Securing Passive Objects in Mobile Ad-Hoc Peer-to-Peer Networks

Rene Mayrhofer [a] Florian Ortner [a] Alois Ferscha [a]
Manfred Hechinger [a]

[a] *Institut für Praktische Informatik*
Johannes Kepler Universität Linz
Altenberger Str. 69
A-4040 Linz
Austria

Abstract

Security and privacy in mobile ad-hoc peer-to-peer environments are hard to attain, especially when working with passive objects without own processing power. We introduce a method for integrating such objects into a peer-to-peer environment without infrastructure components while providing a high level of privacy and security for peers interacting with objects. The integration is done by equipping passive objects with public keys, which can be used by peers to validate proxies acting on behalf of the objects. To overcome the problem of limited storage capacity on small embedded objects, ECC keys are used.

1 Introduction

Currently, ad-hoc networks is a highly active research topic with many publications covering different aspects of this inter-disciplinary field (e.g. [15]). These aspects include, but are certainly not limited to, hardware (e.g. size, rugged design, power consumption, communication), software (e.g. operating system/platform, communication protocols, memory usage), interaction (e.g. interaction models, HCI aspects), security and application issues. In this paper, we will focus on privacy and security aspects of ad-hoc, peer-to-peer networks within the SmartInteraction project.

The SmartInteraction project is an approach to interact with persons, things and places in a natural and non-obtrusive way. As for example people meet each other, their "interaction profile" is mutually compared in analogy

[1] This work has been developed in cooperation with Siemens Munich, CT SE2

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

to their natural, automatic choice of sympathy. Following the vastly successful way of human communication and coordination, the Peer-to-Peer (*P2P*) paradigm is used for direct communication among all participating devices. This offers complete device autonomy, independence of central authorities and reliability due to redundancy. Within the SmartInteraction project, this principle is even taken one step further by also being independent of any common communication infrastructure: we utilize solely ad-hoc wireless networks, currently either IEEE802.11b Wireless LAN (*WLAN*) or IEEE802.15.1 Bluetooth (*BT*). To match the flexibility of the P2P approach, local profiles describing the device capabilities, user attributes and preferences are kept on every peer. Upon spatial contact with other peers, these profiles provide the base for matching user interests and determining further, automatic coordination. Additionally, context constraints defined in profiles provide the necessary context awareness for ubiquitous applications; different situations, identified by context parameters, demand different behavior. As in any ubiquitous system, privacy and security are major concerns and are taken seriously by utilizing active and passive privacy control backed by strong cryptography. We do not aim to develop new cryptographic algorithms or novel security protocols, but instead utilize and combine well-known and secure techniques. However, we were unable to find protocols or methods for securely integrating passive objects without own processing capabilities into a P2P infrastructure. As this is an issue in our project, we developed a method to secure remote proxies that act on behalf of passive objects; this is our main contribution in the present work. The other aspects of the SmartInteraction project's framework are only presented as far as necessary to understand the security aspect.

This paper is organized as follows: In section 2, we start by shortly explaining the hard- and software environment the SmartInteraction project is situated in, including our definition of (passive) objects. Section 3 then gives an overview of related work, while section 4 presents our approach to P2P privacy and security between powerful peers. An addition to this approach to securely integrate (passive) objects with powerful peers – the main contribution – is presented in section 5. After that, we give a short conclusion and an outlook on our planned future research in section 6.

2 Environment

In this section, we provide a short overview of the environment in which our work has to take place. The SmartInteraction project aims to provide a flexible framework for ad-hoc, mobile P2P interaction of multiple, heterogeneous devices. A software framework has been developed which already handles many aspects of ad-hoc, P2P interaction and therefore allows the efficient construction of applications in this fast-growing domain. Since this paper focuses on security aspects of the framework, further details concerning the interaction model and our profile approach will be published later. The framework is able

(a) iD-2 (b) i-CARD

Figure 1. Identec RFID hardware

to run on a wide range of platforms (*peers*), the only requirement is a Java 1.1 compatible JVM and arbitrary communication technology. However, we also want to integrate devices without any processing capabilities (*objects*) into the P2P interactions.

