Available online at www.sciencedirect.com

ScienceDirect

Computer Communications 29 (2006) 3277–3294

computer communications

www.elsevier.com/locate/comcom

A framework for seamless service interworking in ad-hoc networks

Linda Källström [a,1], Simone Leggio [b,*], Jukka Manner [b], Tommi Mikkonen [c], Kimmo Raatikainen [b], Jussi Saarinen [c], Sanna Suoranta [a], Antti Ylä-Jääski [a]

[a] *Helsinki University of Technology, TML Laboratory, P.O.Box 5400, FIN-02015, HUT, Finland*
[b] *University of Helsinki, Department of Computer Science, P.O. Box 68, FIN-00014, University of Helsinki, Finland*
[c] *Tampere University of Technology, Department of Information Technology, P.O.Box 553, FIN-33101, Tampere, Finland*

Received 17 June 2005; received in revised form 3 May 2006; accepted 12 May 2006
Available online 8 June 2006

Abstract

Local area wireless networks are becoming commonplace in our everyday lives. It would be beneficial to establish such wireless networks in an ad-hoc manner so that they are infrastructure-free. In the current IP networks, various infrastructure elements play cordial roles, and most of the service-related Internet technologies cannot be deployed in infrastructure-free ad-hoc networks. In this paper, we develop mechanisms for seamless service interworking in ad-hoc networks: first, finding devices, people and services in an ad-hoc environment, and then establishing communications sessions between two or more parties. To guarantee interworking with current networks, we reuse as much as possible well-known client–server Internet technologies and with minimal changes modify them to be applicable in a peer-to-peer manner in ad-hoc networks. We have developed methods for service discovery, session management, and security support that can be used in infrastructure-free ad-hoc networks. We have verified the developed methods with a proof-of-concept implementation in a WLAN testbed network. We also evaluated the realized implementation, in terms of bandwidth and CPU consumption. We present the evaluation results to conclude the paper.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Ad-hoc networks; Seamless service interworking; Service discovery; Session management; Authentication and authorization; Security; SIP; SLP

1. Introduction

Wireless local area connectivity will be supported by numerous complementary technologies in the near future. Local area wireless networks have already gained a lot of momentum through the deployment of WLAN technologies in enterprises, in public hot-spots, and in homes. Another popular short-range wireless technology is Bluetooth, which is deployed in mobile handheld devices and laptops. Ad-hoc networking without the need for centralized servers provides a highly potential prospect for new types of communication scenarios. In this paper, we address issues that are common challenges to all local area wireless ad-hoc networks in order to support seamless service interworking. We argue that the current technologies as such are not mature enough for commercial level deployment for ordinary consumers.

The end-user perspective must be emphasized when we aim to provide services in new communication environments like wireless local area ad-hoc networks. In order to make ad-hoc network communication environments appealing for consumers, an easy-to-use user experience must be supported. The technical details, e.g., complexities in configuring the systems, must be hidden from the end-users. This is a major challenge, since none of the currently available technical solutions is mature enough as such for building a complete system. The solutions developed here provide an easy-to-use user experience for utilizing services in ad-hoc environments. The use-cases that our system supports include, for example, confidential information

* Corresponding author. Tel.: +358 50 4869494; fax: +358 7180 37290.
E-mail address: simone.leggio@cs.helsinki.fi (S. Leggio).
1 Authors in alphabetical order.

distribution based on group memberships, multiparty gaming sessions with authenticated players, and confidential multiparty conferencing over unsecured radio networks. When the users and service providers so wish, the above services could be executed to anybody without security as general public services.

Seamless service interworking means the ability to find devices, people and services available in the ad-hoc environment in order to set up communication sessions and inviting multiple parties into them. In this context, the framework for service interworking in ad-hoc networks covers IP-based security, service discovery and session management. We believe that in the future, IP will be the most common network layer technology; thus, we focus on wireless ad-hoc environments providing IP-based network connectivity. We define the ad-hoc environment so that there are no centralized servers available for either network or service related functionalities. This lack of centralized servers creates a fundamental technical challenge in ad-hoc networks, since most of the techniques used in today's IP networks make use of servers, e.g., DHCP and DNS. There are several proposals for how to enable various lower layer techniques in ad-hoc environments; however, service level interworking in ad-hoc networks has not been addressed in detail in the literature.

Reusing existing, already adopted, promising technologies to maximal extent is the fundamental guideline for our research methodology. Extending the current technologies with minimal changes is beneficial compared to reinventing completely new solutions. Thus, it is more realistic to penetrate the new ad-hoc marketplace using the common technology basis, whenever feasible, that has been used in the infrastructured networks. The same holds for interworking between different devices and services in ad-hoc networks. We look for the best candidate technologies that call for minimum changes in order to fulfill the requirements in ad-hoc environments. In a way we define an architectural framework, but we do not develop new reference architectures. Our solution builds on individual technologies – mostly further developing Internet Engineering Task Force (IETF) protocols – and our architecture defines how to build a complete functional system with the enhanced protocols.

In this paper, we develop middleware services to enable security, service discovery, and session management in wireless ad-hoc networks. Proof-of-concept prototypes have been developed and implemented on Linux laptops. The implemented methods have been combined into a larger prototype system; we have been able to demonstrate seamless service interworking in 802.11 WLAN ad-hoc networks. The resulting system is a distributed system that does not require centralized servers for any of the functionalities in wireless ad-hoc networks. Considering security, we have developed a Public Key Infrastructure (PKI)-type authentication and authorization (AA) module that works in a peer-to-peer manner without servers in an ad-hoc network. We have modified the Service Location Protocol (SLP) [1] so that it can autonomously provide service discovery functionality in ad-hoc networks without servers. For session management, we have modified the Session Initiation Protocol (SIP) [2] so that it can be used in ad-hoc networks in a distributed manner without SIP proxies, registrars or others servers. The modified SLP and SIP modules can use the developed security module in order to authenticate or authorize users and services; for SLP, confidentiality also is supported. Multiple users and several service providers can co-exist and communicate simultaneously in the ad-hoc network, for example, establishing secured multiparty communication sessions. The current design and implementation has been tested in an isolated ad-hoc network. However, the system is designed in a way that it can be extended to support service interworking between ad-hoc networks and the Internet.

The described work is based on research carried out within the scope of the SESSI (Seamless Service Interworking in Heterogeneous Mobile and Ad-Hoc Networks) project. The project is divided into two phases, where the first phase is restricted to ad-hoc networking in an isolated network, and the second phase extends the scope to actual interworking with the Internet. The results presented in this paper are based on the first phase. The rest of this paper is structured as follows. Section 2 introduces research challenges related to service discovery, session management protocols, and security, together with the necessary background information. Then, Section 3 introduces the technology choices we made within the scope of the SESSI project. Section 4 discusses the way we have integrated the different software components to form ad-hoc capable secure session management and service discovery applications. Section 5 provides an evaluation of the realized implementation, and Section 6 concludes the paper.

2. Research challenges in ad-hoc networks

This section briefly reviews the state-of-the-art for the services that form the SESSI service framework. In addition, it addresses the challenges that must be faced when trying to deploy these services in a peculiar environment, such as ad-hoc networks. Related work on the deployment of security, service discovery, and session management architectures in ad-hoc networks is presented to conclude the section.

2.1. Security

As with ad-hoc network research in general, research into the security issues in the ad-hoc networks is also heavily concentrated on routing. However, there are many other security challenges than just routing. Frank Stajano and Ross Anderson have listed that denial of service, authentication, and naming are the most demanding security problems because they differ most from those found in conventional network environments [3]. In the following, we describe the security challenges and the current state of the art of security solutions for ad-hoc networks, mainly from the authentication point of view.

Network security consists of authentication, authorization, access control, integrity, confidentiality, and availability. The biggest security problem is availability because in an ad-hoc network denial of service attacks are easy to execute by totally jamming the radio network by sending garbage signals over the frequency spectrum. Authentication and authorization are key components for the access control that is essential in ad-hoc networks: nodes in the ad-hoc network offer services to each other thus acting as service providers. As a consequence, all the nodes should have some kind of access control mechanism that indicate which other node (or user) can use, how, and which services offered by the node. When the other party is authenticated or its authorization to use the service is verified, it is easy to agree on the security parameters for the communication: an encryption method for confidentiality and a signature method for integrity protection.

Two popular wireless networks in use today are Bluetooth and Wireless Local Area Network (WLAN). Bluetooth can interconnect laptop computers, mobile phones, handheld computers (PDA), and their accessories such as handsets. Bluetooth has a short radio range, and its radio uses a frequency hopping mechanism that makes the jamming of connections and eavesdropping harder for an attacker [4]. Bluetooth only authorizes or authenticates devices, not users. The device authentication is based on shared keys set by the users. The security architecture of Bluetooth provides three levels of access control, that is access to all, authentication required, and authorization and authentication required. Access restrictions of services can be defined separately for each service. The Bluetooth security architecture is suitable for ad-hoc networks, but the authentication based on a user that has access to all devices does not scale for larger coverage networks such as WLANs.

Similarly to Bluetooth, WLAN security is based on shared keys between an access point and user devices. The user device is authenticated with the shared key, and the communication between the device and the access point can be encrypted. WLAN security does not provide mutual authentication: only the user device is authenticated. Moreover, even the authentication is not automatically in use. The WLAN security mechanisms have several vulnerabilities, described in [5]. Besides, peer-to-peer ad-hoc networks need mutual authentication that is not provided by the WLAN security mechanisms.

One good solution for providing ad-hoc type authentication and confidentiality is Pretty Good Privacy (PGP) [6]: PGP offers a "web of trust" type of key management for public cryptographic keys. In PGP, a user signs the key of another user, and this key can be used both for verifying the origin and encrypting the content of a message. Every user has his own storage for keys, and the trustworthiness of the keys is marked. For example, a key can be marked to be always trusted. PGP requires either face-to-face authentication between the users or earlier contact with some trustworthy user that is known to both of the communicating peers, but the keys can also be available for only temporary use. In particular, PGP is mainly targeted for electronic mail, and it does not provide enough flexibility for different security services that devices and applications in an ad-hoc network can offer to each other.

