship. Each node has a list of available contexts or services (c_1, c_2, c_3, … , c_i, … , c_k). In our model we consider the following notations:

$D(AT(c_i)B)$ = Direct trust of A on B for context c_i.

$D(ATB)$ = Average direct trust of A on B.

Where $D(ATB) = \frac{\sum_{i=1}^{k} D(AT(c_i)B)}{m}$

Here k = number of available contexts for A.

m = number of contexts for which A has direct trust value for B.

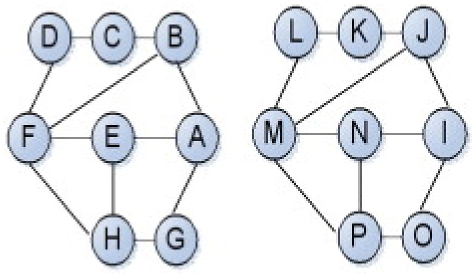


Fig. 15. A topology of nodes with trust relationships.

6.1.2. Recommended trust

Recommended trust is used in the absence of a direct trust value and is obtained when one node uses suggested trust values from the nodes with which it has direct trust. In Fig. 15, A might want to have a recommendation from B whether or not to serve C.

We devised a general equation for the calculation of recommended trust. Consider a node ωz that requests a context c_i from $\omega 1$. If $\omega 1$ (Service Provider, or SP) does not have a direct trust value for ωz (Service Requester, or SR), then it needs to know the recommended trust value to make the context sharing decision. Let us assume that there are n paths (p_1, p_2, p_3, … , p_i, … , p_n) with a hop length greater than 1 from $\omega 1$ to ωz

$$Tp_i = \left(\frac{T(\omega 1,\omega 2) + T(\omega 2,\omega 3) + \cdots + T(\omega x,\omega y) + T(\omega y,\omega z)}{\lambda}\right) \times \left(1 - \frac{(\lambda - 1)\times\psi}{10}\right) \quad (1)$$

where $\omega 1,\omega 2,\ldots,\omega x,\omega y,\omega z$ are the nodes on the path p_i from SP ($\omega 1$) to SR (ωz)

$$T(\omega x,\omega y) = \begin{cases} \mathrm{D}(\omega x T(ci)\omega y), \\ \text{where} D(\omega x T(ci)\omega y) \neq \phi \\ \omega x T \omega y, \text{otherwise} \end{cases}$$

λ = Hop distance between ω1 (SP) and ωz (SR),

Ψ = Distance based aging factor.

The recommended trust value of $\omega 1$ on ωk is calculated as:

$$R(\omega 1 T(c_i)\omega z) = \frac{\sum_{i=1}^{n} Tp_i}{n} \quad (2)$$

The term $(1 - \frac{(\lambda-1)\times\psi}{10})$ has been used as a weight factor to satisfy the 'distance based aging' property. Justification of ψ can be used from Ahamed et al.'s approach (Haque and Ahamed, 2007).

6.1.3. Active, passive and discrete recommendation

Active recommendations are possible only from neighboring nodes; passive recommendations may have the node consider every path that has a hop length $\geqslant 2$. Again when an SP node can't reach any path to consider it for recommendation, it needs some way to resolve the issue. That is what we term discrete recommendation. For the same context c_i it considers recommendations from other nodes that are in same discrete graph (Akers, 1978) relative to SR. In Fig. 15, if *A* needs a recommendation for *N*, the recommendation values for the paths {*M*, *N*}, {*I*, *N*}, and {*P*, *N*} are considered.

Here the equation takes the following form

$$Tp_i = \left(\frac{T(\omega i,\omega z)}{\lambda}\right) \times \psi = T(\omega i,\omega z) \times \psi [\because \lambda = 1, \text{Considering only 1 hop paths}]$$

Since, we are getting recommendations from nodes that are in no way connected to SP, we used ψ=0.5 which is a relatively lower weight factor.

6.1.4. Determination of optimal hop value

In the trust model, a device has the flexibility to define the maximum length of a recommendation path which is denoted as 'Initial Hop' value (IH). Here, we provide some guidelines for choosing the value of IH.

(1) If we consider a very small value for IH, the overall process for calculating the recommended trust will take less time. On the other hand, it will discard several possible recommendation paths with lengths greater than that specified in the IH. So, we need to consider a trade-off

between time and accuracy of the recommended trust value for specifying IH.

(2) The upper bound of an IH value is related to the weight factor.

$$IH \leq (\text{Initial Trust Value}) / (\text{Decrement rate of weighting factor per hop}) = \frac{0.5}{(1-(\lambda-1)\times\psi/10)-(1-((\lambda+1)-1)\times\psi/10)}$$

(3)

In this model (Haque and Ahamed, 2007) the decrement rate of the weight factor is 5% (95% of the recommendation value is counted for a recommendation path of length 2, 90% value is counted for a path of length 3, and so on). From the above equation, the maximum value for IH would be:

$$IH \leq \frac{0.5}{\left(1-(2-1)\times\frac{0.5}{10}\right)-\left(1-\left((2+1)-1\right)\times\frac{0.5}{10}\right)} = \frac{0.5}{0.05} = 10$$

The trust model will not consider any recommendation path with higher trust values in its intermediate links. However, a longer recommendation path that generates a poor recommendation value (less than or equal to 0.5), when it is multiplied by a small weight factor, will also be discarded by the model. So, it is not meaningful to consider such a high IH value. IH value beyond a specified limit will always generate an overall recommendation value of less than or equal to 0.5.

Let us consider a scenario where a 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, $Tp_i = \frac{(1+1+\cdots+1)}{10} \times (1 - \frac{(10-1)\times 0.5}{10}) = 0.55$

Now, consider a recommendation path 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 model. This indicates that any IH value greater than 10 will generate the same overall recommendation trust value which will be generated when IH = 10. But the increased IH value will certainly increase the computational time. So, the scenario supports the justification for the upper bound of IH discussed in Eq. (3).

6.2. Security management unit

The security management unit decides the mode of communication based on the situation and security level for a specific service. For a confidential service with high security level, only a neighbor with a high trust value could successfully acquire it.

This model is service and context specific. Here each device maintains a table of available services and corresponding security levels that range from 1 to 10. The security level is configured by the user. The Trust Manager of each device maintains a trust table indicating the current trust value of all the neighbors. SSRD performs automatic updates of trust values of neighbors. Based on the security level, once the service is granted, Unicast, Multicast, or Broadcast strategies are applied. For services with low security levels, no security mechanism is incorporated. But for higher security services, a public/private key mechanism has been adopted to ensure security. The trust model has been elaborated on (Sharmin et al., 2006a,b).

6.3. Algorithm and main data flow chart

The Resource Discovery model is implemented in a way similar to that of Device Discovery. Fig. 16 provides the interaction diagram of resource discovery.

Step 1: A device broadcasts its available service names to its neighbors.
Step 2: A service requester sends a service request to a service provider or broadcasts a service request to all service providers.

Step 3: The service provider retrieves the service security level based on the service type by calling the security manager unit.
Step 4: The service provider calculates the trust level of the service requester.
Step 5: The service provider decides whether or not to grant the service to the service requester based on the requester's trust level and service security level.
Step 6: If the service is granted, the service requester chooses one ultimate service provider from all available candidates.
Step 7: If the service is granted, the final service provider passes over the control to the ORB for service transfer.

