Multiple Personal Security Domains

Reinaldo Matushima¹, Yeda R. Venturini¹, Rony R. M. Sakuragui¹, Tereza C. M. B. Carvalho¹, Wilson V. Ruggiero¹, Mats Naslund² and Makan Pourzandi³

¹Laboratory of Computer Architecture and Network, PCS-EP, University of São Paulo – São Paulo, Brazil
{yeda, rony, reinaldo, carvalho, wilson}@larc.usp.br
²Ericsson Research - Stockholm, Sweden - mats.naslund@ericsson.com
³Ericsson Research, Open Systems Laboratory - Ville Mont-Royal, Canada - makan.pourzandi@ericsson.com

ABSTRACT
Mobility, usability and security are major requirements for any Ad Hoc network systems, and there have been numerous papers in regards to them. However, often these requirements are addressed separately. For a valid solution, these requirements must be considered from an integrated view. In this paper, taking into account mobility and usability, we implement a framework which allows to securely share resources and services between devices in Ad-hoc networks, based on security policies defined by the owners of those devices. In addition, we extend our framework to support inter-domain sharing of services and resources. We detail our design, present the preliminary results of our prototype, and discuss the lessons learned, in particular how user experience led to several re-designs of the initial security solution.

Categories and Subject Descriptors
C.2.0 [Computer-Communication Networks]: General – Security and protection (e.g., firewalls).

General Terms

Keywords

1. INTRODUCTION

Many research projects have proposed service frameworks for mobile computing [1], exploring benefits from these environments to enhance the user experience. At the same time, many other frameworks have proposed security solutions e.g. [2][3]. However, security and usability need be integrated to compose a consistent and valid solution. The challenges are greater for Personal Area Networks (PAN) and MANETs. In such environments there are management, operation and security requirements and constraints that, in existent solutions, are not integrated from end-user view.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IWCMC ’06, July 3–6, 2006, Vancouver, British Columbia, Canada.
Copyright 2006 ACM 1-59593-306-9/06/0007...$5.00.

Therefore, a main purpose of this work was developing a framework that addresses all these issues. The framework supplies authentication, communication security and privacy and access control enforcement in a transparent way. These services are accessed through an API, published for application development. The infrastructure comprises a look-up service and entities that enable remote service discovery and communication. These components enable the creation of Personal Security Domains (PSD).

Besides the challenge of a solution that integrates distinct issues, sometimes with conflicting requirements, this paper also proposes a model enabling domains composition. New domains Multiple PSDs (MPSD) are created from existent PSDs, for easy and secure service sharing. Furthermore, our framework supports a fine-grained access control to individual resources in the MPSD based on security policies, enforced at run-time, which is essential for Ad hoc networks.

The result is a comprehensive study and a prototype implementation. Many issues related to details of protocols, algorithms, and implementation are not discussed. Rather, we give a brief description of the solution and focus on lessons learned through its development. For instance, user experiences with the prototype led to protocol re-design to increase user-friendliness. For more information, see [4][5][6].

The paper is organized in 7 sections. Section 2 presents related work. Section 3 presents the MPSD concepts. In section 4 we present the elements that compose the layer that supports the security services enabled by our solution. Section 5 shows the basics of operation and management and, Section 6 presents information about prototype implementation, and the lessons learned. Finally, Section 7 presents the final considerations.

2. RELATED WORK

In [7] so-called Personal Security Domains were introduced. Motivated by user need to share resources, this extended the concept of PAN to allow part of the PSD to be geographically distributed, i.e. having long distance connectivity to other parts of the PSD. It also allowed individual PSDs to be (temporarily) merged into a so-called ad-hoc PSD, motivated by use-cases, e.g. a business meeting, while allowing access control to still be enforced by each PSD.

In this paper, we extend the PSD concept, prototype our extended PSD solution and additionally implement a Security Enforcement Layer (SEL), providing the needed security support to applications run in the PSD environment. The SEL supports PSD applications with a uniform and easy-to-use interface. The
SEL is a security “middleware”, enforcing a user’s security policies and resource sharing, both within the PSD, as well as when interacting with other users’ PSDs.

