Abstract

Portable devices have become day-to-day necessities in our lives. These devices have several critical constraints due to their small size and environment, relying on the importance of middleware in the pervasive computing environment. The significance of security is well-known in this field and it is long overdue to have middleware in the pervasive computing environment that deals with security. In this paper, we present S-MARKS, middleware that is secure by design. Features of S-MARKS include validating devices, discovering resources, modeling trust, handling malicious recommendations, and avoiding privacy violation.

1. Introduction

Today’s technology enables users to be connected by carrying portable devices. Pervasive computing [1] has started to establish its viability in education, healthcare, industry, and elsewhere. These portable devices have limitations that lead to dependency among devices through mutual co-operation. This is enormously significant in a pervasive computing environment. Middleware can play a vital role to cope with the ever growing requirements. A few types of middleware have been designed to deal with these challenges but none can be considered as the ultimate solution.

In a pervasive computing environment, devices can join and leave arbitrarily since the users are very mobile. In these situations, security is a very critical issue. Therefore, very important for a device is the knowledge of current valid neighbors, because any malicious device acting like a valid one can lead to the collapse of security and privacy. As a result, having a protocol to validate devices before communicating is very important.

Resource discovery is another integral part of every device present in this environment [2]. The resource discovery feature explores devices in the vicinity those are capable of offering resources. Portable devices present in the pervasive computing environment typically have resource limitations. Each device is dependent on others in the environment. But due to lack of a fixed infrastructure support and the ad hoc nature of the environment, connections between devices may not remain for a long span of time. Additionally, more than one device may simultaneously request the same service from the same device. Moreover, security has a significant role in resource discovery [3, 4, 5]. Trust is also related to security concerns in the service sharing environment of pervasive computing [6, 7, 8]. Modeling trust relationship among devices is important. Trust will help devices to decide whether to share the service (from both the requester’s and provider’s points of view) after discovering it. Furthermore, handling a malicious recommendation is also a key factor which needs to be addressed. Privacy is another important issue related to security of a handheld device in pervasive computing environment [8-12]. What is the point of sharing resources even to a valid trusted device if one’s privacy is violated while doing that? No one will risk their privacy in order to share resources. There may be a situation when unintentionally some sensitive information is revealed to an unwanted party; this may be termed information leakage and normally leads to a privacy violation.

These aspects of security concerns demand a middleware that is secure by design for the pervasive computing environment. The present middleware [13-24] for this environment does not provide secure solutions in areas of validating devices, modeling trust, handling malicious recommendation, and avoiding privacy violation. Therefore, a novel approach is required to provide such middleware. In this paper, we present “S-MARKS”, a middleware which is secure by design. S-MARKS provides the above mentioned functionalities.

The rest of the paper is structured in the following manner. In Section 2, we discuss the requirements needed for a middleware to be secure by design. We present our approach and the architecture of S-MARKS in Section 3. We also demonstrate implementation of our approach in Section 4. Section 5 contains a brief discussion about related middleware design in pervasive computing environment. Finally we conclude by providing our future works and acknowledgement in Sections 6 and 7 respectively.

2. Required Functionalities

For a middleware to be secure by design, certain functionalities are required. These are briefly presented below.

2.1. Valid Device Discovery

In a pervasive computing environment, a device needs to discover the valid neighbors around it. In order to ensure authenticity of information, every device needs to maintain an authenticated existing device list. A device will not be involved in any kind of interaction with another device that is not present in the valid neighbor list.

2.2. Trust Based Resource Discovery

After the validation of devices, the question arises of trust in resource discovery and sharing. Though all the devices that are interacting with one another at this phase have passed the aforementioned validation phase, the trust level of devices are not the same. The trust level of each device is built up through its interactive behavior and should be updated dynamically through the passage of time. Before acquiring a service, the service requester has to prove that it has the required level of trust.

2.3. Dealing Malicious Recommendation

The trust mechanism works by obtaining recommendation from others directly and indirectly. In this process, a device may provide a false recommendation about the service requester. A mechanism is needed to handle these kinds of incidents. The malicious recommendation should not have any effect on the overall trust related to the service requester.