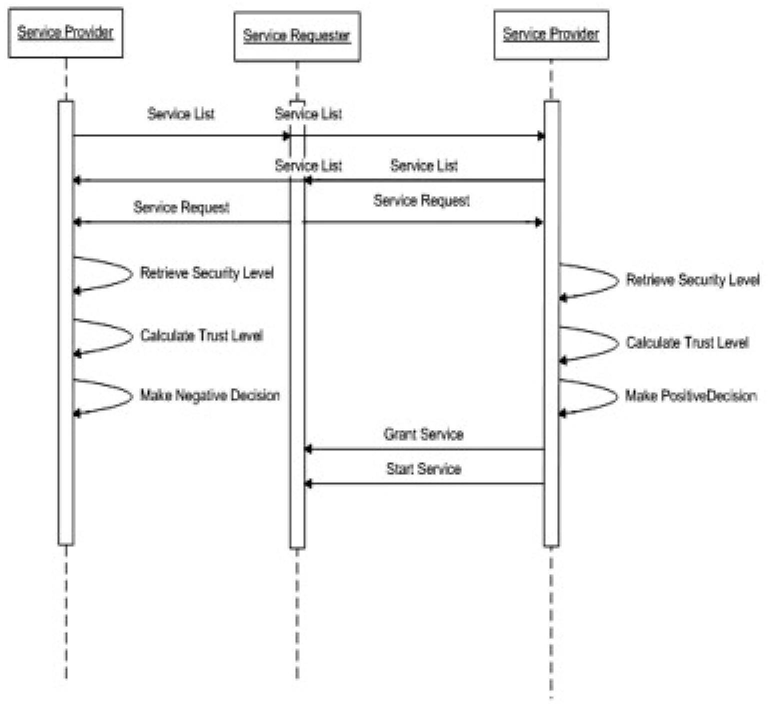


Fig. 16. Resource discovery interaction diagram.

In the implementation of SSRD, we define three messages which correspond to service request, service grant, and exit, respectively. Upon receiving a service request, a device would call both a security management unit and a trust management unit to determine whether to provide the service to the requester. Upon receiving a service-granting message, a device would set a required connection with the service provider, and then pass over the control to

ORB. Upon receiving an exit message, SSRD would quit. Fig. 17 shows the main data flow of SSRD.

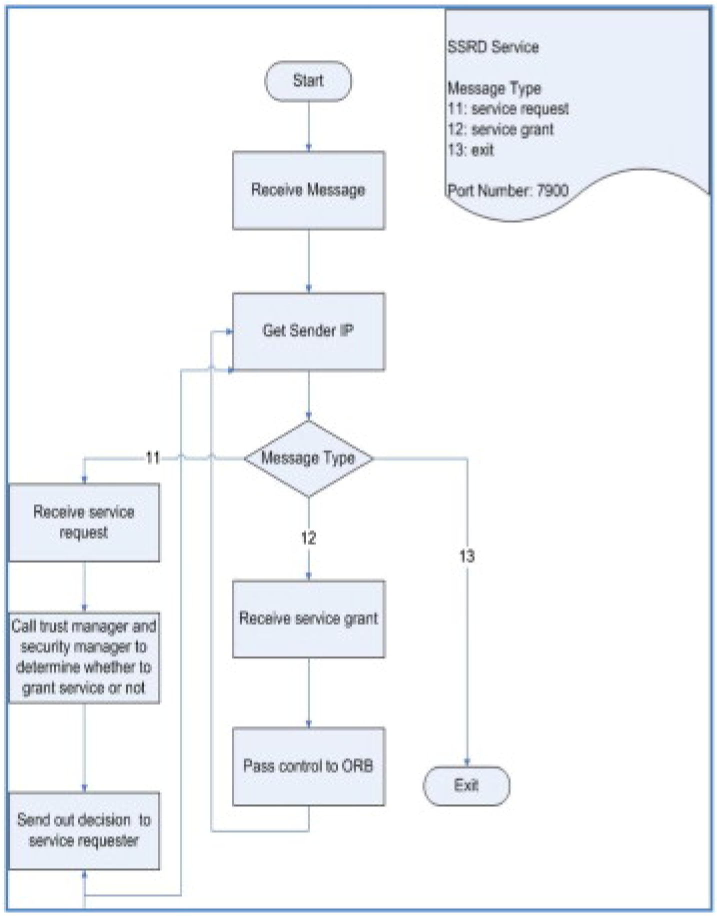


Fig. 17. Data flow chart of SSRD.

We divide the security into 3 levels. Any device could obtain a level 1 service without a trust level being checked. For level 2 service, only devices with trust levels greater than 0.4 could get the service. For level 3 service, which is highly confidential, the user has to respond to the request manually.

The trust value changes all the time reflexively and transitively. When calculating the trust value based on other devices' recommendations, the result is influenced by the user's interpretation. For example, if we are calculating a device A's trust level on device B, we know that A's trust level on C is 0.4, and A's trust level on D is 0.8. Given the same data, different users have different ways of calculating

A's trust level on device B according to their own interpretations. So in order to let S-MARKS run correctly, it is important for device users to agree on a common protocol for determining the trust level.

6.4. Service provider choosing strategy

As we have mentioned above, more than one service provider might respond to a service request with a positive answer. The service requester has to choose one from all these candidates. One strategy is to choose the one with strongest signal. The device with the strongest signal indicates a stable and constant connection between devices. The drawback is that the service requester has to wait a certain period of time before all positive answers come back. A signal strength checking program also occupies the computing power of the device. The other strategy is to take the service provider whose service-granting message arrives first as the ultimate service provider. The strategy is advantageous because it is simple and allows devices to act more quickly to acquire the service.

7. Privacy module

A privacy module is incorporated as an additional component in S-MARKS for supporting privacy-aware context-based applications. The privacy module takes the approach of the formal model to measure the quality of information and the level of anonymity of the users. The chief aim is to thwart any attempt by the attacker in re-identifying the user.

Fig. 20 shows how the privacy module works. The application obtains contextual and static profile information and shapes the list of information to be disclosed to the external entities. The privacy module consults the privacy preference of the user and then the quality of service requirement from the external entity (provided by the application object). The privacy module determines the amount and level of disclosure of the static and contextual information.