This work also comprehends works as [3] which proposes a model for the establishment of Ad hoc network communities, [2] that presents a framework for security provision and [1], which consists in a framework for a service based network. Besides dealing with issues from all these works in an integrated and consistent way, this paper also proposes a reputation-based [8] security enforcement mechanism. In other words, we believe the work in this paper to be first providing a complete, fully operational and practical solution.

3. MPSD

A Security Domain (SD) is a group of fixed or mobile components with security associations, subject to a common security policy. Security association can be established using a shared key or a common Certificate Authority (CA). A shared key allows a light authentication between the SD members, but does not allow an individual member authentication. A CA allows individual authentication, but at higher processing costs.

The SD must have a responsible, which takes a controller role. The SD responsible is a person, who creates the SD with components that he owns or components he has access to, but owned by others. Such a SD is called Personal Security Domain (PSD). Due to mobility of PSD devices and the support for extended PSD (EPSD, see below), the PSD must work independently on the service names, addresses and location.

3.1 Security Domain Partitions

A SD Partition is a subdivision of a SD; formed by all SD components that are in the same location (physical proximity) and can communicate directly (layer 2) among themselves. Partitions are classified by location relative to the SD responsible, that is, partition classification is user-centric:

Local Partition: the unique partition where the SD responsible is present;

Remote Partition: a non-local partition(s). A SD may have zero or more remote partitions. Thus “remote” is here defined by absence of direct, infrastructure independent connectivity.

3.2 Security Domain Types

PSD (Personal Security Domain) - domain formed by the local partition.

EPSD (Extended PSD) - domain formed by the local partition and at least one remote partition. This could be the set of all devices owned by the PSD responsible.

MPSD (Multiple Personal Security Domain) - domain formed by joining of other domains (PSDs and/or EPSDs). Within a MPSD a (E)PSD may share resources (services, information, etc) with other (E)PSDs. On MPSD creation, each controller keeps managing policies related to his own (E)PSD. However, each controller also creates new access rules to permit/deny entities from other domains in the MPSD to access his resources. Figure 1 shows an example of a MPSD composed by two PSDs.

4. ARCHITECTURE

4.1 Personal Security Domains Components

Entities: The PSD entities are applications and devices. Entities are subject to identification to enforce security/authorization. Entities know a shared PSD secret key and have an identity certificate issued by the PSD CA. Some services are required to provide functionality and security in a service-based network. They are the LS (Lookup Service), Controller, and RAS (Remote Access Service). These provide the functionality for PSD creation, use and management.

Controller Entity: The controller performs PSD security management. A main task is creation of the PSD and addition of new entities to it. The controller functions are:

- Domains formation: when a security domain is created a self-signed certificate (PSD CA) is issued, a secret key (seed) is generated and some security parameters are defined, e.g. a PIN (Personal Identification Number), used by entities for registration in the PSD.

- Certificate issuing/revocation: the controller adds new entities in the domain, issuing signed digital certificates to them. Member removal is performed through the Certificate Revocation List (CRL). Revoked certificate are on the CRL until expiration.

- Definition/distribution of security policies: determining conditions for requests to be accepted/rejected.

When creating a MPSD, one of the PSD controllers takes the role of the MPSD controller (e.g. by arbitration), managing addition of PSDs that will compose the MPSD. Thus, the MPSD controller has the role of the MPSD (root) CA. The MPSD CA issues a cross certificate of individual PSD CAs, using the super domain model of Shimaoka [9].

![Figure 1. A MPSD composed by two PSDs](image)

![Figure 2. Example of trust relation in a MPSD](image)
When a MPSD is created, each PSD controller defines access control rules for communication and resource sharing with other MPSD entities. Figure 2 shows how a MPSD member (B1) verifies if another entity (A1) is a member of the same MPSD. A1 sends its certificate chain to B1. The certificate chain contains the identification certificate of A1 (CertA1), issued by PSD controller A, and the PSD controller certificate A (cross-certificate CertPSDA), issued by MPSD controller C. The entity B1 trusts in C as Relaying Party (RP) of the MPSD. The entity B1 is able to verify the certificate path from the RP to A1. The system must have intelligence to use the PSD controller certificate when in a PSD context and to use the MPSD controller certificate when in a MPSD context. Similarly, the system must use only the identification certificate to authenticate an entity in a PS and to use a certificate chain to authenticate the same entity in a MPSD. If a PS belongs to more than one MPSD, it must know the current context and select the correct PSD controller certificate to the considered MPSD.