In the last decade, decentralized authorization mechanisms have been developed to offer authorization provided by the service owner. For example, Simple Public Key Infrastructure (SPKI) [7] allows the owner of the service to create certificates that can be further delegated. When a user wants to use the service, he presents a chain of certificates that certify his right to use the service. The service owner verifies the certificate chain. Another example of a decentralized authorization mechanism is KeyNote [8]. It works in a similar way allowing trusted third parties to be used as repositories for the credentials. KeyNote provides a separate compliance checker that verifies authorizations. In an ad-hoc network, either a server is needed or every node should have its own compliance checker. Both of these mechanisms require a lot of computing power and can be too heavy for small devices. Some decentralized authorization systems assume a few trusted powerful devices that perform most of the resource-intense operations [9,10]. Decentralized authorization mechanisms still have serious usability problems that make them hard for users to understand; this is a big problem with ad-hoc network devices that are maintained by the users themselves [11].

This section has described different security problems and some solutions proposed to solve them. None of the solutions are well-suited for an ad-hoc network environment because devices lack computing power and the environment does not provide servers. In Section 3.2, we present our decentralized solution for authentication and authorization management in ad-hoc networks.

2.2. Service discovery

Generally, there are two different ways to locate a service in an IP-based network: its address is known or its address is queried for. However, knowing the address is not an option in an ad-hoc environment, and therefore we concentrate on the latter. At the level of infrastructure, support is commonly offered for service discovery. In fact, there are numerous approaches currently in use in the fixed Internet, so one would assume that this could be copied to the ad-hoc environment as well.

Unfortunately, implementations of many service discovery mechanisms are based on centralized and dedicated high-performance registers or databases that contain information regarding several service providers. This scheme is not suited for an ad-hoc environment, where the resources of nodes, including, e.g., performance and bandwidth, are restricted, and no bound roles exist, but all nodes can assume general roles.

Based on the above, a revisited mindset must be adopted. Provided that nodes can assume several roles even in parallel (a node acting as a server in one application can be in the client role in an other application), and change their roles dynamically, it is up to individual nodes to provide information about their current roles and services they potentially provide. For this purpose, several protocols are available. In the following, we introduce the most relevant ones within the scope of this paper.

The Service Location Protocol (SLP) [1], is an all-IP protocol for locating services in the Internet. SLP is a mature protocol and there are open-source implementations available. Universal Plug'n'Play (UPnP) [12] is a protocol for establishing ad-hoc networks. The purpose of the protocol is to simplify, e.g., home networking by enabling seamless interworking between PCs, intelligent appliances, and wireless devices. Web service discovery (WSD) [13] is a relatively new protocol for locating web services distributed in a network. However, the standard seems to be relatively immature, as it has been experiencing constant evolution. The same applies to available implementations. Peer-to-peer protocols that operate at the IP-level are also available, such as JXTA [14]. These protocols could be used as a basis for service interworking between ad-hoc networks and the Internet, assuming that the services of the Internet would rely on the same protocol.

To enable seamless interworking, the selected protocol should be capable of locating services over a wireless connection, regardless of whether the services are situated in the Internet or on another peer of an ad-hoc network. For practical purposes, we also wanted to have a relatively mature specification and implementation. These were the reasons that led to the decision to use SLP as the basis for our work on service discovery presented in Section 3.3.

Service discovery protocols that use a centralized register are not generally suitable for use in an ad-hoc environment. While SLP allows the use of such a register, it is also possible simply not to install such a service, and only use parts that are related to service providers and users. This was our solution to achieve fully distributed operation.

Due to security threats associated with the ad-hoc environment presented in Section 2.1, we deemed that SLP's original security scheme [1] is insufficient. The original one way authentication system was replaced by mutual authentication, real-time timestamps were replaced by logical timestamps and encryption was added. This was realized by using our new AA module.

2.3. Session management

Among the various session management and telephony signaling protocols emerged so far, two are particularly worth mentioning: the Session Initiation Protocol (SIP) [2], standardized by the IETF, and H 323, developed by the International Telecommunication Union (ITU) [15].

The two proposals aim essentially at defining a standard for telephony signaling, although their design and overall functionalities are noticeably different. While SIP is text based, H 323 is binary encoded; this makes the SIP protocol more flexible and easier to extend. Indeed, the SIP protocol has been explicitly designed with the possibility of defining extensions and enhancements to support new features. This structure gives the possibility to build more lightweight SIP end devices, which only need to implement the basic protocol functionalities, and eventually the extensions needed by the application. The price to pay for this flexibility is the large size of SIP messages. H 323 is binary encoded and therefore the header overhead is smaller, which is a beneficial factor in mobile environments.

The specification of the baseline H 323 is extensive; the protocol was designed with the intent of providing a large set of telephony related functionalities. Telephony functionalities in H 323 are richer than in SIP and more maturely defined. This is due to the fact that H 323 was initially designed specifically for IP telephony signaling, and with the goal of providing services as close as possible to traditional telephony. SIP session descriptor parameters are carried in the body of SIP messages; since SIP can transparently carry any kind of body type, it is suitable for a broader set of applications than H 323. The extensiveness of H 323 specification has an implication for the complexity of the H 323 client devices. They must in fact support the whole large set of features of H 323 to be fully compliant, and therefore they are not always the best choice for implementation in a small, resource constrained, mobile device.

In the literature, there are several comparisons between the two protocols. A good survey was presented by Glassman et al. [16], where the authors briefly describe the two protocols, compare the services provided and evaluate them. The included forecast suggests that the two standards will coexist, as they provide different functionalities. We have chosen to use SIP for our framework due to its flexibility, advanced maturity level and also because it has been chosen by 3GPP as the signaling protocol in 3G networks. This is an important factor, as a potential extension target for our architecture lies in interworking with external infrastructured networks, such as the Internet or 3G. Interoperability for session management procedures is easier if the same protocol is used in both ad-hoc and infrastructured network environments.

The inherent problem with the deployment of SIP and H 323 in ad-hoc networks is that the protocols strongly rely on servers. SIP users in particular, must register their presence in the SIP network to a SIP server, called a registrar. In the majority of situations, SIP users only know the SIP user name (address of record, AOR) of the remote party to contact, but not the exact contact address where they can be reached. For example, when inviting the SIP user Alice, whose AOR is sip:alice@example.com, the caller cannot know whether Alice will pick up the call on her work desktop computer, or on her home laptop. This information is given by Alice to the registrars during her registration operations. Other SIP entities, the proxy servers, take care of forwarding the message addressed to a particular AOR to the current contact destination, by accessing the user's contact information stored in the responsible registrar.

The standard SIP architecture is not feasible for ad-hoc networks, which are created on demand by end devices, and where no support from the network is available. As a result, the SIP protocol cannot be deployed in ad-hoc networks, due to the lack of centralized servers taking care of registering SIP users and forwarding messages on their behalf. In ad-hoc networks, without the support of proxy servers, SIP users do not have means to reach other users. Without registrars, SIP users cannot be reached by other users as there is no entity for them to which communicate their contact information. Our solution to this problem, detailed in Section 3.4, is to make SIP operations decentralized and distributed among all the nodes forming the network, so to bypass the need for centralized entities support.

2.4. Other challenges for IP connectivity

Auto-configuration is one of the fundamental challenges in unmanaged networks, such as ad-hoc networks. Basic functionalities of IP communication include the allocation of an IP address, and setting up the IP routing, e.g., allocating a default router and gateways to other networks. In addition, for an application to be practical, DNS information is needed. There has been a lot of very different work in this area. An IP address can be allocated for each link separately with IETF-based mechanisms, the stateless address auto-configuration for IPv4 [17] and IPv6 [18]. Yet, some research suggests that IP address management should be tied to the underlying ad-hoc routing protocol [19]. The IETF MANET working group has only recently started looking into a common mechanism to perform IP address allocation in a multi-hop ad-hoc network. A proposal for address auto-configuration for ad-hoc networks, and a review of other proposals, can be found in [20].

Gateway discovery for ad-hoc networks has also received attention. Two distinct mechanisms can be identified, to tie the discovery to the ad-hoc routing protocol, or to flood router advertisements into the ad-hoc network and let the nodes choose the best gateway to use. DNS information can be provided with a similar scheme than the address of the gateway. In our research project, we decided to focus on the higher layer problems, and rely on existing work in the area of auto-configuration of the IP connectivity.

2.5. Related work

Several interesting secure service discovery schemes have been presented. One of these, the one that most closely resembles our work, is Service Discovery Service (SDS) [21]. SDS efficiently protects the authenticity and confidentiality of services with a scalable architecture that relies on a dynamically formed server hierarchy. The solution has five distinct entities: SDS servers, services, capability managers, certificate authorities and clients. The services may register themselves to the SDS servers which in turn advertise them to clients. Clients may also make requests for certain services to the SDS servers. The capability managers and certificate authorities are used to aid the other entities in the security implementation. Capability managers help the network hosts to establish the access control features while certificate authorities are used to distribute certificates between nodes. In SDS each service is responsible for contacting the capability managers and setting up the capabilities for single users. Communication between clients and servers is handled with Authenticated Remote Method Inoculation and the messages are mostly signed, with timestamps, and encrypted. The service data is stored in XML format and the searches are made with the XSet XML engine.

The hierarchical server structure of the SDS can offer a great deal of scalability, but it fits poorly in ad-hoc networks since a dynamic hierarchy would be infeasible for nodes in the ad-hoc environment to uphold. Inside the hierarchy service information flow from one server to another could also lead to situations where the information on some servers could be invalid. Our implementation has no such limitations because it is fully distributed although this is achieved by preset keys and certificates. The SDS implementation also relies on the correctness of standard timestamps. This issue is solved in our implementation by the use of logical timestamps described in Section 4.1. The SESSI SD scheme has limited scalability but is however aimed for notably smaller networks than the SDS.

There is not much work yet done on the decentralized deployment of SIP. An Internet Draft [22], individual submission to the IETF SIPPING Working Group, proposes the decentralization of SIP registration operations by means of distributed hash tables (DHT). The use of DHTs removes the need for centralized entities in the SIP architecture. However, DHTs are not optimized for mobile ad-hoc networks, as the maintenance operations and the look-ups related for keeping the DHT update result in a large amount of SIP messages, which is an undesired situation in mobile environments.