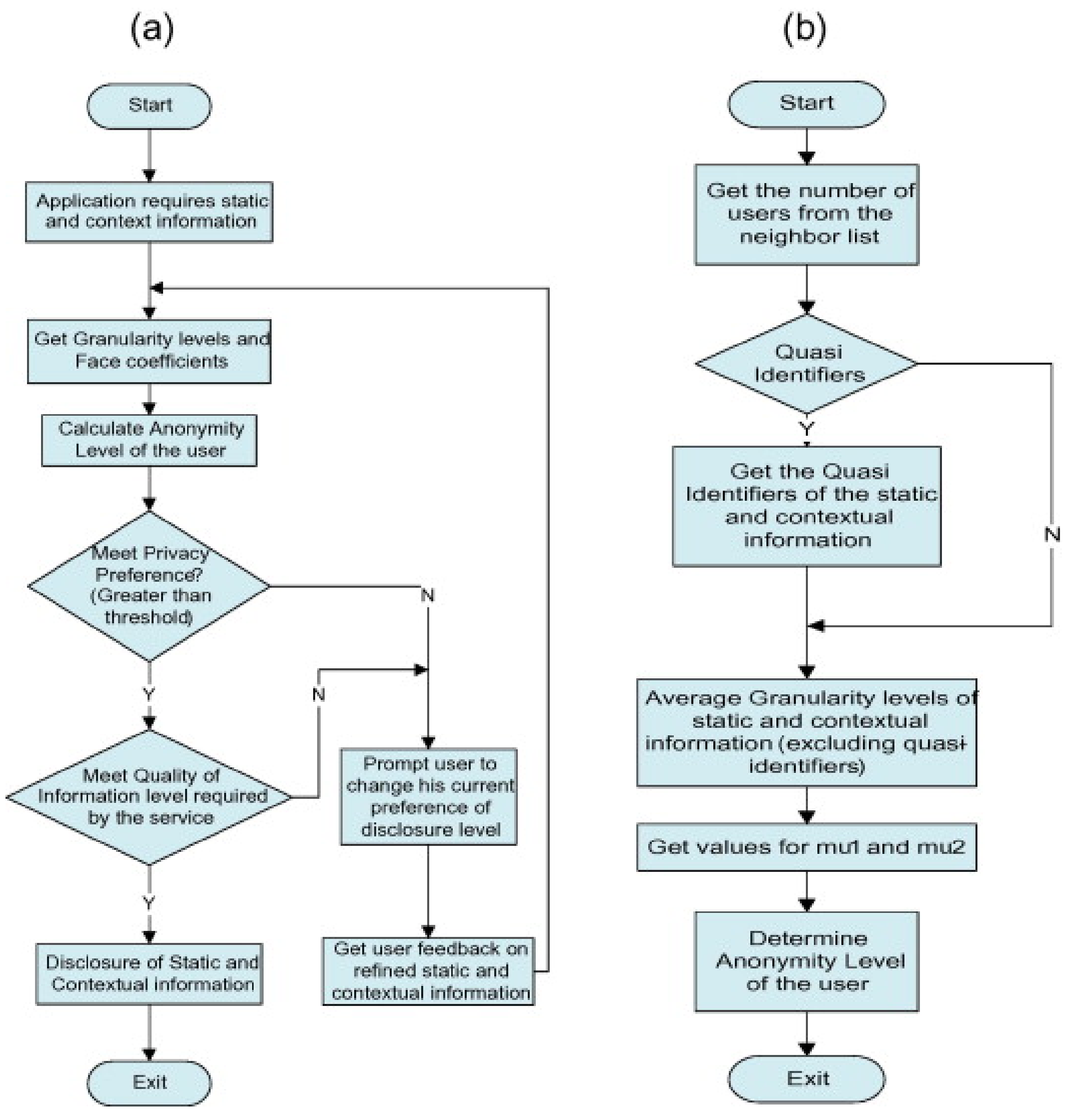

(a)
Start
Application requires static and context information
Get Granularity levels and Face coefficients
Calculate Anonymity Level of the user
Meet Privacy Preference? (Greater than threshold)
Y
N
Meet Quality of Information level required by the service
Y
N
Prompt user to change his current preference of disclosure level
Get user feedback on refined static and contextual information
Disclosure of Static and Contextual information
Exit
(b)
Start
Get the number of users from the neighbor list
Quasi Identifiers
Y
N
Get the Quasi Identifiers of the static and contextual information
Average Granularity levels of static and contextual information (excluding quasi identifiers)
Get values for mu1 and mu2
Determine Anonymity Level of the user
Exit

Fig. 20. Data flow chart for (a) privacy module (b) anonymity measure.

Some of the definitions are worth reviewing before we introduce the privacy module of S-MARKS.

Context: The context-service which is yet under development is aimed to provide contextual information to the users. The context (Dey, 2001) can be defined as the sensed information to describe some physical phenomena like temperature or location, nearby persons etc. of the user. We also use the term static information which can be referred to as user profile and hence not sensed from the environment. We refer to the context tuple of an entity throughout the paper as: $\langle t, l, c_1, c_2, c_3, \ldots, c_n \rangle$, where n no of contexts are always considered in spatio-temporal coordinates $\langle t, l \rangle$. As mentioned in the introduction section the contexts can take on different levels of granular values.

Faces for information disclosure: The 'face' is the concept of providing a pervasive application user the ability to control the disclosure of personal information, earlier proposed by Lederer et al. (2003). Since entities interact among themselves, it is imperative to restrict the amount of information shared. In an attempt to determine Privacy Preference Determinants (Lederer et al., 2003) four ordinal levels of precision have been presented. In this paper, we consider the *face coefficients* of the context or static information to be Boolean values. This suggests that a piece of contextual or static information will either be fully disclosed or not sent at all in the generalized request. The privacy level of the information to be disclosed is determined basically by the level of granularity of the information.

The quality of the request is measured by the *face coefficient* and *granularity level* of the information, provided no other information is known to the user (especially when the entities are in the purely pervasive environment). The measure of quality is required by the service provider to determine whether or not the request meets specific requirements.

Provided that all the contexts and static information have been considered in the same priority level (no quasi-identifiers are assumed (Sweeney, 2002), the normalized quality of information measure $=$

$$\left(\sum_{i=1}^{n} w_i c_{gi} + \sum_{j=1}^{m} v_j s_{gj}\right) / (m+n)$$ where

$c_{gi} = \frac{k_{ci}}{K_{ci}}$, $s_{gj} = \frac{k_{sj}}{K_{sj}}$ and $w_i, v_j \in \{0,1\}$, k_{ci}, k_{si} stands for the granularity level of the context and static information, while K_{ci} K_{si} stands for the maximum level for contextual information *ci* or static information *si*. w_i and v_j are the face coefficients. c_{gi} and s_{gj} are contextual and static granular attributes, respectively.

The information hierarchy proposed by Hengartner and Steenkiste (2006) can be used to granulate context. For example, the location information “Marquette University Cudahy Hall Room 301” can be split into three levels of granularity. The coarsest level of information “Marquette University” can be assigned value 1, whereas the finest information (the whole piece) can have 3. Likewise, we can arrange age information of an individual with an age group range rather than using the age itself. That provides us the flexibility of 2 granularity levels. The level of anonymity is used to quantify anonymity while disclosing information.

The architecture with trusted third party can effectively measure the *k*-anonymity level based on the quasi-identifiers (Sweeney, 2002) of the information since it has access to all the requests made at that time. But a purely pervasive environment can’t leverage such an advantage for obvious reasons. Hence, we have proposed three different approaches to measure the anonymity; they consider with or without prior information of quasi-identifiers.

7.1. Quasi-identifiers

The quasi-identifiers play a pivotal role in the re-identification of the individual considering the attacker has complete access to the contextual and static information. In very naïve terms, the quasi-identifiers are those piece(s) of information (one or more grouped together) that can distinguish records among users effectively attributing to one person. The *k*-anonymity model aims to make the information indistinguishable among *k* persons.

Let us examine a set of data with quasi-identifiers {age, gender, location}. The maximum and disclosed granularity levels of the information here are: {age, gender, location} = { ⟨3, 3⟩ , ⟨1, 1⟩ , ⟨3, 3⟩ }. Note, the example considers both context (location) and static (age, gender) information.

It is evident from Table 3 that each tuple can be attributed to different persons. Rearranging the granularity level of the contextual/static information we get:

Table 3. Data with distinct tuples for each person.

Age	Gender	Location
23 y 3 m	F	MU Cudahy Hall, #301
23 y 1 m	F	MU Cudahy Hall, #301

To preserve the quality of information and to prevent a re-identification attack, the revised granularity set will be { 〈3, 1〉 , 〈1, 1〉 , 〈3, 2〉 }. The quality of information with regards to the quasi-identifiers is:

$$Q\_Quasi = (1/3 + 1/1 + 2/3)/3 = 0.66$$

And the k-anonymity level achieved is 2.

But in the purely pervasive environment, there's no way to know the number of potential users for the service and others' contextual and static information. So, there is no effective way to determine the k-anonymity level of the user. Therefore, in our approach to determine the anonymity level of the user, we considered the impact of the known quasi-identifier set and the number of neighbors.

7.2. Level of Anonymity

The privacy module in S-MARKS considers three different approaches to measure the anonymity. The first two consider with or without prior information of quasi-identifiers, whereas the last approach demonstrates the impact of pre-determined quasi-identifiers.

7.2.1. Anonymity measure with no prior information of quasi-identifiers

We have considered the anonymity measure as an opposite measure of quality of information in the scale [0, 1], since the measure of quality is already normalized to that range. The reason is pretty clear. As we go on increasing the quality of the information, the user has to compromise all of his context and static information with achieving the least generalization. Therefore, *Anonymity measure* = 1 − *Quality of Information measure*.