2.4. Avoiding Privacy Violation

Privacy violation is the most sensitive issue regarding service sharing. After all the required functionalities are in place, people still may not want to share if they think their privacy will be compromised. Thus, a middleware needs to have a mechanism to avoid privacy violation in terms of providing full fledged security in a pervasive computing environment.

3. Our Approach

Our middleware S-MARKS, as shown in Figure 1, is composed of both core components and other components (or services). Core components include ILDD (Impregnable Lightweight Device Discovery), SSRD (Simple and Secure Resource Discovery) with Trust Management [25, 26], and ORB (Object Request Broker). Device Discovery and Communication is part of the ORB. It has an open architecture so that services can be added as components later on. Right now it supports context-service, PriVA (Privacy Violation Avoider), and MaRcHer (Malicious Recommendation Handler). Applications communicate with the services provided by the middleware. Components then contact the Object Request Broker. In the following subsections, we present valid device discovery, trust based resource discovery, handling malicious recommendation, and avoiding privacy violation to have a middleware that is secure by design.

Figure 1: Architecture of S-MARKS

3.1. Valid Device Discovery

In order to restrict the interaction to only valid devices, we developed a device discovery mechanism named ILDD (Impregnable Lightweight Device Discovery). The model has two units: Challenge Response Unit and Validation Unit. Figure 2 shows the architecture of ILDD. Every device needs to maintain a list of authenticated devices to ensure authenticity of information. This list is updated dynamically in a periodic manner. Our method works on a modified version of the well-known LPN (Learning Parity with Noise) algorithm [27, 28]. Here, all the authenticated devices will carry a secret $x$ of a specific length. A leader node will be chosen based on battery power and trust level. Whenever a new node tries to join the network, every other node present in the network will send a challenge to this new node. The requesting node will calculate the result based on the challenge and the secret $x$, and reply to the challengers. Each challenger node will check the answer for validity and send to the leader node its recommendation (true or false) about the requesting

After getting all recommendations, the leader node will declare the new node as valid if the number of true recommendations surpasses a pre-specified threshold. We divided the network into two categories, small and large, and developed two different models for them. Most importantly we provided several possible attack scenarios including active and passive attacks and described how our model can withstand these attacks.

### 3.2. Trust Based Resource Discovery

Resource discovery is a core component in middleware. We developed a trust based resource discovery model named SSRD (Simple and Secure Resource Discovery) [26]. Our model has two functional units: security management unit and trust management unit. The security management unit decides the mode of communication based on the situation and need for a specific service. The trust management unit maintains the trust relationship with others. Figure 3 shows the architecture of SSRD.

![Figure 3: Architecture of SSRD](image)

In our trust model, 0 represents complete distrust whereas 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. The trust model has the characteristics of refflexity and partial transitivity (if \( \gamma \) denotes the trust value by A on B and \( \delta \) denotes that of B on C then the trust value of A on C is a function of \( \gamma \) and \( \delta \)). This model is service and context specific. Here each device maintains a table of available services and corresponding security level that ranges from 1 to 10. The Resource manager of each node maintains a trust table indicating the current trust value of all the neighbors for all available services. Dynamic upgrade of trust values are performed. Unicast, multicast, or broadcast strategies are followed based on the level of security of the offered service. For services with low security levels, no security mechanism is incorporated. But for higher security services, a digital signature based on a public key mechanism is adopted to ensure security. The trust model is elaborated in [26].

### 3.3. Handling Malicious Recommendation

We developed a model named MaRcHer (Malicious Recommendation Handler) to handle malicious recommendation so that it doesn’t have effect on the overall actual recommendation. This helps S-MARKS to handle a false recommendation. It consists of two functional units: alpha and beta. Both the units check all the recommendations, identify the malicious one and try to handle the overall recommendation with the least effect from the malicious recommendations. Alpha unit works when there are few numbers of recommendations. Beta unit comes into action when the model has to deal with a large number of recommendations. Figure 4 shows the architecture of MaRcHer.

![Figure 4: Architecture of MaRcHer](image)

We fit the control chart method [29] in the alpha unit since the number of recommendations is small and we can easily find the standard deviation. But for the beta unit, we utilize the statistical distribution named Student’s \( t \)-Distribution [30, 31]. This approach is used when the standard deviation is not known which is true for virtually almost all real life scenario. In both cases, we get some interval region. We discard any recommendation value that is outside of this region since those values differ from the majority of the recommendations.