Remote Access Service (RAS): The RAS enables access to resources across partitions, which can be independent networks, thus dealing with conflicting addresses in different partitions and enforcing security between them. The RAS needs to have a valid IP address but other services do not. The RAS uses http motivated e.g. by facilitating usage in proxy-scenarios.

Lookup Service (LS): In a service-based network, the user does not need to know the service location (e.g. IP address or DNS name). The user searches by service type, receives the available service list, and selects a service to use. The list of available services has all information required for service connection. Example service location protocols, are SLP or JINI. We developed a new service location protocol [5] with additional functionalities related to the remote connections and security. The Lookup Service (LS) is the entity taking the service locator role. It advertises available services and handles access conditions for PSD members. In the case of EPSDs, the LS identifies to members which services that are in the local/remote partition, and the RAS available for the remote service access. Figure 3 shows remote service discovery.

![Figure 3. Steps of Remote Services Discovery](image)

1. The client D1 (Device 1) requests search and update of the available remote services to the local LS.
2. The LS sends the connection request for the remote partition to local RAS.
3. The local RAS establishes communication with the remote RAS.
4. The remote RAS discovers the LS location and establishes the communication.

Through RASs, a temporary logical end-to-end connection is established between two LSs, which update their remote service lists. Details on remote services usage, LS and RAS protocols and dealing with Network Address Translation (NAT), firewalls and mobile connections in WANS considered in this work are described in [5].

5. SECURITY ENFORCEMENT LAYER (SEL)

The SEL provides secure communication between entities of the (E/M)PSDs with a goal of user/application transparency. The SEL ensures authenticity of entities, security of the communication and access control of the applications. The SEL dynamically enforces the security rules for the MPSD. Upon each update of security rules, the new rules are propagated to SEL and enforced. SEL is also responsible for management of entities’ trust values, which are used in the trust based authorization model (see Section 6). SEL has two basic components: the SelSocket Encapsulation (SSE) and the Security Domain Data Manager (SDDM). The SSE enforces the security and the SDDM manages security data. Only an overview on how the security services are supported by the SEL is presented here. Information on SEL protocols and performance results can be found in [4].

5.1 Security Domain Data Manager

The SDDM is composed of the Security Domain Data (SDData) and an API for management see Figure 4 The SDData is distributed and each device has its own SDData containing the information of the SD entities (client and service applications) hosted in the device itself. The SDData is composed of listings of: controllers, domains, members and rules.

![Figure 4. SD Data Manager Architecture](image)
5.2 SelSocket Encapsulation

To make it easier and more transparent for application developers, the SelSocket works similar to a standard socket. The SelSocket components are shown in Figure 5.

The SelSocket supports two authentication levels: domain and entity levels. The first level, Domain Authentication, authenticates a SD (PSD or MPSD) member based on shared keys (the SD “seed”), see [10]. This level also provides the base security and privacy for the second level Entity Authentication, which is based on the certificate. This level can authenticate an individual entity and also provides mutual authentication. At each authentication level, the SelSocket uses a different key for confidentiality of the communication (performed by the Crypto component). Specifically, a key generated by the Domain Authentication is used to secure message exchanges during the Entity Authentication (thus providing third party privacy for certificate exchange), and the key generated by the latter authentication is finally used to protect application data.