Other work [23,24] specifically addresses modification to SIP for deployment in ad-hoc networks. The proposals share the idea that the exchange of messages is tied to a specific underlying ad-hoc routing protocol. Our approach does not impose this strong requirement. Decentralized SIP requires that messages used to register a newcomer node to the ad-hoc network are distributed to multiple users at the same time. In link-local ad-hoc networks, this implies broadcasting. When decentralized SIP is used in a multi-hop networks, multicasting is used, and the only requirement posed is that nodes use a multicast-capable ad-hoc routing protocol.

Currently used authentication and authorization (AA) systems rely heavily on centralized servers. Network service providers can offer authentication services, but in a pure ad-hoc network, such network service providers or servers do not exist; other means are needed for authenticating

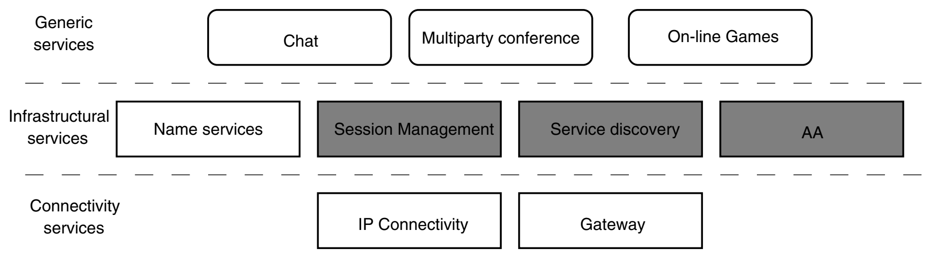

Fig. 1. The service layers in the SESSI architecture.

users. Schemes are proposed for situations where either service providers [25] or even the ad-hoc network nodes [26] have concurrent access to an overlay network, but these do not address truly infrastructureless environments. Schemes have also been presented on how to form a chain of trust to a key [27] or how to use recommendations as the basis of trust [28].

3. A service framework for ad-hoc networks

This section presents the solution we have envisaged for deploying the services selected for our framework in ad-hoc networks. Although in this section they are presented as independent modules, they are closely related to each other; their interoperation is discussed in Section 4.

3.1. Architecture of the framework

The SESSI service framework consists of three layers: connectivity services, infrastructural services, and generic services (Fig. 1). Each layer is further divided into modules. Each module may use the functionality of layers that are located under it and of the modules that are at the same layer with it.

Connectivity services are the lowest layer of the SESSI framework. They provide IP connectivity within the ad-hoc network and possibly connectivity to external networks for other parts of the SESSI framework. In the first phase of the SESSI project, the main focus was in developing the infrastructural services shown in the shaded boxes in Fig. 1. The second phase will extend the framework to connectivity services, where the focus is placed on the interoperability with external infrastructured networks.

Infrastructural services form the core of the SESSI framework. They provide an abstraction layer for the generic services on top of the connectivity services. Our framework aims at providing a seamless way to provide a connectivity-independent way for locating, setting up, and accessing services. The AA module located in this layer forms an exception to other modules. It can be accessed from all parts of the framework. Generic services are the top layer of the SESSI framework. They are the actual services offered to (or by) users. Examples of generic services include instant messaging and presence applications, games, and multiparty conferencing. The design of the SESSI framework has been optimized for use in small link-local ad-hoc networks, where all the nodes are on the same link, and the size is restricted to at most a few hundred nodes. The reasons motivating this choice are that we have identified several use cases for our framework. Most of them could be well-addressed by a link-local ad-hoc network, without the need for the introduction of additional complexity due to IP routing issues.

Most of ad-hoc networking related research addresses big ad-hoc networks with several hundreds of nodes or even thousands. We believe that such big ad-hoc networks cannot be realistically deployed, at least with current technologies[2]. The same concept about the realistic sizes of ad-hoc networks is shared by Tschudin et al. [29]. They feel that simulation-based studies for very big ad-hoc networks are not the right approach to achieve wide deployment of ad-hoc networks in real life. Their claim is that, although simulation-based research on ad-hoc networks has produced noticeable results, there is the need to carry out experiments addressing realistic MANETs deployment scenarios. Such realistic scenarios are deemed to be small-scaled ad-hoc networks, with a few dozen nodes (typically, around 20) at most, two or three hops away from each other. In addition to this, we believe that, when possible, exploiting link-local environments must be carefully considered. A few dozen nodes directly connected at the link layer would form realistic ad-hoc network, where the nodes are relieved from routing operations, suffering therefore from less power consumption. Increased batteries lives and bandwidth availability due to the absence of routing control messages can be powerful triggers for the commercialization of ad-hoc networking.

For example, an attractive scenario is a meeting or a classroom, where the speaker shares his slides with the other participants on their laptops connected in ad-hoc mode. The secretary types the minutes on a whiteboard, and all the participants can read them while they are typed. The session management procedures needed for setting-up the shared slide show can be deployed with our decentralized version of SIP. Security can be enforced using the mechanisms defined for our framework.

Similarly, in an Internet cafe customers may discover in ad-hoc mode other people with a matching profile using

[2] This reasoning does not apply to sensor networks, where we do acknowledge that sizes of thousands of nodes constitute an effective use case.

SLP, and begin a SIP-based instant messaging session with them. Communication would be free of charge, as it does not utilize any infrastructure. An access point would only be utilized for communication with nodes in the Internet, and would not be overloaded by internal communication. The decentralized approach also suits other kinds of network environments where the link-local paradigm holds, not just ad-hoc networks. For example, the SESSI architecture can be deployed in the machines forming a LAN in a university campus. SESSI operations would be performed in a decentralized way and servers would only be used for operations that cannot be made decentralized (e.g., mail server). The difference between these scenarios is that, unlike ad-hoc networks, a LAN environment can support several hundred of nodes.

Although the main target network environment is small link-local ad-hoc networks, our framework can also be deployed in bigger ad-hoc networks, where nodes are multiple hops away from each other. In link-local ad-hoc networks messages are distributed to all the users in the network by broadcasting them. Broadcasting is an efficient trade-off between complexity of the solution and bandwidth consumption, given the target network environment of relatively small ad-hoc networks.

In multi-hop ad-hoc networks messages can be distributed to a large set of users with multicast. The only requirement in our framework is the presence of an underlying multicast ad-hoc routing protocol. All the nodes interested in a particular service register to its well-known multicast address (e.g., for SIP it is 224.0.1.75) to receive messages. Multicast can also be deployed in smaller link-local ad-hoc networks; in this case, however, broadcasting messages is still a preferable solution due to its simplicity. In link-local networks, all the nodes receive any messages sent, regardless of the fact that it is unicast or multicast. The IP address determines which nodes will process it at higher layers. Keeping the solution simple is especially important in networks that may be formed by small devices, which have limited computing capabilities.

3.2. The SESSI authentication and authorization module

The SESSI Authentication and Authorization (AA) Module has two main functions. First, it provides a credential repository where all authentication and authorization credentials and public cryptographic keys are stored in a hierarchical structure. Second, it offers a two-way access control mechanism for managing services that can be offered to a user or services that can be provided by the user. Every device in a SESSI enabled ad-hoc network has its own AA module.

In the AA module, users are authenticated by a base public key. An external trusted third party can provide a certificate (here called the base certificate) that binds the base public key to the user's identity. However, this is not required: the base certificate can also be self-signed by the user. This enables services that do not require authentication of the user but only verify the user's authorization because the user can create arbitrary base certificates. In addition, the AA module supports groups that are handled similarly to users when making access control decisions or storing service specific credentials. Every group has a base public key and a base certificate that is defined by a group administrator.

Because services require different kinds of credentials, the AA module allows the use of application specific certificates and keys. They are bound to a base certificate of a user as Fig. 2 illustrates. The certificates are presented by the user to whom they belong and the user must digitally sign the application specific certificates with the base key. Also, symmetric cryptographic keys can be used, and they are handled similarly to public keys. This way the credentials form a hierarchy where the base certificate is the root and application specific certificates are the leafs. Applications can identify the used key since the certificates contain application specific information such as the security parameter indexes of SLP or the SIP user names.

The AA module offers a similar access control mechanism as Bluetooth by defining three levels of security: no security required, authorization or authentication required, and authorization and confidentiality required. In addition to these three levels, the AA module has a device specific security level. It can be used when a user enters an untrustworthy environment and wants a higher security for all the services. The user changes the device specific level to higher one, and lower service specific access levels are overruled.

However, an ad-hoc network environment requires modifications to the traditional access control. In a peer-to-peer environment where all nodes offer services to each other, in addition to checking that a user is allowed to use a service, the user must check that the service provider can offer the service to his device. In the AA module, the access control entries have six fields: whether the user acts as himself or herself or as the member of a specific group, subject base key, service ID, security level required, validity period, and use or serve. The last field indicates if the service is used by the subject or if it is offered by the subject.

The AA Module consists of five internal parts as is shown in Fig. 3. The credentials are stored in an SQL database. On top of the database, the AA manager handles the access control lists and provides functions to manage user credentials and actions like signing, verifying and encrypting the messages. A separate Crypto Service Module provides the cryptographic functions used in the AA Module. It is based on the OpenSSL library[3]. The service location and session management modules, as well as other SESSI-enabled services and applications, can use the AA module through an API. For the user, the AA module provides a graphical user interface that can be used to add new credentials or to modify already existing ones.

[3] http://www.openssl.org/

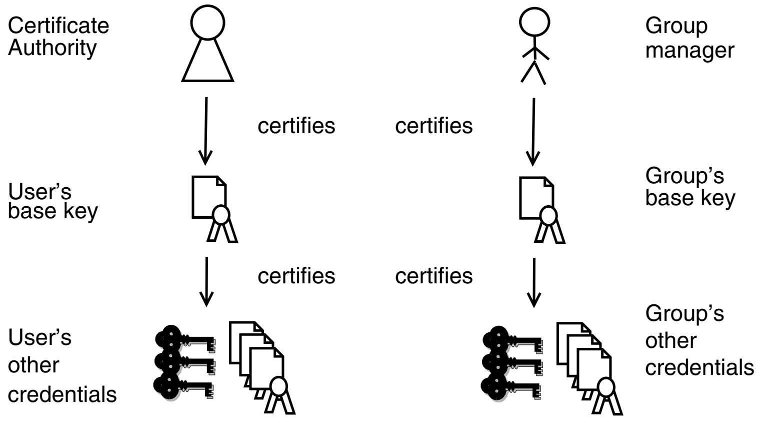

Fig. 2. A user or group certifies its application-specific keys with its base key.