7.2.2. Anonymity measure with prior assumptions on quasi-identifiers

Apparently, a good measure of anonymity can be obtained with prior knowledge of the environment. We made a number of assumptions before devising such a measure, such as:

1. The number of users in the network is known.
2. The only impact the presumed quasi-identifiers will have is the average granularity level of the static and contextual information.
3. The number of contextual and static information revealed with their average granularity level will have a relatively lower impact.

Hence, the measure of anonymity is defined as,

$$Anonymitymeasure = 1 - e^{-(n-1)^2/2\sigma^2} \text{ and},$$
$$\sigma = \mu_1/(N_{cs}k_1) + \mu_2/k_2, \mu_2 > \mu_1$$

where *n* = Average number of users in the system.

N_{cs} = Number of context and static information revealed to the service provider except the quasi-identifiers k_1 = Average granularity levels of the context and static information except quasi-identifiers in the requests.

k_2 = Average granularity levels of the quasi-identifiers in the requests.

The equation presented above is worth an explanation. The anonymity measure has the range [0, 1]. We can observe from the pattern of the curves in Fig. 18, as the number of users in the network increases, the anonymity measure increases to the maximum at some point governed by the parameter σ. σ is determined based on the number of the contexts and static information and the granularity levels of the same.

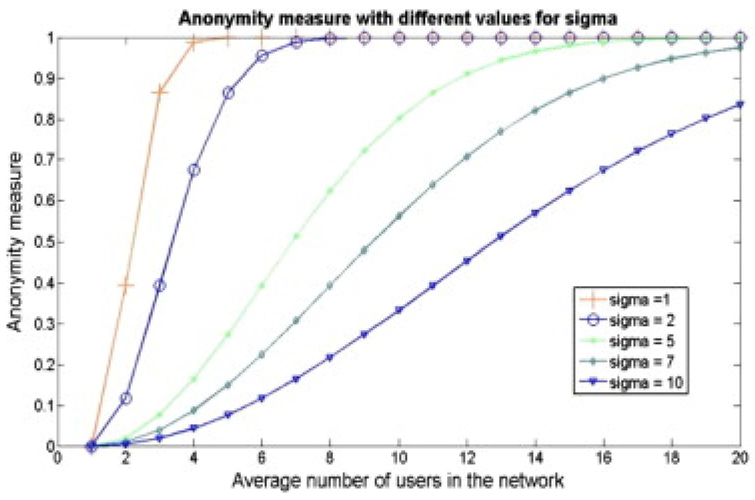


Fig. 18. Anonymity measures with different σ choosing $\mu 1$ and $\mu 2$.

The most careful consideration is required while choosing the parameters $\mu 1$ and $\mu 2$, because they govern the value of σ or in other words the flatness of the curve. The reason why $\mu 2$ is chosen larger than $\mu 1$ is that the average granularity level of the quasi-identifiers have been considered to have significant contribution in shaping the anonymity curve. The larger the value of σ, the slower will be the rise of the measure of the anonymity. It might be highly desirable for a system to achieve higher anonymity levels even with a fewer number of users. A good rule of thumb will be to use higher values of μ_1 and μ_2 with higher values of revealed context and static information to observe a gradual rise of the anonymity measure with respect to increasing number of users in the network. This can also be seen from Table 5. Still, it differs greatly with applicative contexts.

Table 5. Different values of σ in different scenario.

s/n	l_1	l_2	$^N cs$	k_1	k_2	r
1	10.0	30.0	10	2.5	1.5	20.4000
2			50	1.3	1.0	30.1538
3	15.0	25.0	10	2.5	1.5	17.2667
4			50	1.3	1.0	25.2308

Table 6 demonstrates the choices of and for a situation where the non-quasi-identifiers are completely disclosed (k_2 = 1.0). In Table 4, the number and the average granularity level of the quasi-identifiers are considered to have values 4 and {2.0, 3.0}, respectively. The gradual rise of the anonymity measure can be controlled by the value of σ, as evident from the table. For a network, where there is a high risk of malicious users to be found, the higher values of σ are chosen so that the malicious users have less impact on the anonymity measure. In other words, higher value of is set so that it ensures the privacy of the users' information even with lower quality of information and higher number of users. From the Fig. 19, it is evident that the higher values of $\mu 2$ constitute to higher values of σ and eventually result in flatter curves. This is how the anonymity measure responds to the network with a high number of malicious users.

Table 6. Different values of σ with non-quasi-identifiers completely revealed.

s/n	l_1	^{N}cs	l_2	k_1	k_2	r
1	5.0	4	5.0	2.0	1.0	5.625
2			15.0	3.0		15.625
3	20.0		5.0	2.0		6.6667
4			15.0	3.0		16.667

Table 4. Preserving *k*-anonymity, *k* = 2.

Age (group)	**Gender**	**Location**
⩾20 and <24	F	MU Cudahy Hall,
⩾20 and <24	F	MU Cudahy Hall,

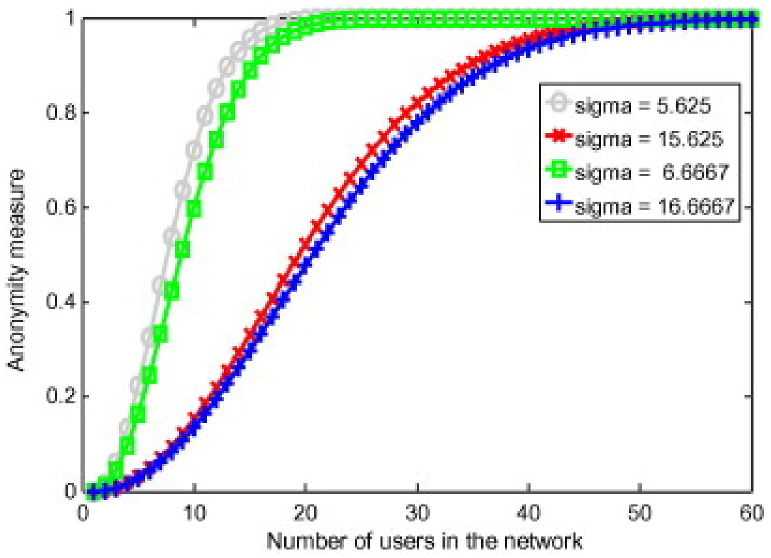


Fig. 19. Anonymity measures with different σ, where non-quasi-identifiers are completely revealed.

7.3. Safe and unsafe request threshold

The anonymity measure is used by the generalization function to see whether it meets the predefined threshold. Anonymity $\leq h \in [0,1]$, where, *h* is considered the safety threshold of the request. Anything above *h* will be considered unsafe for the requester. The requester may be asked to discard the request or carry on, knowingly disregarding his own preference.

7.4. Flow diagram

The Privacy Module flow diagram is shown in the Fig. 20. The first part shows how, with the help of Privacy Module, the application determines the optimal disclosure levels (granularity and disclosure) of contextual and static information comparing the anonymity measure with the user preference and the quality of information with the service requirement. The second part demonstrates how the anonymity measure is determined with the help of the parameters μ_1, μ_2 of the equation shown in the Section 7.2.2. The number of users is considered to be the number of valid neighbors in our case. We are working on this scalable module which is intended for supporting context-aware applications that share services in the exchange of information and ensures privacy of the shared information.

7.5. Utility of the anonymity measure

Different network anonymity quantification techniques (T´oth et al., 2004) found in the literature are largely influenced by the 'Shannon's information theory' (Shannon, 1948). All the approaches consider the initial assumptions on the probability of senders and recipients' anonymity sets. The basic equation for entropy of information (sometimes denoted as unlinkability) has the form: $H(X) = \sum_{i=1}^{N} p_i \log_2(p_i)$, where *N* stands for the number of the users, and p_i stands for the probability of individual users in the anonymity set. Now, the maximum entropy will be determined by, $H(M) = \log_2 N$.