2.1 Peers

For complete, instantaneous, ad-hoc P2P interaction, processing capabilities are required on each interaction partner. These so-called *peers* can run our software framework, which allows them to discover and communicate with each other. Possible platforms for peers are standard servers (especially for remote proxy peers, described in the next section), notebooks, sub-notebooks, handhelds, PDAs or even mobile phones. Small, mobile devices will normally have limited resources such as processing power, RAM or storage capacity, but they are nonetheless capable of securing their own communication with strong cryptography (see appendix A).

2.2 Objects

As already mentioned, we do not want to restrict ourselves to only integrating peers in the P2P interactions, but we also want to have *objects* participating. In our environment, we define an *object* in the following way:

> An object is passive in regards to executing custom code, i.e. it does not have processing power that could be exploited to run parts of a custom software. Objects are required to have some kind of identification number (*ID*).

This definition does not prevent objects from having a CPU and carrying out computations. Thus, the following devices are examples for objects in our definition:

- RFID (*Radio Frequency Identification tags*): These are either passive (powered by the reader's radio field) or active (with own power supply), via RF (Radio Frequency) accessible small memory devices, which are available for a broad range of applications from multiple vendors like Identec Solutions, Inside or Texas Instruments. An Identec iD-2 tag and the i-CARD PCMCIA reader are shown in Fig. 1.
- IrDA (*Infrared Data Association*) beacons: These are active devices, periodically sending infrared packets that can be received by any device with a standard IrDA port (e.g. notebooks, PDAs, mobile phones). IrDA beacons have already been deployed on larger scale for various applications (e.g. [17]).
- Bluetooth (IEEE 802.15) devices: There already exist many Bluetooth devices with own processing power which could be peers in our definition. But even Bluetooth devices without own processing power can be used as objects by utilizing their MAC address as unique object ID.

As those devices, especially the RFID tag technology, become smaller with each generation, embedding them into real-world objects (e.g. food packaging, clothes[2], books, posters, doors or any other tangibles[3]) allows those real-world objects to take part in interactions within the SmartInteraction project, creating a digital representation of the real-world object. This digital representation can accomplish any appropriate task to support the real-world interaction, e.g. offering detailed information on food nutrition or allowing to place reviews on a book borrowed from a public library. There are two possibilities for integrating objects into a P2P interaction environment that allow peers to interact with objects:

- Local proxies: One possibility is to keep the actual data on the objects themselves (e.g. on a RFID tag's custom data storage area) and process it on the peers that wish to interact with the objects. On the peers, a wrapper acts as local proxy for the object, performing all computations and possibly also modifying the data on the object.
  The obvious disadvantage is that a wrapper for each type of application and type of object must be installed on each peer that wishes to interact with those objects. Furthermore, Securing objects is virtually impossible when peers are allowed to modify the object's data (e.g. posting reviews on a book), because the objects themselves, having no processing capabilities, are unable to control access to that data.
- Remote proxies: The other possibility is to only keep a unique ID on the object itself and set up remote *proxy peers* that act on behalf of the objects. When detecting an object, an *ordinary peer* will store its ID in a cache and try to find a peer which is responsible for this object, i.e. which is a remote

[2] Benetton recently adopted Philips RFID technology for 'smart' labels
[3] Tangible interfaces try to give physical form to digital information[8].

proxy peer for this ID (*synchronous proxy interaction*). When no reachable peer claims to be responsible for the object ID, it will start the interaction as soon as one becomes available (*asynchronous proxy interaction*). This allows very flexible interaction patterns between peers and objects (represented by proxy peers) because the proxy can be arbitrarily complex. More importantly, we are able to guarantee object privacy and security with this scheme, which is our main contribution in this paper. We would like to point out that the distinction between an ordinary peer and a proxy peer only stems from the applications running on them; the underlying framework is equivalent on both, the proxy peer just declares to be responsible for certain object IDs. The different terms are only used to distinguish both sides of a P2P communication in protocol explanations.
However, the disadvantage is that an additional peer is needed for the interaction. For some applications, a hybrid scheme may be appropriate: read-only data could be stored directly on the object to communicate first information and a remote proxy could be available for further, more flexible interaction.

For the SmartInteraction project, we decided that the flexibility and security of the remote proxy approach outweighs the disadvantage. Therefore, in the remainder of the paper we will only talk about this method.

3 Related Work

As ad-hoc, P2P interaction between mobile, heterogeneous devices is a young research topic, there are currently not many publications about adding privacy and security measures to this constellation. To the best of our knowledge, this is the first publication to bring up the topic of securely integrating passive objects in an ad-hoc, P2P environment.