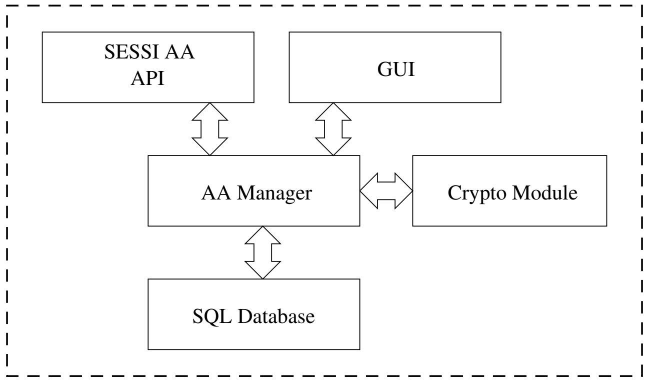

Fig. 3. Software architecture for the AA module.

3.3. SESSISLP

In this section, we give an overview of the Service Location Protocol, while we present modifications made to it in Section 4.1. The SLP protocol is based on three types of entities, which are listed below.

- User agent (UA) is an element that can be associated with a user of a service.
- Directory agent (DA) is an element that acts as a registry from which addresses of different services can be queried.
- Service agent (SA) is an element that provides actual services to users.

Of these, user agents can query for services, and directory and service agents can advertise them and respond to UAs queries. SAs may also register services to DAs.

As already discussed in Section 2.2, the use of a centralized registry is not feasible for an ad-hoc environment. Therefore, the DA element was eliminated from the architecture, as if one would be included, all nodes would be bound to acquire information regarding services from it [1]. Instead, due to the dynamic environment of our framework, we wanted that service providers advertise services themselves. A similar yet ad-hoc enabled solution is to combine a UA and SA in a node, and let the node advertise services of other nodes as well. However, this option was not considered any further. In our future research, we plan to consider an option where such a special node would be used to provide Internet connectivity to the other nodes in the ad-hoc network.

Another issue that needs reconsideration is security. In plain SLP, it is possible to authenticate the SA which originally created a URL that expresses the location of a service or an attribute list of a service. However, there are no feasible means to ensure the integrity protection of entire messages. The SLP specification [1] suggests the use of Encapsulating Security Payload (ESP) to provide encryption, but due to complexity of IPSec [30], which is utilized in this approach, we have chosen not to use it. In plain SLP each signature is accompanied by a timestamp, which marks the time the authentication expires and a security parameter index (SPI). Timestamps are calculated in seconds and are not suitable for ad-hoc environments, as they would require clock synchronization among all the ad-hoc nodes. The SPIs are used to pass information from SA to UA about the keying material, key length, and the algorithm needed for signature verification.

In plain SLP the security features related to URLs and attribute lists are encapsulated in the SLP authentication block illustrated in Fig. 4.

For the ad-hoc environment, this leaves the following challenges to be tackled:

- No means to connect more advanced security properties to a service,
- sufficient integrity protection against tampering cannot be established with original signatures,
- no authentication of UA to SA,
- no available clock synchronization required by the original timestamping,
- no secured means to bind messages to a certain request-reply message sequence.

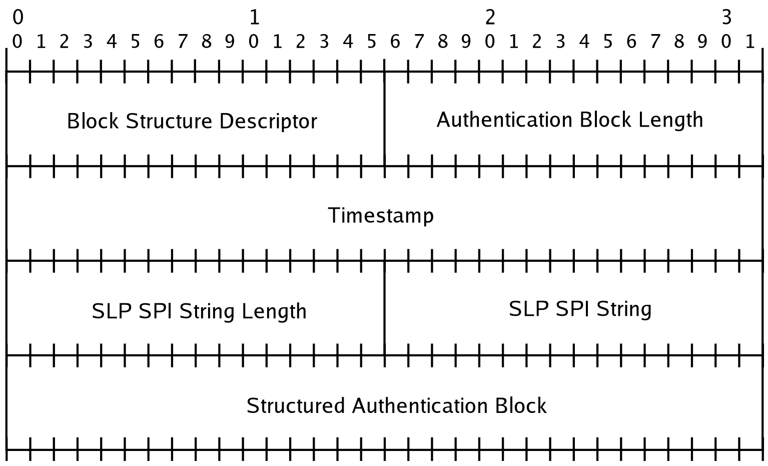

Fig. 4. Standard SLP authentication block.

In order to provide an answer to the above challenges, integration between service discovery and security features are needed. A detailed description can be found in Section 4.1.

The architecture of the service discovery subsystem is illustrated in Fig. 5. The generic SD API provides an interface for external applications and enables changing of the used service discovery protocol. The API is in fact non-SLP specific. The API uses the Locator module for service searching and the Advertiser module for service advertising. The Locator and Advertiser interact with each other through sockets to allow timestamp caching while relying on the SESSI AA API to provide the security features.

3.4. Decentralized SIP

The solution we envisaged for making SIP decentralized is to add to each end node limited SIP proxy and registrar server capabilities. The end node exploits SIP client features as well in the same device. All the end nodes have the functionalities for registering their contact information in the ad-hoc network and acquiring that of the other users in the network. This approach enables decentralized SIP sessions. We refer to this solution as *decentralized SIP* or dSIP. The main design goal of dSIP is that native SIP application can be used in ad-hoc networks as well without need for modification. A more detailed description of dSIP can be found in [31].

SIP operations in ad-hoc networks consist of two steps: user discovery or registration, and session management. User discovery refers to the necessity of acquiring the Address of Records (AOR) of SIP users in the network, together with the IP addresses where they can be contacted. The association between a SIP user name, the AOR, and the contact IP address is called binding. Discovering the AOR of users in an ad-hoc network is important as usually there is no previous knowledge of the users that are online. In infrastructured networks the information on connected users is provided by servers; in ad-hoc networks, the list of online users must be retrieved by the end nodes themselves.

After gaining connectivity in the ad-hoc network, nodes can register by broadcasting (or multicasting) a SIP REGISTER message. The REGISTER message contains the user name and the contact IP address of the registering user. The nodes that receive the message store the binding of the registering user and can send a reply specifically addressed to the registering user, since they know his contact IP address from the REGISTER. The decision on whether to reply to a REGISTER message is based on local policies; for example, a user may wish to reply only to registration messages sent by users in his contact list. The reply message, a SIP 200 OK response, contains the AOR and the IP address of the replying user and allows the registering node to retrieve the bindings of the replying nodes.

Session management procedures follow the standard SIP approach, where an inviting user agent forwards its messages to a proxy server, which will take care of

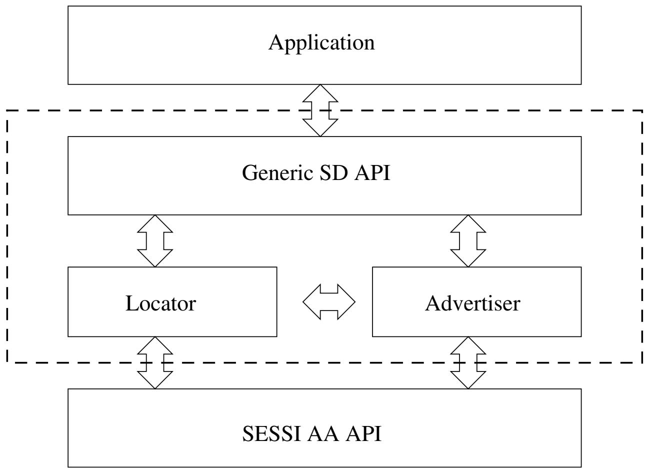

Fig. 5. Modules used in service discovery.

forwarding them to the right recipient. The only difference in dSIP is that the outbound proxy server is colocated in the same node as the user agent. The list of available ad-hoc users is stored at the server. Since the user agent needs the AOR of the contacting user to successfully begin a SIP session, we have enabled the possibility for the user agent to request the list of available users from the local server. This exchange, and in general all communication between user agent and local server, is done using SIP messages. Since user agent and local server are not bound by any function call, we obtain independence between the two modules.

A decentralized SIP node has two main modules: the user agent module, with the functionalities of a baseline SIP user agent enhanced for operations in ad-hoc network, and the server module. The two modules are independent of each other, and are logically unaware of the fact that they are running on the same node; they communicate only with SIP messages sent through the loopback address. In other words, the server module acts as an outbound proxy and registrar server for the user agent module.

This scheme extends but does not substitute baseline SIP functionalities; a decentralized SIP enabled node can be used as a standard SIP user agent for operations in infrastructured networks. In this case, the communication between the user agent and the external proxy server is carried out simply by sending messages to the address of the external server, rather than to the loopback address.

Fig. 6 shows the modules that form the architecture of decentralized SIP. The shaded modules are those that would only be present in a standard SIP node. The low level SIP library provides basic SIP functionalities like message parsing or syntax checks. This library is utilized by the two upper modules, the user agent and the server modules, whose features have already been mentioned.

The cache is used by the server to store information about the users currently in the ad-hoc network. The session management (SM) API is a standard SIP API that any application can use to exploit SIP user agent functionalities. The SM API is further enhanced for enabling operations in ad-hoc networks. For example, the SM API, based on whether the operations are in ad-hoc networks or not, passes to the user agent module the correct address of the registrar/proxy server to use. In ad-hoc networks, the SM module instructs the user agent to use the server running at the address 127.0.0.1. Otherwise it gives the IP address of the preconfigured registrar/outbound proxy server. The enhancements made to the SM API allow native SIP applications to be deployed in ad-hoc networks.