Hence the anonymity measure is, $d = H(X)/H(M)$.

If all the users are considered equi-probable as the sender of next message in the network, the anonymity measure gains the highest value 1. However, the difficulty in comparing this measure with ours is that the former is constrained to considering anonymity of only single message. Therefore, the measure is not affected by the increasing number of users in the network.

On the contrary, our anonymity measure is based on the fact that the order of the messages is not an important issue as long as the users exchange messages for a small duration of time. The proposed measure is primarily affected by the increasing number of users in the network. The anonymity will be higher when there is larger number of users in the network with potential to exchange information in the network. See the difference between two anonymity measures in Fig. 21.

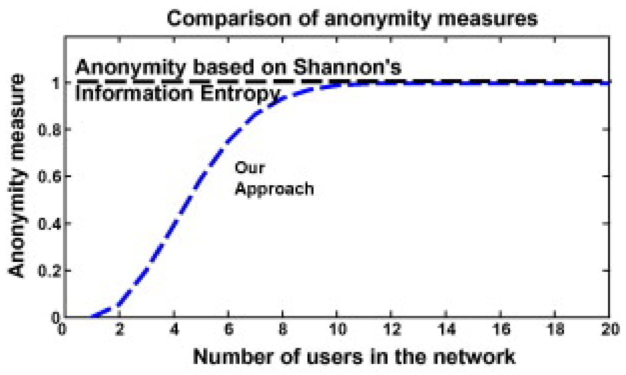


Fig. 21. Comparison of anonymity measures with the increase of users in the network considering equal probabilities of senders.

In an attempt to compare these two approaches, we proposed a common parameter. We considered unequal probability of the users of being the sender of the message and tried to map it to the number of users having the potential to send a message in our approach. Let us consider the following example: where a network consists of 10 users (N = 10) and the probabilities of individual users, being the sender of a request is arbitrarily chosen as:

$$P_i = p/4, \; 1 \le i \le 4, \; P_i = (1-p)/6, \; 5 \le i \le 10$$

Apparently, in Shannon's method, the anonymity measure, d will reach the highest (1) when p = 0.4. Our approach doesn't consider prior probability, but uses the fact that changes in probability distribution will have impact on the cardinality of the anonymity set. Thus, we calculate the number of users in the network, with prior probability distribution through this formula:

$$n = N \times (1 - sd(P)),$$

where *sd* stands for standard deviation of the probability distribution of the users with different *p* values. We obtained the *n* values for *p* = {0.1:0.1:10} as {9.3545, 9.5697, 9.7848, 10.0000, 9.7848, 9.5697, 9.3545, 9.1393, 8.9242, 8.7090}. For *p* = 0.4, our method also obtained the highest anonymity measure and it can be regulated through different sigma values as shown in the Fig. 6. Furthermore, the number of users doesn't undergo significant change. In other words, the measure is rather flatter throughout the distribution. Clearly, it requires a larger deviation in the distribution to significantly affect the anonymity measure. Fig. 22 shows the anonymity levels measured by the 'Shanon's approach' and our proposed approach with *σ* values 3, 5 and 10.

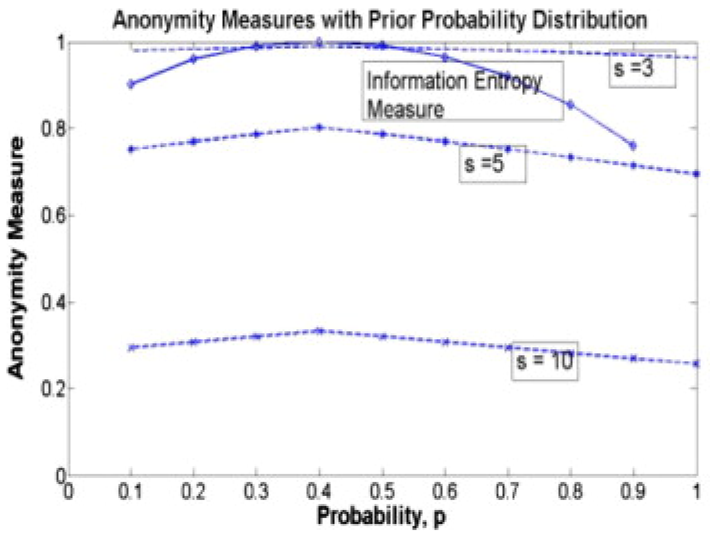
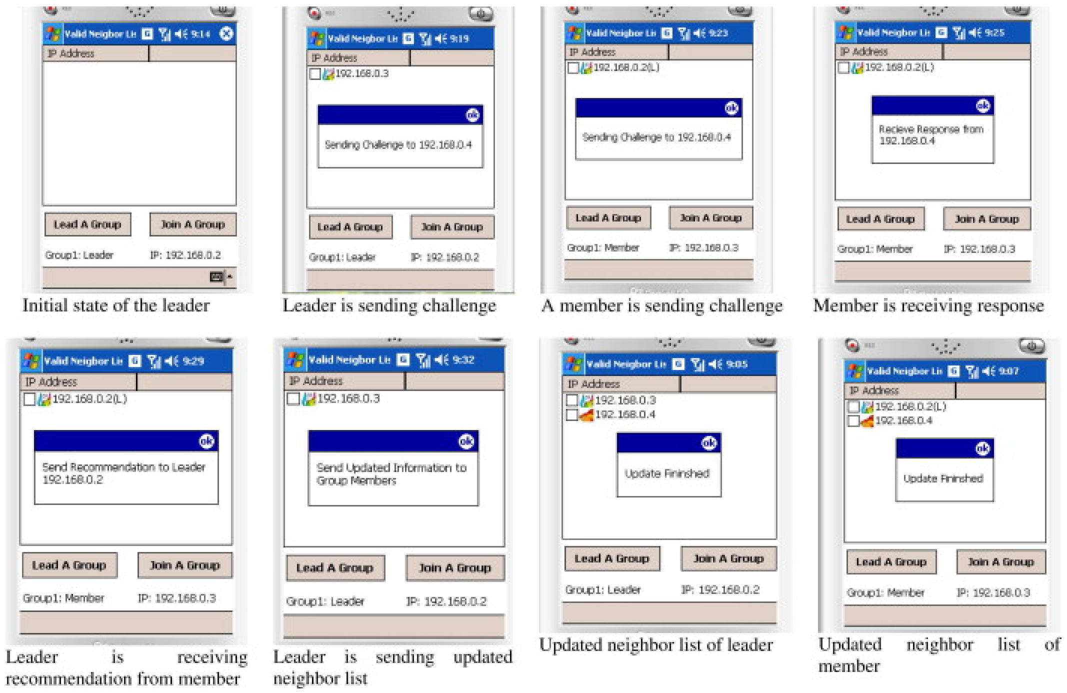


Fig. 22. Translation of anonymity measures with prior probability distribution with message sender.

8. Prototype of S-MARKS

Fig. 23 gives a detailed description of how ILDD in S-MARKS works. The device shows a list of users with their IP addresses. This device also sets up a group which takes the device as a leader device. (Under this situation, there is only one member in the whole group -- the leader device.) Our description then shows that there are two devices in the group. The leader device is sending a challenge to the newcomer "192.168.0.4". Then a member device is sending a challenge to the newcomer. The next three figures show how responses and recommendations are transferred between devices. The last two indicate the update is finished.