Like Frank Stajano pointed out, it is not possible to provide a certificate authority (an online server for all peers) for authentication in a highly dynamical, ad-hoc P2P environment[16, pages 85ff]. Additionally, he suggests to exchange all information which is needed for security measures (certificates, keys) during the bootstrap phase like our system does[16, pages 91ff].

The Freenet project[2] was probably one of the first projects to integrate high-standard privacy and security in a completely distributed P2P architecture.

Marc Langheinrich described a privacy awareness system integrated into an ubiquitous computing environment[11], building on the P3P standard by the W3C (World Wide Web Consortium). The difference to our system is that it depends on infrastructure components whereas our approach is completely based on the P2P paradigm.

The W3C published another recommendation for implementing XML signatures[4] that provide integrity, message authentication and signer authenti-

cation for data of any type.

Additionally, there are numerous papers about security measures for RFID technology, which focus on the physical layers[14] (e.g. preventing denial-of-service attacks). Security in sensor networks is currently also a very active research topic (e.g. [13]).

4 Privacy and Security in mobile ad-hoc networks

4.1 Motivation

For pervasive computing environments in general, security is an important issue because the possibilities for attacks are enormous. Mobile devices like notebooks, PDAs or mobile phones usually carry important data like the user's phone numbers, calendar, notes and other private data (cf. [16]). Following the Code of Fair Information Practices (*FIPs*)[6], any information processing system (mobile ad-hoc P2P systems are in essence only information processing systems) must assure the reliability of data and prevent misuse (principle 5: security). In addition to ensuring this required data security and reliability, our privacy control addresses principle 1 (openness) and partially principle 3 (secondary usage) with our concept of active privacy. Principles 2 (disclosure) and 4 (correction) require organizational precautions in our environment and are thus not covered by this technical solution.

Our core objective is to provide a high level of privacy and data security to the users of mobile, ad-hoc P2P systems. We are currently not working on secure authentication of users to infrastructure components or on any other application that is not focused on the user's own privacy. Therefore, all decisions and policies concerning privacy and security should be local to the respective peers that participate in some secure environments.

Basically, we can distinguish between two different aspects of privacy from the user's point of view, to be tackled with two different privacy policies:

- "Passive" privacy: The goal is to shield the user from incoming information and only present desired messages. As we are all inundated with information, protection has become a necessity. By means of profile matching (to match interests and preferences), the SmartInteraction Framework already provides good shielding to the user, but it can be enhanced by using authentication information in the shielding process.
- "Active" privacy: The goal is to filter outgoing information and only allow non-private information to leave the peer. One possibility to implement it is to allow a fine-grained definition of "access control" in the local peer's profiles and to use authentication information to determine the level of trust in other peers.

In this paper, by the term "privacy control", we describe the active software component that actually makes the decisions on active and passive privacy - it

determines which messages are allowed to be sent or received. With the term "privacy policy", we describe the set of rules and preferences a user defined for the mobile device - they will be enforced by the privacy control.

4.2 Peer-to-Peer Security

The whole communication between ordinary peers is based on XML-messages. Consequently, it is necessary to secure those. This is accomplished with a hybrid system similar in design to PGP which uses both symmetrical (with session keys) and asymmetrical encryption (with private/public key pairs) and digital signatures for authentication.

Our current architecture is based on the following, featuring a very high privacy and security level while operating in an ad-hoc P2P environment and retaining maximum autonomy of peers:

- Use of hybrid encryption.

  Symmetric encryption: To comply with the current best practices, it is advisable to use Rijndael, the AES winning cipher, as the symmetric cipher: it is secure, well-analyzed and fast (the speed penalty compared to RC6 is tolerable, cf. table A.1). In this document, Rijndael will simply be named AES. However, there are some doubts on the security of Rijndael[3], which are currently only theoretical. AES ciphers have a block size of 128 Bit and possible key lengths of 128, 192 and 256 Bit, but some (including Rijndael) are capable to use keys with a higher length (and can therefore be adapted to a higher security level).

  Asymmetric key management:
  - RSA: RSA has the main advantage that it can be used for creating digital signatures as well as for asymmetric encryption, requiring only one keypair for each role.
  - EC ElGamal/ECDSA: Elliptic curve cryptography (ECC) variant of the ElGamal key exchange algorithm. ECC keys offer the same level of security as other methods with significantly smaller key sizes[12] (e.g. 163-bit ECC in contrast to 1024-bit RSA). However, it seems to be generally slower.