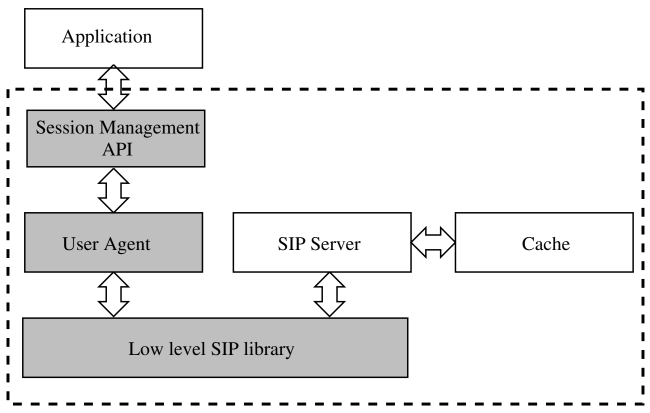

Fig. 6. Software Architecture for decentralized SIP.

4. Seamless service interworking

The previous section presented the three components of our service architecture. In this section, we discuss the interoperability of the components, i.e., how the AA module provides security to the SLP and SIP protocols, and how SLP can be used to distribute the bindings of SIP users.

4.1. Secure service discovery

For the purposes of service discovery, we have identified two new security levels, as defined in Section 3.2, which are authentication and confidentiality. These have been implemented with two SLP application specific key pairs; one is used for authentication, and the other for confidential communication. When aiming at any of these levels, the user is assumed to distribute his public keys before authentication or confidential communication. To encapsulate support for this in SLP messages, the extensions presented in Fig. 7 have been introduced.

To answer to the security challenges discussed in Section 2.2, we have defined the following solutions. To establish a connection between a single service and more advanced security properties, we use the abstract part of the original SLP service type as a security identifier. As a consequence, service type requests were discarded, as they fit poorly with the new security scheme. Problems related to authentication and integrity were solved by replacing the original SLP authentication block (Fig. 4) with an improved structure (Fig. 8) with the following contents: first, the SPI is replaced with the sender's and group's identifiers. Second, the original authentication block field is replaced by a signature calculated over the entire message including the receiver's identifier. Third, a real-time time stamping system was replaced with logical timestamps, where each node caches a number of messages from different peers and uses their logical timestamps to determine whether a received message is new or replicated. The final issue, secured means to bind messages to sequences, was implemented by adding the receiver's identifier to each message addressed to a particular node. This, together with a XID number, that is the same for each request-reply pair, and message integrity protection results in a straightforward yet improved way for clients to manage sequences. These security enhancements are also discussed in [32].

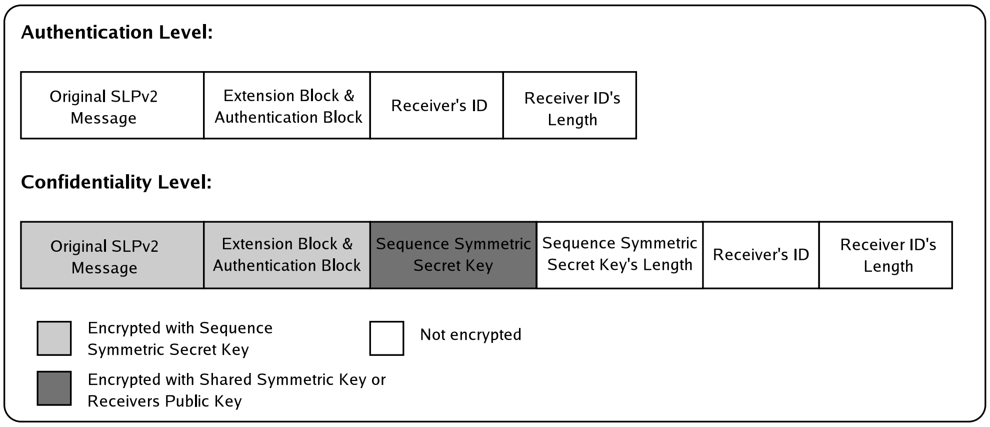
Authentication Level:
Original SLPv2 Message
Extension Block & Authentication Block
Receiver's ID
Receiver ID's Length
Confidentiality Level:
Original SLPv2 Message
Extension Block & Authentication Block
Sequence Symmetric Secret Key
Sequence Symmetric Secret Key's Length
Receiver's ID
Receiver ID's Length
Encrypted with Sequence Symmetric Secret Key
Not encrypted
Encrypted with Shared Symmetric Key or Receivers Public Key

Fig. 7. Message structures at authentication and confidentiality levels.

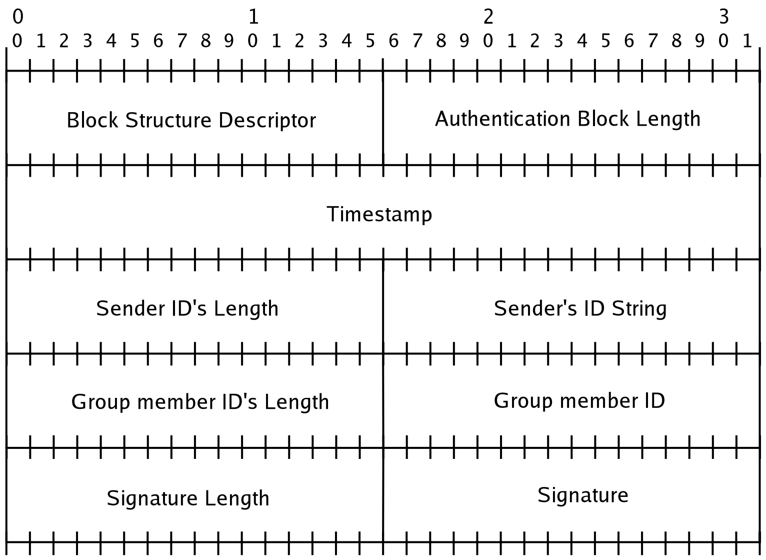

Fig. 8. Improved SLP authentication block.

4.2. SIP user discovery with SLP

Section 3.4 described how decentralized SIP can be used in ad-hoc networks to perform user discovery. An alternative approach for user discovery leverages the presence of a service discovery framework. The interworking between decentralized SIP and SLP in order to discover users' identities in the ad-hoc network is therefore natural. This approach is referred to as SLP-aided SIP, or sSIP.

Instead of broadcasting a SIP REGISTER message upon connection to the ad-hoc network, in sSIP a device uses SLP to query for all the nodes running the service *SIP* and the service attribute *AOR*. All the nodes that have locally registered the service SIP with SLP reply by sending the address where the requested service type is available, that is, their IP address and the user's AOR as SLP service attribute. The reply from the other nodes allows the server module to construct the bindings of the users in the network. Once the bindings have been retrieved, the operations of dSIP and sSIP coincide, and session management is handled in a similar way.

User discovery with SLP has some advantages over decentralized SIP: the most important is that SLP consumes less bandwidth than SIP, since its messages are binary encoded. Section 5 provides an analysis of the bandwidth consumption caused by the two approaches. A disadvantage of using a service discovery framework lies in interoperability issues. If all the devices in the ad-hoc network use the same service discovery framework, then it is possible to retrieve the bindings of all the users; otherwise, if devices use different and incompatible service discovery frameworks, retrieval of users' bindings may be impossible.

A major difference between dSIP and sSIP is that with broadcast REGISTER, receiving nodes can store the bindings of the registering user. When an SLP query for the service "SIP" is received, nodes reply but cannot store the bindings of the registering user. This implies that when SLP is used, new nodes in the network get to know the bindings of older nodes, but the reverse does not hold.

There are two solutions for this problem: one is to refresh periodically the broadcast SLP query for the service "SIP", so that if new nodes have arrived in the interval between two refreshes they can communicate the bindings. Another option is enabling "SLP Passive Service Discovery". This is an extension to the baseline SLP protocol, which allows nodes entering the network to advertise their service, by broadcasting an SLP message that contains the bindings upon connection to the network. The message contains the type of message advertised ("SIP" in this case), the address where it is reachable and the user's AOR as attributes. The scheme is called passive, as nodes passively receive information about active services in the network directly from the service providers. Passive service discovery is a possible subject of future work.

4.3. Secure SIP

The SIP specification, and its extensions, take several security mechanisms into consideration. Some of these are not applicable to the ad-hoc environment. First of all, mechanisms based solely on pairwise shared symmetric keys are not applicable since they do not support broadcast messages. This rules out the otherwise popular Digest authentication [33] mechanism. Further, Transport Layer Security (TLS) [34] cannot be used since it relies on TCP, which does not support the broadcast messages that dSIP uses. S/MIME [35] is a more feasible solution. A major drawback of S/MIME is that, in order to protect the header's integrity, the message is tunneled, which remarkably increases its size. It also increases the risk for message fragmentation at lower layers. In our framework, where SIP runs over UDP, fragmentation of SIP messages should be avoided, as reconstructing the messages adds unnecessary complexity.

We have selected a new proposal, Authenticated Identity [36], where message integrity and authenticity is provided in a bandwidth-efficient way by adding an asymmetric signature to the SIP header. We have modified this scheme to fit our framework. The Authenticated Identity draft allows UAs to receive SIP requests from entities with which they have no previous association, and to verify the identity of the calling user. An identity is defined in this context as the user name or the AOR of a SIP user.

In this proposal, the sender's proxy server first authenticates the sender. It then creates a signature over a set of pre-defined SIP header fields and the body of the request. The signature is added in a new header field. Another header field indicates a secure URL where the domain certificate of the proxy server is available. The only addition to the SIP message are the two new header fields. Since the signature is smaller than 200 bytes, the integrity of the message is protected with a reasonable overhead and within each message.

The ideas presented [36] have been modified to fit the SESSI framework. Each user has an asymmetric key pair – a SIP application specific key – and uses it to sign his or her own messages. This choice is mandated by the fact that in ad-hoc networks there are no domain servers that would authenticate users and sign messages on their behalf. The SESSI framework currently assumes that users willing to engage in secure communication share each other's security information; in this case, the users possess each other's public key certificates. The keys are stored and managed with the AA module. The AA module is also used for access control and security level decisions, as described earlier in this paper.

Since all the authorized users possess the corresponding certificate, they can verify the authenticity of the messages. With this approach, nodes can sign the broadcast REGISTER, receive and verify signed 200 OK messages, and ignore messages that do not contain the authentication string. If a non-signed REGISTER is received, and the security level for that node requires authentication, then the node can reply with a targeted signed REGISTER to the node and wait for the 200 OK to store the binding of the user. Similarly, INVITE messages can be protected for integrity.