The authorization component supports a fine-grained authorization. For example, it is possible for an application to request authorization. For example, it is possible for an application to authenticate is finally used to protect application data. For example, it is possible for an application to

6. MANAGEMENT AND OPERATIONS

6.1 Management

For the prototype implementation, we created a management tool. It is responsible for:

- Device Management: operations related to the device itself and to domains that it is part of, e.g. creation of a new PSD (taking the controller role for the new PSD) and joining the device to an existent PSD.
- Application Management: operations related to the applications within each device including addition of new applications and their execution.
- Controller Options: operations related to controller entity, enabled if the device is hosting the SD controller.

As described in section 4, the controller is responsible for the security domain management. The operations associated to the controller are: to create a new MPSD, to join the PSD to an MPSD, and, finally, to manage and distribute the access control rules associated to its PSD.

We adopt the least privilege approach in (M)PSD: all accesses from a client to a servers is denied unless explicitly allowed by a rule. The rules are distributed from the controller to the concerned entities. They are then enforced by the SEL, transparently to applications. The rules are also sent to the LS. According to these rules, the LS only divulgate services to authorized clients. Therefore, we enforce the security rules both at service discovery and service access points.

6.2 Service Usage

Before using a service, the client entity needs to locate the available services. The client sends a query to the LS. This query can contain service attributes so that the LS can filter the response. Then, the client selects the service and starts the service usage request.

6.3 Authorization

Within security domains, before any information exchange, three phases are required. The first one is the Domain Authentication, followed by Entity Authentication mentioned above. The third phase is the authorization process (see Figure 6). In the proposed authorization model, the purpose was to create a complete access control solution based on the trust concept that could be effectively applied to meet the restrictions of Ad hoc environments and to provide a solution where the access control enforcement is automatically changed based on the (mis)behavior of the members.

![Figure 6. Authorization Steps](image-url)
7. PROTOTYPE

7.1 Implementation

A complete implementation was developed in Java. PSD services (Controller, LS and RAS) and a simple peer-to-peer file sharing for test purposes were implemented. In the SEL implementation, authentication and authorization are done as defined by security policies in an XML file. The described APIs allow software development for applications to run over PSDs were implemented. The SEL uses X.509 certificates and AES encryption provided by the BouncyCastle package.

7.2 Scenarios

To give an impression of usability, we describe user actions and data flows associated with a few basic scenarios.

(M)PSD Creation: The user (PSD owner) launches the Controller. The user initiates the PSD creation using the Controller GUI. He selects a PSD name and the access PIN for the PSD (Figure 7). The PIN is then used to add different entities to the PSD (application, device) as part of a short-lived key, securing the entity addition. For usability reasons, a similar approach was adopted for MPSD formation. One PSD controller is chosen as MPSD controller and at this time, the user enters the MPSD information (e.g. the selected MPSD name and its PIN). For other PSDs to take part in the created MPSD, each user must request to join through their PSDs controllers (entering the PIN, exchanged out-of-band).

![Figure 7. PSD Creation](image7)

During MPSD formation, each PSD owner defines which entities in their PSDs will be associated/shared to the MPSD, i.e. which entities that will have an “allow” access rule for the MPSD and will receive the cross-certificate issued by the MPSD Controller. Through these steps the trust relationship is established between different MPSD entities (see Figure 2). This enables the secure communication between different domain entities and the policy enforcement control (see section 6) enhances a fine grained security.

Exemplary MPSD application: File sharing: Assume the MPSD is created as above and that all entities from each PSD were associated to it (a typical default), see Figure 8 For the Filesharing application in device A1 to accept a request from MPSD entities, Alice must create an access control rule allowing this. When Bob wants to share files with Alice, he first looks for (Alice’s) available services. Bob selects the service, the access control rules are checked, and a (secure) connection is established for Bob to download Alice’s files. If one of the PSDs is an EPSD, the steps look exactly the same from users’ point of view. In this case, the communications between partitions are transparently handled by RAS.es.

![Figure 8. File sharing through MPSD](image8)

7.3 Lessons Learned

Our prototype showed that the MPSD functionality can be implemented using high-level mechanisms like Java and Http. Though an OS level implementation would be more efficient, our preliminary results show an acceptable performance and ease of integration for the tested applications.