  Digest generation: For computing digests of messages, the standard SHA256 algorithm is used. Digests are generally created over the whole XML message and then signed with the private key associated to the respective role that sent the message.
- Use of the X.509v3 standard for issuing and validating certificates.
  For proving the authenticity of (public) keys and their association to roles, certificates are needed. These certificates are issued by certificate autorities (*CAs*) and bind the public key to the role description, digitally signed by the private key of the CA. For mutually authenticating peers that do not know each other directly, certificates seem the best option. However, since a single, hierarchical PKI poses many security risks[5], we opt to not depend

on one. Instead, we will utilize multiple, independent CAs that are completely autonomous (there will be no hierarchical structure between CAs) as well as webs of trust between users.
X.509v3[7] is a well-established standard for certificate formats and is, among others, used for SSL/TLS and S/MIME. Therefore, it is used far more often than OpenPGP as a mere certification standard (outside the domain of email and Usenet net news). There are various free implementations for handling X.509v3 certificates (e.g. OpenSSL, SUN Java JSSE), including Java libraries. The main advantage of X.509v3 is the possibility to define arbitrary fields in the certificate, which can be used to add meta-data (e.g. adding the department in addition to the company name). Although this is also possible with OpenPGP, it would need to be done application-specific – X.509v3 offers this in its standard form. Additionally, X.509 supports Certificate Revocation Lists
  - Certificates can be issued by multiple CAs and each device can store multiple independent certificates for the different roles of its user.
  - Key pairs, role certificates and CA certificates are transferred to the device in a bootstrap phase.
  - Certificates are exchanged between peers before actual data is transferred. When an ordinary peer wants to use authentication or encryption to communicate with another peer, both have to exchange their profiles. Therefore, certificates are automatically embedded in the first message (cf. Fig. 4, step 5).
- All verification, validation and authentication decisions are made locally and autonomously by each peer. In a mobile P2P environment we can not rely on an infrastructure with central servers that are constantly available; thus the devices are forced to be completely autonomous. This not only enhances the privacy of users by keeping important decisions local, but also allows to produce a detailed log of which personal data was sent to whom.

Although our current work concentrates mostly on mutual authentication of peers via certificates, signed and encrypted messages, the privacy control should be able to intervene with all parts of the framework. One example would be to completely turn off the radios of all wireless communication channels on the hardware layer, becoming fully invisible to other peers.

5 Integration of objects

5.1 Problem description

When trying to integrate passive objects (according to our definition in section 2.2) into a security infrastructure, a number of problems arise. The main cause is that an object is, due to not having any processing power, unable to perform any authentication – neither authenticating itself nor verifying the authenticity of other peers. In Fig. 2, the interaction with an object is de-

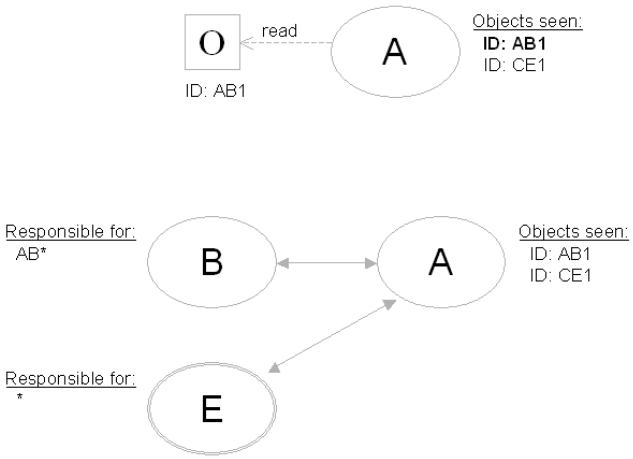

Figure 2. Communication between an ordinary peer and an object via a remote proxy

picted. After detecting an object O in range, the ordinary peer A will search for a proxy peer B which claims to be responsible for O. Because no direct interaction between O and A is possible (A can only detect O and read it's ID and possibly some custom data), all authentication will need to be performed between A and B. As already mentioned in the architecture description, we can not depend on a single trusted third party like a CA to certify A and B and therefore the validity of B's responsibility for O. In real-world scenarios it will be virtually impossible to pre-authenticate A and B (without a single trusted third party) via a web of trust. Thus, it would be possible for an attacker E to claim responsibility for objects (by setting up a remote proxy for the respective IDs) she/he doesn't own, opening up a number of security threats such as:

- Interception of data: When A tries to send private data O, E could easily intercept this data by pretending to be a valid proxy for the respective object.
- Forgery of data: E could send forged data to A, pretending to be legitimately representing O and thus exploiting A's possible trust in O.
- Tracking of users: E could construct a proxy peer claiming to be responsible for all objects in some geographical area and capturing and logging all communication requests. Since ordinary peers will try to contact proxies when interacting with objects, they will also contact E as pretended proxy. E can then track the movement of peers in the geographical area, which can be seen as a severe threat to the privacy of ordinary peers (cf. [14, section 4.1]).

A has, in this situation, no possibility to distinguish between the claims of B and E as depicted in Fig. 2.

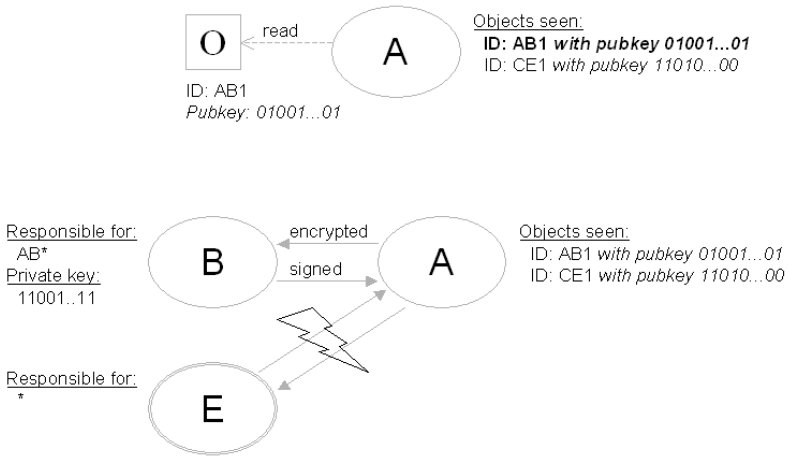

Figure 3. Secured communication between an ordinary peer and an object via a remote proxy

In the following, we present a solution to these privacy threats. However, due to the nature of our project environment, it is inherently impossible (on this level) to prevent against one additional threat, which we also want to describe: An attacker could place an object P (or multiple objects) and a correctly associated proxy peer E in spatial proximity to the real object, gaining a reasonable chance that ordinary peers will find P instead of O and thus contact E instead of B. This "physical attack" can not be overcome with software tools (cf. [16]). However, this sort of attack can successfully be prevented by using certificates to authenticate B, as described in section 4.2. A can then validate the authenticity of B and thus refuse to connect to E when it does not provide a valid, accepted certificate.

Finally, we want to note that it is not necessary to protect against malicious ordinary peers on this level. When the proxy peer is trusted, attacks by ordinary peers can also be inhibited using the techniques described in section 4.2.

5.2 Solution

Our proposed solution is to store a public key on the object itself, and the associated private key on the proxy peer that is responsible for interacting on behalf of the object. As depicted in Fig. 3, O stores a public key in addition to its ID. This key is either an EC (elliptic curve) or RSA key, depending on the available storage area for custom data. The reason for choosing EC ElGamal/ECDSA[10,1] as asymmetric algorithms is that keys with a significantly smaller size offer the same level of security, compared with RSA. On the Identec i-D2 active RFID tags, there are only 64 Bytes available for storing the public key. Table 1 lists the sizes (in Bytes) of public keys in (binary)

Algorithm	Parameters	Key size
RSA	1024 Bit modulus	162
DSA	1024 Bit prime	442
EC	secp160r2 curve	64