The Identity mechanism does not guarantee the confidentiality of data exchange. One solution for confidentiality could be TLS, which is not applicable to our framework. Using S/MIME for confidentiality is a better solution, but presents the problem of noticeably increasing the size of exchanged messages. Moreover, implementing S/MIME in the end nodes would add complexity and may not be suitable for small devices. Within the constraints of our framework, we do not deem worthwhile to make the SIP signaling confidential, as the trade-off between added complexity and overhead and received benefits bends to the side of complexity. However, we plan to build applications based on SIP, namely add the support of SIP based Instant Messaging and Presence services. This will require that the confidentiality of exchanged user data messages is ensured. In this case, exploitation of S/MIME encryption may be necessary.

In order to provide confidentiality to SIP signaling messages, a new security mechanism, called datagram TLS (dTLS) [37], could be used as well. dTLS enables TLS-like protection at transport layer over a datagram protocol, like UDP. If SIP would run over dTLS, confidentiality of messages would be achieved without excessive overhead, and at the same time broadcast operations would be supported. However, the proposal is very new and its interworking with SIP has not been completely defined. An Internet Draft [38], which is still an individual submission to the SIP Working Group, defines how SIP can run over dTLS. A mature proposal seems still far away. Analyzing the integration of dSIP with dTLS is a possible subject for future work.

When the user discovery is performed with the help of the SLP protocol, the security of this phase is naturally provided with the enhanced SLP security mechanisms, described in Section 4.1. The Authenticated Identity mechanism is still used to protect the later phases in session management.

5. Evaluation of the implementation

This section presents a performance evaluation of our implementation. The focus is on analyzing the overhead that SLP and SIP protocols introduce when looking for services (namely, the service SIP) or registering in the network with different levels of security. The analysis is based on an estimate of the number of messages sent over the air and their size in link-local ad-hoc networks of varying size. The processing load introduced in the ad-hoc device by the security module is analyzed as well.

5.1. Prototype implementation

The described framework has been implemented and tested under Linux. Four laptops were used to form a link-local ad-hoc network running with an IEEE 802.11 WLAN in ad-hoc mode. The design goals, in terms of functionalities of the single services and their interoperability, were successfully met.

Besides the infrastructural services, we also implemented a chat application as an example of a generic service running on top of the infrastructural services. The service discovery module was utilized for discovering the other users in the network and consequently begin the chat session. When the security level was enforced, only authorized users could join the chat channel of a user; if confidentiality was requested, the messages were encrypted.

Interoperability between SLP and SIP was successfully tested as well; the SIP server module first retrieved the other users' bindings with SLP, and after this a chat session was established using the SIP INVITE-based model. Decentralized SIP has proved to be successful too; all users in the network were successfully discovered using the REGISTER – 200 OK exchange and chat sessions were initiated between them. When authentication was requested, the 200 OK messages were returned only to authorized users. Unauthorized users did not receive the contact information from the nodes that did not grant authorization, based on the certificates and rules stored in the AA module.

5.2. Comparison between dSIP and sSIP

Retrieving the bindings of SIP users with SIP-only methods, as discussed in Section 3.4 is complementary to retrieving them using SLP, described in Section 4.2. We carried out a comparison of the overheads due to the two approaches.

When a user registers in an ad-hoc network where there are already $N-1$ nodes, a broadcast SIP REGISTER message is sent and up to $N-1$ SIP 200 OK responses are received. When SLP is used, the registering node sends an SLP broadcast Service Request message, to which up to $N-1$ unicast SLP Service Reply messages are returned. After waiting for a predetermined time, the registering node issues a broadcast SLP Attribute Request query, for retrieving, as service attributes, the AORs associated to the discovered service addresses. Again, up to $N-1$ unicast responses are returned. We consider the worst case, from the bandwidth consumption point of view, that all the other $N-1$ nodes reply to a registering user when computing the registration overhead.

Table 1 shows typical SIP and SLP messages sizes at transport level, in the various defined security levels. The size of the message depends on the carried payload, like, e.g., user or service name; we have measured the sizes with different payload contents, and taken the sizes (bytes) reported in the table as average reference values.

In the rest of the discussion, we assume that in SLP there is a single exchange of broadcast messages and unicast replies, where the message size is the sum of the two messages used in each phase, respectively. The value 300 bytes in the broadcast section for SLP integrity (or authentication) level messages, means that the average sum of a broadcast Service Request and Attribute Request is 300 bytes. The same reasoning applies to the other SLP entries of the table. SIP messages are not protected for confidentiality because with S/MIME, which is the most suitable method in our framework for providing confidentiality for SIP, SIP messages would become too big. The bandwidth consumption, and processing load as well, were considered too high compared to the benefits that encryption of SIP signaling messages would produce.

The overhead of a node registration with both approaches increases linearly with the numbers of nodes in the network. Fig. 9 shows the bytes sent over the wireless link to register an incoming node and finding other users in the network, with dSIP and sSIP when no security protection is used, in a worst case scenario where all the other $N-1$ nodes send a unicast reply to the broadcast message. In this and in the following figures, dSIP data is indicated by a thick solid line, while sSIP data is indicated by a thin solid line. The increase is linear because if the $N+1$th node joins the network, 1 broadcast message and N unicast messages are exchanged in the network. So, the increase in bytes is proportional to the average size of a unicast reply, about 310 bytes for dSIP and 110 bytes altogether for the two SLP replies. Expectedly, dSIP is more bandwidth aggressive than sSIP, as SIP messages are bigger than SLP messages. Fig. 9 shows that registering, e.g., the 50th user with dSIP consumes as much bandwidth as registering the 140th user with sSIP, about 15 kbyte. However, we must bear in mind that a network of 50 nodes is already quite close to the scope of our target network environment. For smaller networks, the overhead of dSIP is still quite close to the sSIP counterpart.

Fig. 10 shows the bytes sent altogether to register N users. The figure refers to the worst case scenario, with security level of none. For every registration a number of

Table 1
SIP and SLP typical message sizes

	None		Integrity		Confidentiality	
	Broadcast	Unicast	Broadcast	Unicast	Broadcast	Unicast
SIP	330	310	570	550	–	–
SLP	110	110	300	290	480	460

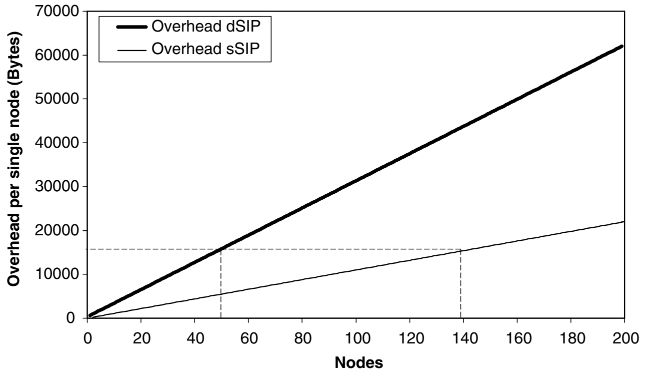

Fig. 9. Overhead of a single registration. Security level none.

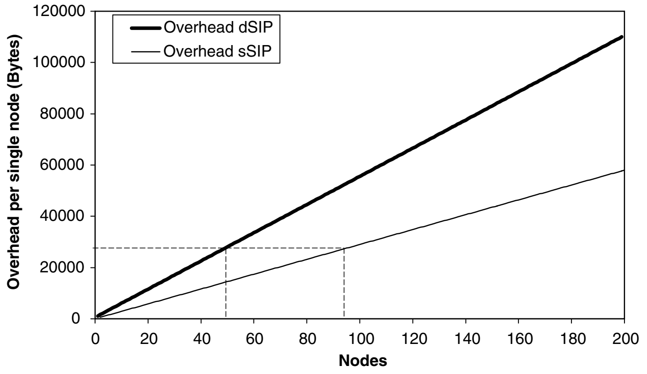

Fig. 11. Overhead of a single registration. Security level integrity.

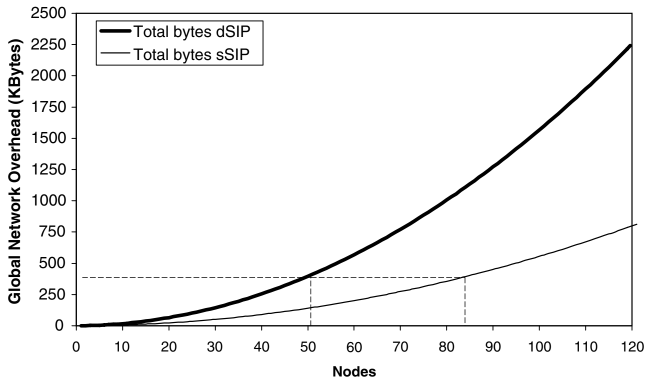

Fig. 10. Global registration overhead. Security level none.

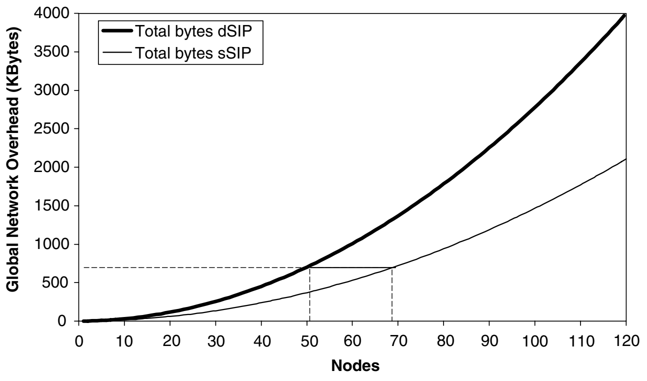

Fig. 12. Global registration overhead. Security level integrity.

messages equal to the number of users N in the network are sent; that is, one broadcast and $N-1$ unicast replies. Consequently, the total number of registration related messages sent in the ad-hoc network after that the Nth user has registered is:

$$\sum_{k=1}^{N} k = \frac{N(N+1)}{2}. \qquad (1)$$

The total registration related bytes sent to register 50 users with dSIP, a little less than 400 kbyte, correspond to the amount of data needed with sSIP to register 85 users.