Initial state of the leader

192.168.0.3
Sending Challenge to 192.168.0.4
Group1: Leader IP: 192.168.0.2

Leader is sending challenge

192.168.0.2(L)
Sending Challenge to 192.168.0.4
Group1: Member IP: 192.168.0.3

A member is sending challenge

192.168.0.2(L)
Recieve Response from 192.168.0.4
Group1: Member IP: 192.168.0.3

Member is receiving response

192.168.0.2(L)
Send Recommendation to Leader 192.168.0.2
Group1: Member IP: 192.168.0.3

Leader is receiving recommendation from member

192.168.0.3
Send Updated Information to Group Members
Group1: Leader IP: 192.168.0.2

Leader is sending updated neighbor list

192.168.0.3
192.168.0.4
Update Finished
Group1: Leader IP: 192.168.0.2

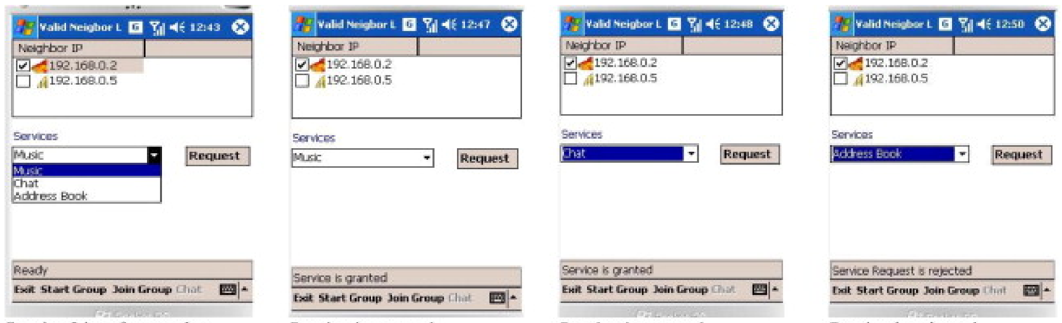
Updated neighbor list of leader

192.168.0.2(L)
192.168.0.4
Update Finished
Group1: Member IP: 192.168.0.3

Updated neighbor list of member

Fig. 23. Screenshots of ILDD implemented in S-MARKS.

The screenshots in Fig. 24 illustrate how SSRD works after members have passed the authentication phase. The neighbor device 192.168.0.2 has three services: Music, Chat and Address book. Music service has security level 1; Chat service has security level 2; Address Book includes the confidential information which has the highest security level, 3. The device asks for both the music and chat service, and both services are granted. But when the device asks for the address book service, the request is rejected by the user.



Ready

Service is granted

Service is granted

Service Request is rejected

Service List of a member Service is granted Service is granted Service is rejected

Fig. 24. Screenshots of SSRD in S-MARKS.

9. Illustrative example revisited – S-MARKS in campus ad hoc network

Let us review the scenario we illustrated in the Section 2. Now we will be able to understand what part of the service people were using and the roles they were playing. S-MARKS could be used as the middleware service to serve all the purposes in the scenario. First of all, Alice was already a valid member of the ad hoc network mentioned. She was also playing the role of a service provider, as is Earl at the later stage. Bob, Carl and David came as new members to join the network. The authentication was performed by ILDD challenge response. Even though Bob and Carl were authenticated, David had previously been revoked of all privileges. He was trying to regenerate the key and provide a perfect response to all the group members. But he was denied because he didn't know the noise (refer to Section 5) in the responses which is shared only among the valid group members. Again, Carl was added to the valid neighbor list of everyone in the group, but he was not privileged enough to use some of the services with lower trust levels. He could start using some other service and achieve the required trust level to be able to access the desired services.

The SSRD and trust management group provide the necessary support for secured access to services by maintaining the trust levels for devices with service specific contexts. Finally, Earl introduced a new service that requires some static and contextual information to get access. But not everybody is willing to compromise his own privacy. The task of the privacy module in S-MARKS is to provide support to applications for exchanging information that meets the mutual preference of the service requester and the provider. Fig. 25 provides a quick overview of the situation where everyone is using S-MARKS in their handheld devices.

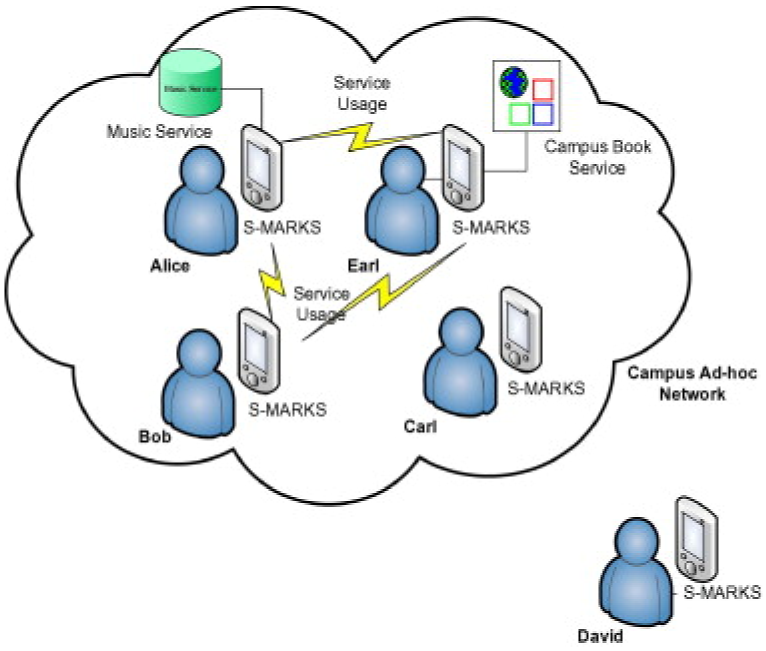


Fig. 25. Campus ad hoc network.

10. Future work

There are three unresolved issues in S-MARKS. The first is how to choose the leader device. Theoretically, the leader device should be chosen based on battery power and trust level, as mentioned above. In reality, batteries in different devices have different maximum levels. Even for the same type of devices, we cannot guarantee that their batteries provide the same performance. So it is hard to find reliable criteria to determine which device has the highest power level. A power level-checking and comparison module also adds a burden to the device's computation power. ILDD is based on the presumption that the leader device would never leave the group. A problem would arise when the leader device leaves the group since no other device would take control of the whole group. One solution is that before leaving, the leader device would pass over control to one of the group members. Another problem emerging here is how the leader device determines that it is, itself, leaving the group and not all group members leaving the group.

The second problem is determining appropriate time length to collect all recommendations for updating the neighbor list. The leader device in the established secure group waits a certain period of time for members to send back their recommendations. If the time length is too short, the leader device will have finished updating the neighbor list before all member devices send out recommendations. In such situation, some devices, which otherwise would be authenticated, might be treated as malicious devices because some positive recommendations are not received. If the time length is too long, the neighbor list may not reflect the current valid neighbors.

The third problem is the limitation of the pre-configuration. Every device which incorporates S-MARKS must be configured into the same local network in advance. In other word, every device could only communicate with devices in the same network section. It is neither user-friendly nor convenient for users to find out what the current network ID is and its corresponding network mask. It raises conflicts if a user configures a device with an already used IP address. The ideal scenario would be for devices to merge into a pervasive environment without inconveniencing the device user. Our next goal is to build a protocol which would be used to set up a network built purely on devices' hardware addresses.

11. Existing middleware solutions

Researchers are involved in middleware design (Capra et al., 2001; Sharmin et al., 2006a,b; Campbell et al., 1997; Dertouzos et al,. 1999; Wyckoff et al., 1998; Sousa et al., 2002; Murphy et al., 2001; Mascolo et al., 2002; Cerqueira et al., 2001; Yau et al., 2002; Kumar et al., 2003; Eichberg et al., 2004), for portable devices running in a pervasive computing environment. But they are not yet close to providing an optimum solution that is secure by design.