Making SEL transparent to user applications: The project provided transparent security services to the users. To this end, one of the main challenges became creating an API easy to use by application developers, providing security in a transparent way while requiring a minimum change in existing applications. We achieved this by wrapping the Java socket API by SEL functionality.

Optimizing the number of messages exchanged in protocols: Another problem was to make the right trade-off between user friendly transparent protocols which need more information exchange between entities and less user friendly protocols demanding less information exchange to limit power consumption of small devices. To solve this issue, we adopted a mix of manual and automatic information exchange. An analogous problem was communication between EPSD partitions, with a high number of message exchanges. This was solved by all operations starting by an explicit manual request, e.g., to discover the remote services (Figure 3).

Usability: All sensitive information in each device are stored securely. A distinct access key can be assigned to each device and controller (see section 5) and users are prompted each time secured information is required (e.g. certificate recovery). During the prototype evaluation, we realized that this protection mechanism is not very user friendly. To avoid the users having to enter the access keys several times, the key is cached in the memory and asked only the first time. For sensitive operations (e.g. domain management), the access key is, however, always requested.

When a new entity is added to a PSD, there is no previous trust relationship. For security, a shared secret key is used (based on a PIN) and the same mechanism was adopted in the addition of
all entities (see 7.2). During the demo, we found usability issues in requiring the PIN value each time an entity is added. We re-designed the member addition mechanism (during the PSD formation), requiring the PIN only for device addition. When an application is added to the PSD, we bootstrap: if the device is trusted in the PSD, then so are its hosted applications.

Security policy management: a main difficulty was to reduce complexity for security policy management. Each PSD owner is responsible to configure security policies. By design, we made many operations completely transparent to the users, for example the certificate management (performed by the Controller and SEL). However, users must sometimes perform operations like managing access rules. In our prototype the user creates permit/deny-rules, which can be further conditioned by parameters like time, access number limit, trust value, etc. We have three rule levels: rules targeted to all domain entities, to a specific device, and targeted to a specific application. Although a GUI for rules management was created, the user still needs to know the precedence between rules. This can be expected from a network administrator, but is quite problematic for a normal user.

Another problem related to security management is that users tend to opt for “insecure” configurations due to the complexity and flexibility restrictions resulting when security options are enabled. Users often lack insight in the security implications of their choices. It is often said that security works best when neither the presence, nor the absence, is noticed by the user. Concerning rules management, a new model is being developed to enable the access control through associated, pre-defined security profiles (“low”, “medium”, “high”, etc). The last issue was solved imposing a minimum-security configuration.

Time synchronization: The domain authentication protocol is based on time stamps [4]. The security could be tightened if the device’s clock synchronization is also implemented. In this case, replay attacks would always be ignored by the entities. Clock synchronization in a mobile network is not an easy task, and a time window of a few minutes was allowed to ease the problem. However for a MPSD, it is very uncommon that users have good clock synchronization and may call upon users to adjust their devices’ clocks, representing a usability problem. We decided to allow user defined time window or to disable the time window verification. An alternative to time stamps would be to use nonces. A drawback of this is that it adds at least one roundtrip to the protocols to reach security.

8. CONCLUSIONS
In this paper, we proposed a framework for securely sharing resources through personal and merged domains over distributed and mobile Ad hoc networks. We believe that our approach explored a more complete approach different from other research projects in that we take into account mobility, usability, and security. This resulted in new challenges, explained in this paper.

A security model to support virtual security domains was developed, and the main components of the infrastructure were developed. In this security model, the concept of personal security domain (PSD) has been extended to the composition of several security domains: Multiple PSD (MPSD). The infrastructure and SEL implementing this security model were presented. The result is a middlelayer software that provides secure communication in a transparent way. We believe that this design can be used further to support virtual security domains in Ad Hoc networks. Another result from this work is an authorization model that supports granular access control enforcement and takes into account quantitative trust, enabling an adaptive enforcement based on members’ (mis)behavior. At the end, we presented our experience from the prototype implementation and showed the feasibility of our approach. The lessons learned can be useful to other projects tackling the complexity of interactions between mobility, security and usability.

9. REFERENCES