Figs. 11 and 12 show the same data when security level is integrity. Both SIP and SLP messages used in this case are bigger. SIP messages have the Identity header fields, which add a overhead of 240 bytes compared to the situation of no authentication; SLP messages carry the additional security block and the signature. Although SIP messages remain bigger, the overhead introduced by the Identity mechanism is lower than its SLP counterpart. The main design goal for secure SIP was in fact to provide integrity with minimal added overhead. Particularly, Fig. 11 shows that the overhead of a single registration in dSIP for the 50th node is about 27 kbyte, approximately equal to the overhead of sSIP for a 95 node network. We recall that when no security mechanisms were used, a dSIP network of 50 nodes generated an overhead similar to an sSIP network of 140 nodes.

In terms of global registration overhead, it is possible to see from Fig. 12 that after the 50th user has registered with dSIP, a total of about 700 kbyte of data has been exchanged in the network. This amount corresponds to the registration of 69 nodes with sSIP. When no security mechanisms are used, the equivalence was reached for an sSIP network of 85 nodes.

5.3. Discussion of dSIP and sSIP

The choice of the better option between dSIP and sSIP cannot be based only on the basis of bandwidth requirements. One advantage of dSIP over sSIP is the interoperability between different solutions, as discussed in Section 4.2. SIP functionalities are mandatory to implement if session based applications are to be deployed; therefore, using SIP for user discovery operations as well is reasonable. On the other hand, in ad-hoc networks a service discovery protocol is often necessary to discover what services are available; even if dSIP is used, an ad-hoc node may need to use a service discovery protocol to acquire information about the availability of other, non-SIP, services in the network. If bandwidth limitations are an issue, sSIP can thus be chosen instead of dSIP for SIP user discovery.

Another point that favors the use of dSIP over sSIP is the availability of the bindings at each node. With dSIP, with a single message handshake the bindings are exchanged between the newcomer and the other nodes in the ad-hoc network; with SLP, this is not possible, as the bindings cannot be retrieved from an SLP request. In an SLP query, the AOR of the querying user is in fact not passed to the other nodes. With sSIP, broadcast SLP requests for users in the network can be periodically refreshed; this allows the broadcasting node to receive replies from newcomers. However, a node would have to wait for the expiration of its registration refresh timeout to discover the newcomers' identities.

If the user or local policies require the confidentiality security level for the user discovery phase, dSIP cannot be used because protecting dSIP messages for confidentiality consumes too much bandwidth. The solution that would ensure confidentiality at the price of reasonable bandwidth consumption is sSIP.

In summary, the two alternative solutions can and should coexist; the user could be prompted to choose which user discovery mechanism he wants to use. The choice depends, among others, on the operational environment, on the application using the underlying SESSI infrastructural services and on the security level requested.

5.4. AA module

The device-local authentication and authorization mechanism is fully distributed; security decisions are made locally in the user's device. However, this flexibility comes with a trade-off in terms of storage and processing requirements. To evaluate the feasibility of the solution, we measured the performance of the AA module and estimated the processing due to dSIP and sSIP at the authentication level. The tests were run on a laptop with a 1.8 GHz CPU. For the test, 9000 access control rules and 300 users were stored in the module. For each user, five application-specific certificates with 1024-bit keys were stored. Further, we assume that even though each node sends two reply messages with sSIP, the device does not check a user's access rights twice but only when the first message arrives.

First of all, the measurements showed that a mobile device can easily store credentials, information and access control rules even for thousands of users. An AA module with 30,000 access control rules and 1000 stored users, each with 10 application-specific certificates, takes about 16MB of storage space. Such a database size is more than enough for most users and can be stored on almost any smartphone on the market. Typically, in fact, a user may have the contacts of a few hundred other users, each with 1–5 application-specific certificates, and around a few hundred access control rules; in this case, 1.5MB are sufficient.

Fig. 13 shows the AA processing load on a registering node using dSIP and sSIP on the authentication level. The figure refers to a worst case scenario, from the computational load point of view, when all the nodes in the network decide to reply to the broadcast registration message. The processing load is caused mainly by incoming messages. All nodes sign their outgoing messages. When the registering node receives a message, it must find the user in the database, execute access control operations, and verify the signature. In the test environment, the dSIP registration phase took about 10 s or a little less than 20×10^9 CPU cycles when there were 20 nodes in the network. sSIP was somewhat less efficient since each registration causes two message exchanges with each node. Then, 14 nodes could participate in the registration phase with the same overhead of 10 seconds. For an old node, the AA processing due to the newcomer's registration was 0.97×10^9 CPU cycles or 0.54 s with dSIP, and 1.44×10^9 CPU cycles or 0.8 s with sSIP. In summary, registration operations pose a noticeable burden only on the registering node; for the older nodes, external registrations are practically transparent from the CPU consumption point of view, which is a desirable property.

The total AA processing load for a registering node was dominated by incoming messages. The time spent in creating and verifying signatures was negligible: an optimized implementation creates or verifies RSA and DSA signatures with 1024 bit keys in a few hundredths of a second.

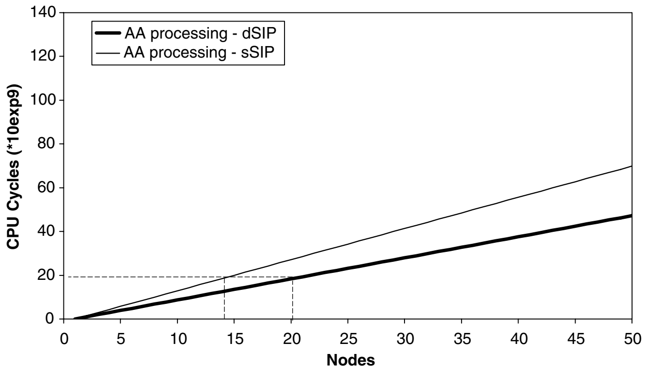

Fig. 13. AA-related processing load of a single registration in dSIP and sSIP. Security level authentication.

Most of the processing was caused by mapping user names to base users and retrieving certificates from the database.

Although the time consumption is quite large, these results are within tolerable limits. For instance, authenticating one device with Bluetooth takes several seconds, and still Bluetooth has gained wide consumer acceptance. Certificate search can also be speeded up several times by redesigning database indexing and using a more sophisticated search algorithm. To further speed up the search, the certificate repository can be read into memory to avoid slow file access from the hard disk. On handheld terminals, memory cards with fast read access are usually used for file storage and file access is faster. Optimized cryptographic implementations will cut the signature creation and verification time as well.

The test environment CPU was multiple times faster than CPUs used on current-day PDAs. However, keeping in mind that handheld terminals are fast becoming more powerful, the time overhead should also be tolerable on a mobile terminal. In any case, protecting messages for security increases the processing load in the device and the message transmission time. Since ad-hoc devices will always be powered by batteries, it is advisable to perform secure communication only when needed or when battery resources allow it.

6. Conclusions and future work

The presented architectural framework enables the secure and dynamic use of services in wireless ad-hoc networks. It has three foundations: device-local management and enforcement of authentication and authorization, secure discovery of services, and secured session management. These three components are present in the secure use of any service, and as they traditionally use server-based architectures, they have formed a barrier for smooth service deployment in networks without special infrastructure. Our distributed framework is especially designed for infrastructureless ad-hoc networks. It can also be extended to support interworking with external infrastructured networks.

Ad-hoc networks are formed by the users that join the network. Our framework therefore deploys a user-rooted public key infrastructure for authentication. Users that have met once can easily authenticate each other, even if a user's application-specific authentication material is renewed in between. The scheme is fully decentralized in that all access control rules are evaluated locally in each device, and all authentication-related cryptographic processing is performed locally. Nevertheless, a user must obtain the other users' certificates and verify them in advance. This phase is eased by a certificate exchange service that we plan to add next. Then, the user only needs to carry the base certificates of other users, or alternatively, also retrieve these in the ad-hoc network and verify them personally. Tests should still be carried out to verify the usability and intuitiveness of the security management scheme. Computational power and battery form a more worrisome bottleneck. Authentication is handled commonly for several services, and so every signing and verification procedure uses a large common certificate repository. The overall performance is yet within tolerable limits especially in small link-local ad-hoc networks.

The service discovery scheme described in this paper provides a secure way to handle distributed service discovery in ad-hoc networks. The distribution was achieved by integrating the user agent and service agent functionality to one node and discarding the directory agents from the scheme. On the other hand, establishing security required interworking of the AA and the SD modules. The service was based on the Service Location Protocol, which was extended to support the three common security levels defined for the framework. Future work is needed to enable seamless and secure service discovery between ad-hoc and infrastructured networks. Extension of SLP features for enabling passive service discovery is another possible target for future research.

Distributed session management was achieved by adapting the Session Initiation Protocol (SIP) to a distributed approach. The scheme allows for discovering SIP users in an ad-hoc network and for establishing media sessions with them in an entirely decentralized fashion. Session management operations have been protected for integrity using a self-signed certificate based mechanism; the AA module was used for managing the security associations among users. A SIP application layer can be implemented on top of the middleware signaling layer; the IETF SIMPLE working group is defining extensions to the baseline SIP protocol for leveraging instant messaging and presence applications. Enhancing decentralized SIP with SIMPLE functionalities is also a possible development direction. Finally, a SIP proposal, based on the SUBSCRIBE/NOTIFY framework [39], seems a promising starting point for the exchange of certificates on-line.

The most important future common development will be a gateway that provides connectivity between the ad-hoc network and an external infrastructured network. Since globally routable IP addresses are not always available to the ad-hoc network, the gateway could act as a Network Address Translator (NAT) for all the nodes in the ad-hoc network. This gateway will act not only at the routing level, but it will also provide additional services like the possibility of discovering services inside and outside the ad-hoc network. The gateway will then operate as a special node that hosts a service discovery proxy: the proxy should store both service information and security credentials from both networks. A SIP proxy will also be implemented in the gateway node. The SIP proxy facilitates session establishment procedures and registrations with nodes outside the ad-hoc network; a major feature of the proxy will be to allow external nodes to contact nodes in the ad-hoc network, even though no global IP addresses are used in the ad-hoc network. The operations of gateway discovery and the utilization of services will be protected, by means of the integrations of the service discovery and session management modules with the extended SESSI security mechanisms. However, the use of NATs imposes changes to the working mode of the protocols and services run in the network; moreover, NAT-based solutions are often protocol or service specific. The final aim is to build a framework for seamless service interworking between ad-hoc and infrastructured networks.