In Reflective middleware (Capra et al., 2001), the concept of user profile was introduced but not utilized to its full capacity. A simple yet powerful algorithm, to unearth available resources, has not yet been devised. Most of the approaches follow the resource announcement policy. Reconfigurable Context-Sensitive Middleware (RCSM+) (Yau et al., 2002) mainly deals with situation-awareness, ephemeral group management, and autonomous coordination for information dissemination. Gaia (Cerqueira et al., 2001) tries to solve the problem of ubiquitous computing by introducing a general operating system middleware, which exports and coordinates the resources contained in a physical space. They introduce the idea of active space that converts a physical space and its ubiquitous computing devices into a programmable computing system. Gaia's activities, however, are confined only within the active space. MIT's Oxygen project (Dertouzos, 1999) turns the inactive environment into an empowered one to facilitate the users. This project focuses on new adaptive mobile devices, new embedded distributed computing devices, intelligent knowledge access technology, automation technology, etc. At present, systems can divert a user in many explicit and implicit ways, which may reduce his/her effectiveness. Project Aura (Sousa and Garlan, 2002) rethinks system design to address this problem. Aura tries to provide each user computing and information services at every level regardless of location. Gaia, Oxygen, and Aura show splendid performance inside the smart space. But their focus is different, as they try to accommodate all the facilities they mentioned inside a particular smart space. Consequently, these are not the ultimate solution for mobile devices running in a pervasive computing environment.

Other noteworthy middleware for mobile devices include MARKS (Sharmin et al., 2006a,b), Mobiware (Campbell, 1997), TSpaces (Wyckoff et al., 1998), LIME (Murphy et al., 2001), XMIDDLE (Mascolo et al., 2002), PICO (Kumar et al., 2003) and ALICE (Eichberg, and Mezini, 2004). MARKS supports knowledge usability, resource discovery and self-healing aspects in the pervasive computing environment. LIME, supporting scarce context-awareness and inadequate ad hoc communication, is the result of a development process assimilating formal modeling integration and application development. TSpaces provides a common platform to facilitate the linkage of all systems and application services. Server software containing data are stored on fixed and powerful machines; this is inappropriate in an ad hoc communication environment. Xmiddle uses a tree-structure for storing data. Here, the unit of replication can be adjusted to accommodate both device and application needs. It is appropriate for mobile computing since it targets ad hoc networks. However, it is implemented using Extensible Markup Language (XML) that increases the communication overhead.

The middleware designs discussed above, as well as other existing ones, did not provide security solutions from the perspective of device validation, resource discovery providing trust, handling malicious recommendations, and avoiding privacy violation. S-MARKS provides these features in a middleware. Table 7 demonstrates the comparative study of major middleware.

Table 7. Comparison of middleware features.

Middleware	**Context and situation awareness support**	**Infrastructure support necessary**	**Ad hoc communication support**	**Authentication based Device Discovery**	**Resource Discovery**	**Trust Management with Malicious user detection**	**Privacy support**
RCSM	Yes	No	Yes	No	No	No	No
Gaia	Yes	Yes	No	No	Yes	No	No
Oxygen	Yes	Yes	No	No	Yes	No	No
LIME	Yes	Yes	No	No	No	No	No
S-MARKS	Yes	Yes	Yes	Yes	Yes	Yes	Yes

12. Conclusion

In this paper, we have addressed device validation and resource discovery providing a trust model and an additional privacy module for context-based applications. Both of these are related to security concerns in the pervasive computing environment. We have incorporated them in S-MARKS, a middleware that is secure by design. In our first prototype, we fully implemented the device validation and resource discovery. In the future, we will design and develop a means for handling malicious recommendations and privacy violations while sharing services, and incorporate this within the current implementation. Our extensible framework will incorporate a context-service and malicious recommendation handler to serve trust management in the near future by providing more secured communication among the devices. It will be designed to be more adaptive for real-time systems.

S-MARKS can have an influential effect on people's lives through addressing security concerns regarding services sharing among portable devices. This may encourage people to utilize portable devices on a larger scale.

Acknowledgements

This project is partially supported by Way-Klinger Young Scholars Award and Wehr grant. Microsoft research has donated some equipment for the experiments.

References

Abramowitz and Stegun, 1972. M. Abramowitz, I.A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, New York (1972). See Section 26.7

Akers, 1978. S.B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27 (6) (1978), pp. 509-516

Ahamed et al., 2006. S.I. Ahamed, M. Haque, K.M. Asif. S-MARKS: a middleware secure by design for the pervasive computing environment. *Proceedings of the Fourth International Conference on Information Technology: New Generations (ITNG 2006),* IEEE CS Press, Las Vegas, Nevada, USA (2006), pp. 303-308

Bellotti and Sellen, 1993. Bellotti, V., Sellen, A., 1993. Design for privacy in ubiquitous computing environments. In: *Proceedings of third European Conference on Computer Supported Cooperative Work (ECSCW 93),* pp. 77–92.

Beresford and Stajano, 2003. A.R. Beresford, F. Stajano. Location privacy in pervasive computing. *Pervasive Computing*, 2 (1) (2003), pp. 46-55

Berlekamp et al., 1978. E.R. Berlekamp, R.J. McEliece, V. Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory* (1978), pp. 384-386

Blum et al., 2003. A. Blum, A. Kalai, H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50 (2003), pp. 506-519

Bryant, 1986. R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35 (8) (1986), pp. 677-691

Campbell, 1997. Campbell, A.T., 1997. Mobiware: QOS-aware middleware for mobile multimedia communications. In: *Proceedings of the IFIP TC6 Seventh International Conference on High Performance Networking VII*, White Plains, New York, United States, pp. 166–183.

Campbell et al., 2002. Campbell, R., Al-Muhtadi, J., Naldurg, P., Sampemane1, G., Mickunas, M.D., 2002. Towards security and privacy for pervasive computing. In: *Proceedings of International Symposium on Software Security*, Tokyo, Japan.

Capra et al., 2001. Capra, L., Emmerich, W., Mascolo, C., 2001. Reflective middleware solutions for context-aware applications. In: *Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns*, September 25–28, 2001.

Cerqueira et al., 2001 Cerqueira, R., Hess, C.K., Roman, M., Campbell, R.H., 2001. Gaia: a development infrastructure for active spaces. In: *Workshop on Application Models and Programming Tools for Ubiquitous Computing* (held in conjunction with the UBICOMP 2001).

Dertouzos, 1999. M. Dertouzos. The oxygen project. *Scientific American*, 281 (2) (1999), pp. 52-63

Dey, 2001. A.K. Dey. Understanding and using context. *Personal and Ubiquitous Computing* (2001), pp. 4-7

Eichberg and Mezini, 2004. Eichberg, M., Mezini, M., 2004. Alice: Modularization of Middleware using Aspect-Oriented Programming. In: *Software Engineering and Middleware (SEM 2004),* Linz, Austria, pp. 47–63.

Gedik and Liu, 2005. Gedik, B., Liu, L., 2005. Location-privacy in mobile systems: a personalized anonymization model. In: *Proc. of ICDCS*, pp. 620–629.

Ghinita et al., 2007. Ghinita, G., Kalnis, P., Skiadopoulos, S., 2007. PRIVÉ: anonymous location-based queries in distributed mobile systems. In: *Proc of WWW* 2007, pp. 371–380.

Gossett, 1908. W.S. Gossett. The probable error of a mean. *Biometrika*, 6 (1) (1908), pp. 1-25

Haque and Ahamed, 2007. M. Haque, S.I. Ahamed. An omnipresent formal trust model (FTM) for pervasive computing environment. *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007),* IEEE CS Press, Beijing, China (2007), pp. 49-56

Haque and Ahamed, 2008. M. Haque, S.I. Ahamed. An impregnable lightweight device discovery (ILDD) model for the pervasive computing environment of enterprise applications. *IEEE Transactions on Systems*, Man, and Cybernetics, Part C, 38 (3) (2008), pp. 334-346

Hengartner and Steenkiste, 2006. Hengartner, U., Steenkiste, P., 2006. Avoiding privacy violations by context-sensitive services. In: *Proc. of Percom*, pp. 222–233.