### 3.4. Avoiding Privacy Violation

To handle any privacy violation issues while sharing services, we developed a model named PriVA (Privacy Violation Avoider). This helps our
middleware S-MARKS to deal with privacy concerns raised in a pervasive computing environment.

Figure 5: Architecture of PriVA

PriVA has three functional units: service classifying unit, analyzing privacy unit, and avoiding information leakage unit. PriVA communicates with the resource manager to handle any privacy issue in resource sharing. Figure 5 shows the architecture of PriVA. All three units are interrelated and work in close agreement to avoid privacy violation. The Service Classifying Unit deals with classifying the services and resources present in a device. It has two modules named TagR and ClsR. TagR acquires the list of resources from the resource manager and tags each service as sharable or non-sharable. In ClsR, all the resources and services are classified in different default groups. For example, all text documents, spreadsheets, and presentations can be classified as documents.

The ClsR module is also used in the analyzing privacy unit. The other two modules present in this unit are CustomP and ComputeP. We used the concept of policies (also known as access-rights) to analyze services in terms of privacy. Each group, classified in the ClsR module, will have a set of default policies. Default policies may not always serve the purpose; in some cases of resource sharing, additional customized policies are desirable. The module CustomP deals with custom policies to be used in addition to the default policies. CustomP works together with ClsR to provide functionality to the analyzing privacy unit. We articulate these policies for a service using Ordered Binary Decision Diagram (OBDD) [32, 33]. The module ComputeP takes an OBDD representation of policies for a given resource and lists the possible ways to satisfy them.

To minimize the computational overhead at the time of the service request, all resources listed in the TagR module are previously computed and maintained in the PlcT module which is part of the Avoiding Information Leakage Unit (AILU). Privacy is correlated with information leakage from the technological point of view. Information leakage can be thought of as unintentionally revealing (sensitive) information to some party who is not entitled to learn about it. The AILU deals with this issue with the help of the two other units, the PlcT and CompareP modules. The CompareP module handles the policies provided by the service requestor. With the help of ComputeP, it compares the provided policies with the list maintained in the PlcT module for the requested resource.

4. Evaluation

To evaluate the effectiveness of our middleware S-MARKS, we will use prototype implementation, cognitive walkthrough, and performance measurement.

We are currently implementing the S-MARKS. Our test bed includes several Dell Axim X50 pocket PCs (Intel PXA270@624 MHz). We use C# on .NET compact framework. This makes our prototypes compatible with different portable devices supported by the framework. We use IEEE 802.11b for wireless connectivity.

By adopting the cognitive walkthrough strategy, we will collect valuable feedback to measure the performance of our middleware. Subsequently, we will improve our prototype based on the feedback.

We are also planning to measure the performance of our middleware S-MARKS using some criteria such as battery power consumption, processor and memory usage, size of ad hoc network etc.

5. Existing Middleware Solutions

Researchers are involved in middleware design [13-24] for portable devices running in pervasive computing environments. But they are not yet close to providing an optimum solution that is secure by design.

In Reflective middleware [13], 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+) [22] mainly deals with situation-awareness, ephemeral group management, and autonomous coordination for information dissemination. Gaia [21] 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 [16] 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 [18] 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 are MARKS [14], Mobiware [15], TSpaces [17], LIME [19], XMIDDLE [20], PICO [23] and ALICE [24]. MARKS supports knowledge usability, resource discovery and self-healing aspects in the pervasive computing environment. LIME, supporting scarce discovery and self-healing aspects in the pervasive computing environment. 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 and other existing ones did not provide security solutions from the perspective of validating devices, discovering resources, modeling trust, handling malicious recommendations, and avoiding privacy violation. S-MARKS provides these features in a middleware.

6. Conclusion

In this paper, we have addressed the issues of device validation, trust based resource discovery, malicious recommendation, and privacy violation. These are all 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 trust based resource discovery. We are working on implementation to handle malicious recommendations and avoid privacy violations while sharing services.

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

7. Acknowledgement

The authors appreciate the help of Paula Stroud for her valuable comments to improve the paper.

References