Acknowledgments

This work is part of the SESSI project (2644/31/03), funded by the National Technology Agency of Finland, Elisa Corporation, the Finnet Group, and Nokia Corporation. The authors thank all the project members for their comments and contributions to the development of the project and the paper.

References

[1] E. Guttman, C. Perkins, J. Veizades, M. Day, Service location protocol, version 2, RFC 2608, IETF (June 1999).
[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, SIP: session initiation protocol, Request for Comments 3261, IETF (June 2002).
[3] F. Stajano, R. Anderson, The resurrecting duckling: Security issues for ad-hoc wireless networks, in: Proceeding of the 7th International Workshop on Security Protocols, Lecture Notes in Computer Science, Springer-Verlag, 1999.
[4] N.J. Muller, Bluetooth Demystified, McGraw-Hill, 2000.
[5] W. Shunman, T. Ran, W. Yue, Z. Ji, WLAN and its security problems, in: Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT2003), 2003, pp. 241–244.
[6] D. Atkins, W. Stallings, P. Zimmermann, PGP message exchange formats, RFC 1991, IETF (August 1996).
[7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen, SPKI certificate theory, RFC 2693, IETF (September 1999).
[8] M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis, The KeyNote trust-management system version 2, RFC 2704, IETF (September 1999).
[9] P. Fenkam, S. Dustdar, E. Kirda, G. Reif, H. Gall, Towards an access control system for mobile peer-to-peer collaborative environments, in: IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), IEEE, 2002.
[10] J. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, A. Joshi, A secure infrastructure for service discovery and access in pervasive computing, Mobile Networks and Applications 8(2) (2003), pp. 113–125.
[11] S. Liimatainen, Usability of decentralized authorization systems - a comparative study, in: Proceedings of the Thirty-Eighth Annual Hawaii International Conference on System Sciences, 2005.
[12] UPnP Forum, UPnP Device Architecture version 1.0.1 (December 2003).
[13] The World Wide Web Consortium, Web services activity, Available online: http://www.w3.org/2002/ws/, 01-May-2006.
[14] L. Gong, Project JXTA: A Technology Overview, Sun Microsystems Inc.
[15] ITU-T, Packet-based multimedia communications systems. Recommendation H323 version 5, Tech. rep., International Telecommunication Union ITU (July 2003).
[16] J. Glassman, W. Kellerer, H. Muller, Service architectures in H.323 and SIP: a comparison, IEEE Communications Surveys and Tutorials 5 (2).
[17] S. Cheshire, B. Aboba, E. Guttman, Dynamic Configuration of Link-Local IPv4 Addresses, RFC 3927, IETF (May 2005).
[18] S. Thomson, T. Narten, IPv6 stateless address autoconfiguration, Request for Comments 2462, IETF (December 1998).
[19] N. H. Vaidya, Weak duplicate address detection in mobile ad hoc networks, in: MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing 2002, pp. 206–216.
[20] Z. Fan, S. Subramani, An address autoconfiguration protocol for IPv6 hosts in a mobile ad hoc network, Elsevier J. Comput. Commun. 28 (2005) 339–350.
[21] S.E. Czerwinski, B.Y. Zhao, T.D. Hodes, A.D. Joseph, R.H. Katz, An architecture for a secure service discovery service, in: Mobicom '99, ACM, Seattle, Washington, USA, 1999.
[22] D. Bryan, J. C., A P2P approach to SIP registration, Internet-draft (work in progress), IETF, draft-bryan-sipping-p2p-02 (March 2006).
[23] H. Khlifi, A. Agarwal, J. Gregoire, A framework to use SIP in ad-hoc networks, in: Canadian Conference on Electrical and Computer Engineering. IEEE CCECE, vol. 2, 2003, pp. 985–988.
[24] N. Banerjee, A. Acharya, S. K. Das, Peer-to-peer SIP-based services over wireless ad hoc networks, in: BROADWIM: Broadband Wireless Multimedia Workshop, 2004.
[25] F. Zhu, M. Mutka, L. Ni, Facilitating secure ad hoc service discovery in public environments, in: The 27th Annual International Computer Software and Applications Conference (COMPSAC'03), 2003, pp. 433–438.
[26] B. Bhargava, X. Wu, Y. Lu, W. Wang, Integrating heterogeneous wireless technologies: a cellular aided mobile ad-hoc network (CAMA), Mobile Netw. Appl. 9 (2004) 393–408.
[27] S. Capkun, L. Buttyan, J.-P. Hubaux, Self-organized public-key management for mobile ad-hoc networks, IEEE Trans. Mobile Comput. 2 (1) (2003) 52–64.
[28] A. Weimerskirch, G. Thonet, A distributed light-weight authentication model for ad-hoc networks, in: The 4th International Conference on Information Security and Cryptology, Springer-Verlag, Seoul, 2001, pp. 341–354.
[29] C. Tschudin, P. Gunningberg, H. Lundgren, E. Nordstrm, Lessons from experimental MANET research, Elsevier J. Ad-Hoc Netw. 3 (3) (2005) 221–233.
[30] S. Kent, R. Atkinson, Security architecture for the internet protocol, RFC 2401, IETF (November 1998).
[31] S. Leggio, J. Manner, A. Hulkkonen, K. Raatikainen, Session initiation protocol deployment in ad-hoc networks: a decentralized approach, in: 2nd International Workshop on Wireless Ad-hoc Networks (IWWAN), London, May, 2005.
[32] L. Källström, J. Saarinen, Secure service discovery protocol implementation for wireless ad-hoc networks, in: 1st International Wireless Summit, Aalborg (2005) 17-22.
[33] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, HTTP authentication: basic and digest access authentication, RFC 2617, IETF (June 1999).
[34] T. Dierks, C. Allen, The TLS protocol version 1.0, RFC 2246, IETF (January 1999).
[35] B. Ramsdell ed., Secure/multipurpose internet mail extensions (S/MIME) version 3.1 message specification, RFC 3851, IETF (July 2004).
[36] J. Peterson, C. Jennings, Enhancements for authenticated identity management in the session initiation protocol SIP, Internet draft (work in progress), IETF, draft-ietf-sip-identity-06 (October 2005).
[37] N. Modadugu, E. Rescorla, The design and implementation of datagram tls, in: Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA.
[38] C. Jennings, N. Modadugu, Using DTLS as a transport for SIP, Internet draft (work in progress), IETF, (draft-jennings-sip-dtls-02) (March 2006).
[39] C. Jennings, J. Peterson, Certificate management service for SIP, Internet-draft (work in progress), IETF, draft-ietf-sipping-certs-03 (March 2006).

Linda Källström, M. Sc. (Tech) is pursuing her doctoral studies at the Laboratory of Telecommunication Software and Multimedia at Helsinki University of Technology. In the past, she has researched authentication schemes in different types of network environments and social environments. She currently focuses on secure service deployment in ad-hoc networks.

Simone Leggio obtained his Master of Science in Electronic Engineering at University of Catania, Italy, in July 2002, with full honors and his Licentiate Degree at the Department of Computer Science of University of Helsinki. Currently he is working towards his Ph.D. in Computer Science at University of Helsinki. His research interests include IP session management in ad-hoc and mobile networks, instant messaging and presence systems, seamless mobility in IP-based networks, and IP quality of service in mobile environments.

Jukka Manner is a professor of computer science at the Helsinki University of Technology and at the University of Helsinki. He received his MSc. in computer science in 1999 and a PhD. in computer science in 2004 from the University of Helsinki. His research and teaching focuses on mobile and wireless communications, especially QoS and mobility in future generation mobile networks, IP technologies, distributed systems, and operating systems. He has participated in several national research projects (funded by National Technology Agency of Finland and industry) and in several EC projects. He has actively participated in the IETF for many years. He has published over 30 articles in conferences and journals, and several IETF documents, and has served on the TPC of many international conferences.

Prof. Tommi Mikkonen (MSc 1992, Lic. Tech. 1995, Dr. Tech 1999, all from Tampere University of Technology, Tampere Finland) works on distributed and mobile systems at the Institute of Software Systems at Tampere U of Tech. Over the years, he has written a number of research papers on distributed systems and their development as well as supervised a number of thesis on the subject.

Kimmo Raatikainen [M'81] received the Ph.D. degree in computer science in 1990 from the University of Helsinki. Since 1998 he has been a professor at the Helsinki University Computer Science Department leading the group of Distributed Systems and Data Communication. Prof. Raatikainen has c. 100 scientific publications on areas of performance evaluation, simulation methodology, real-time databases, Internet protocols, and middleware. Professor Raatikainen has had a leading role in several European projects. He has also led several national research projects (funded by the National Technology Agency of Finland (TEKES) and industry on mobile computing, wireless communication, middleware for mobile computing and on telecommunications software architectures. His current research interests include operating systems, protocols and middleware for mobile computing.

Jussi Saarinen (MSc 2006 from Tampere University of Technology, Tampere, Finland) is currently working as a research assistant at the Institute of Software Systems at Tampere University of Technology. During the last two years, he has contributed to three research papers on security of service discovery.

Sanna Suoranta, Lic. Sc. (Tech) works as a teaching researcher in the Laboratory of Telecommunication Software and Multimedia at Helsinki University of Technology. She is doing her doctoral thesis on management of digital identities in heteregeneous and decentralized network environments.

Dr. Tech. Antti Ylä-Jääski is a Professor of Telecommunications Software, Telecommunications Software and Multimedia Laboratory, Department of Computer Science and Engineering, Helsinki University of Technology, Finland. He is also a Research Fellow in Network Technologies Laboratory, Nokia Research Center, Helsinki, Finland. Prof Dr. Tech. Antti Ylä-Jääski received his MSc. degree in Helsinki University of Technology, Finland and PhD degree in ETH Zuerich, Switzerland. Antti has worked with Nokia 1994–2004 in several research management positions in Nokia Research Center, Nokia Ventures Organisation and Nokia Networks with focus on future Internet technologies, mobile networks, applications, services, service management and service architectures. He has published over 30 articles and he holds several approved patents. Antti's current research interests include mobile networking, heterogeneous network environments, services, service architectures, service management and security issues.