Hopper and Blum, 2000. Hopper, N., Blum, M., 2000. A secure human computer authentication scheme. In: *Technical Report CMU-CS-00-139*, Carnegie Mellon University.

Hopper and Blum, 2001. N.J. Hopper, M. Blum. Secure human identification protocols. *Advances in Cryptology – ASIACRYPT*, 2248 (2001), pp. 52-66

Juels et al., 2005. Juels, A., Stephen, A., Weis, 2005. Authenticating pervasive devices with human protocols. In: *Advances in Cryptology – CRYPTO 2005*, Springer-Verlag, pp. 293–308.

Kagal et al., 2001. L. Kagal, T. Finin, A. Joshi. Trust-based security in pervasive computing environments. *Computer*, 34 (12) (2001), pp. 154-157

Kindberg and Fox, 2002. T. Kindberg, A. Fox. System software for ubiquitous computing. *IEEE Pervasive Computing* (2002), pp. 70-81

Kumar et al., 2003. M. Kumar, B.A. Shirazi, S.K. Das, B.Y. Sung, D. Levine, M. Singhal. PICO: a middleware framework for pervasive computing. *Pervasive Computing*, 2 (3) (2003), pp. 72-79

Langheinrich, 2001. Langheinrich, M., 2001. Privacy by design – principles of privacy-aware ubiquitous systems. In: *Proceedings of the 3rd International Conference on Ubiquitous Computing, Atlanta, Georgia, USA.*

Lederer et al., 2003. Lederer, S., Mankoff, J., Dey, A., 2003. Who wants to know what when? Privacy preference determinants in ubiquitous computing. In: *Proc. of CHI 2003*, pp. 724–725.

Mascolo et al., 2002. C. Mascolo, L. Capra, S. Zachariadis, W. Emmerich. XMIDDLE: a data-sharing middleware for mobile computing. *Journal of Wireless Personal Communications*, 21 (1) (2002), pp. 77-103

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

Mokbel et al., 2006. Mokbel, M.K., Chow, C., Aref, W.G., 2006. The new casper: query processing for location services without compromising privacy. In: *Proc. of VLDB*, pp. 763–774.

Munilla and Peinado, 2007. J. Munilla, A. Peinado. HB-MP: a further step in the HB-family of lightweight authentication protocols. *The International Journal of Computer and Telecommunications Networking*, 51 (9) (2007), pp. 2262-2267

Murphy et al., 2001. Murphy, A.L., Picco, G.P., Roman, G.C., 2001. Lime: a middleware for physical and logical mobility. In: *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, pp. 524–533.

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.*

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.

Satyanarayanan, 2001. M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Wireless Communications*, 8 (4) (2001), pp. 10-17

Shannon, 1948. C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27 (1948), pp. 379-423

Sharmin et al., 2005. Sharmin, M., Ahmed, S., Ahamed, S.I., 2005. SAFE-RD (secure, adaptive, fault tolerant, and efficient resource discovery) in pervasive computing environments. In: *International Conference on Information Technology: Coding and Computing (ITCC'05),* vol. II, pp. 271–276.

Sharmin et al., 2006a. Sharmin, M., Ahmed, S., Ahamed, S.I., 2006. 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., 2006. 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.

Sousa and Garlan, 2002. Sousa, J.P., Garlan, D., 2002. Aura: an architectural framework for user mobility in ubiquitous computing environments. In: *Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture*, pp. 29–43.

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

Stajano, 2002. F. Stajano. *Security for Ubiquitous Computing*. Wiley (2002). pp. 110–111

Sweeney, 2002. L. Sweeney. K-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5) (2002), pp. 557-570

Talukder and Ahamed, 2008. N. Talukder, S.I. Ahamed. FPCS: formal privacy-aware context-based service. *Proceedings of the 32nd Annual International Computer Software and Applications Conference (COMPSAC 2008), IEEE CS Press, Turku, Finland (2008),* pp. 432-439

Toth et al., 2004. Toth, G., Horn´ak, Z., Vajda, F., 2004. Measuring anonymity revisited. In: *Proc. of Nordic Workshop on Secure IT Systems,* pp. 85–90.

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

Wyckoff et al., 1998. P. Wyckoff, S.W. McLaughry, T.J. Lehman, D.A. Ford. T Spaces. *IBM Systems Journal*, 37 (3) (1998), pp. 454-474

Yau et al., 2004. Yau, S.S., Huang, D., Gong H., Seth, S., 2004. Development and runtime support for situation-aware application software in ubiquitous computing environments. In: *Proc. of 28th Annual International Computer Software and Application Conference (COMPSAC 2004),* Hong Kong, pp. 452–457.

Yau and Karim, 2004. S.S. Yau, F. Karim. An adaptive middleware for context-sensitive communications for real-time applications in ubiquitous computing environments. *Real-Time Systems*, 26 (1) (2004), pp. 29-61

Yau et al., 2002. Yau, S.S., Yu, W., Karim, F., 2002. Development of situation-aware application software for ubiquitous computing environments. In: *Computer Software and Applications Conference (COMPSAC 2002),* pp. 233–238.

Yau and Zhang, 2003. Yau, S.S., Zhang, X., 2003. A middleware service for secure group communication in mobile ad-hoc networks. In: *Proc. 27th Annual International Computer Software and Application Conference (COMPSAC 2003),* Dallas, Texas, pp. 10–15.

Zultner, 1999. R.E. Zultner. What do our metrics mean? *Cutter IT Journal*, 12 (4) (1999), pp. 11-19

About the Authors

Sheikh I. Ahamed received the B.Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 1995, and the Ph.D. degree in computer science from Arizona State University, Tempe, in 2003. He is currently an Associate Professor in the Department of Mathematics, Statistics, and Computer Science, and Director of the Ubicomp Research Laboratory, Marquette University, Milwaukee, WI. His current research interests include security in *ad hoc* networks, middleware for biquitous/pervasive computing, sensor networks, and component-based software development. He is a member of the IEEE Computer Society and the Association for Computing Machinery (ACM). He is a program committee member for several international conferences in software engineering and pervasive computing, such as the Computer Software and Applications Conference (COMPSAC 2009) and the Pervasive Computing and Communications Conference (PerCom 2008), and the

Symposium on Applied Computing (SAC 2008). He is the Program Co-Chair of the International Workshop on Security, Privacy, and Trust for Software Applications (SPTSA 2009). He is the Workshop Chair of Computer Software and Applications Conference (COMPSAC 2010).

Haifeng Li received the M.Sc. degree in computer science from Marquette University, Milwaukee, WI, in August 2007. Since 2005, he has been with the Ubicomp Research Laboratory, Marquette University. His research interest encompasses middleware for ubiquitous/pervasive computing, sensor networks and component-based software development.

Nilothpal Talukder received the B.Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2003, and the M.Sc. degree in computer science from Marquette University, Milwaukee, WI, in 2008. Since 2007, he has been with the Ubicomp Research Laboratory, Marquette University. His current research interests include Mobile and Pervasive Computing, Context-aware Computing, Privacy and Security.

Mehrab Monjur received the B.Sc. degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2005. Currently he is doing MS in Computer Science in the Mathematics, Statistics and Computer Science (MSCS) department of Marquette University, Milwaukee, WI, USA. His primary research interests pervasive security and privacy, trust based authentication, secure resource discovery in ad hoc environment.

Chowdhury Sharif Hasan is currently a graduate student in the Mathematics, Statistics and Computer Science (MSCS) department of Marquette University, Milwaukee, WI, USA. He is doing MS in Computer Science under supervision of Dr. Sheikh Iqbal Ahamed. He completed his Bachelors in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET) in 2006. Mr. Hasan has been working as a Graduate Research Assistant at Ubicomp Research Lab in Dept. of MSCS, Marquette University. His primary research interests are Security and Privacy issues in Pervasive Computing.