Table 1
Public key sizes

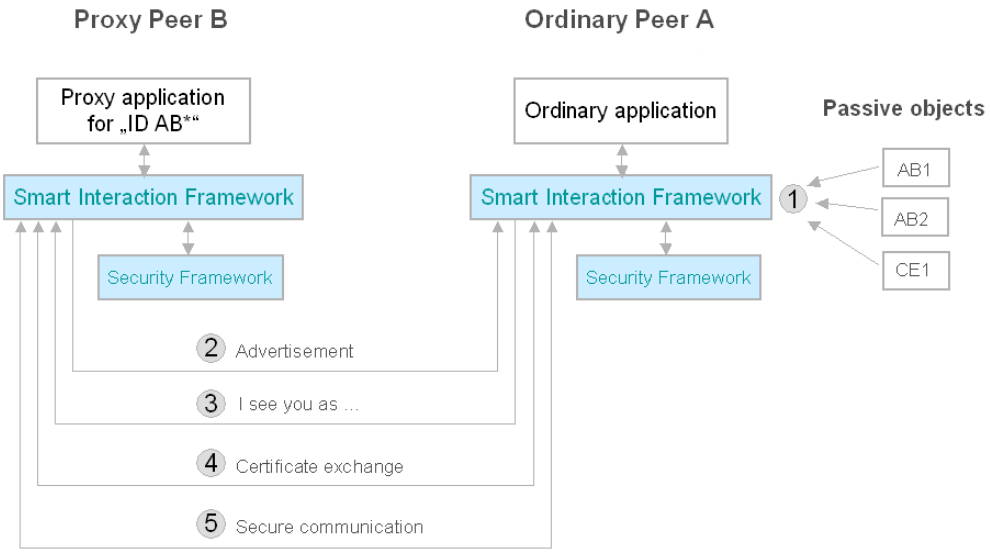

Figure 4. Initiating an interaction between a peer and an object via a remote proxy

DER[9] encoding, generated with the openssl library [4]. When using 160 Bit prime fields for EC, which is considered to be comparably secure as using a 1024 Bit long modulus for RSA (e.g. [12]), the public key will fit perfectly into the objects's custom storage area, even in a standard encoding format. If more storage is available on the used object technology, standard RSA public keys can be used for better run-time performance.

The respective private key (regarding the public key stored on O) belongs to B and is kept there. When reading objects in range, A stores all public keys in an internal table, associating them with the object IDs from which the public keys were read. This table then allows A to decide locally and autonomously if a proxy peer that claims to be responsible for O is valid.

Protocol description

- All messages from an ordinary peer to a proxy peer (from A to B) are AES-encrypted with a temporary session key. Using the public key from the internal table (the public key stored on O), A can send the session key to B utilizing either EC ElGamal or RSA.
- All messages from a proxy peer to an ordinary peer (from B to A) are signed, either with ECDSA or RSA using the private key stored on B. A can then verify all messages received from B for integrity and validity using the public key from its internal table. This includes the announcement messages (cf. Fig. 4, step 2) sent by B. Thus, E can not send messages that A considers as valid, not even the announcement message where E claims to be responsible for O.
- Further security measures to obviate other attacks can be applied one protocol level higher. On this level, the communication is transparent to both involved peers and can thus be handled as it would be between two ordinary peers, including the use of certificates for authentication. Our protocol on this level is shown in Fig. 4:

Step 1 An ordinary peer A finds passive objects (AB1, AB2 and CE1) and stores their IDs and public keys (EC ElGamal/ECDSA or RSA) in a local table.

Step 2 Proxy peer B sends a signed advertisement to A where it claims responsibility for a set of passive objects (AB*).

Step 3 A compares the advertisement with the entries in the local table and reports successful matches (AB1 and AB2) to B, thus notifying the proxy application of the actual object IDs it should act for.

Step 4 Exchange of all needed certificates between A and B (with RSA keys).

Step 5 Finally, secure communication between A and B on behalf of AB1 and AB2 is possible. Messages are fully signed and encrypted in both directions, using the RSA keys from the exchanged certificates (equivalent to normal interaction between ordinary peers).

It is important to note that B does never send a cryptographic key (neither a temporary, symmetric session key nor an asymmetric public key) to A until the certificate exchange; A either generates the key (the session key for AES encryption) or uses the public key that was read from the object, fulfilling our requirement of autonomy. Thus, it is impossible for E to spoof messages from O and to read messages destined for O. However, this holds true only if two assumptions are fulfilled:

- The private key belonging to the public key stored on O is only stored on B and kept safe. As this is a standard requirement for the usage of private keys, it is a matter of physical security of B. In a typical scenario, the remote proxy peers will either run on trusted embedded devices or on

[4] The openssl cryptographic library offers a wide range of symmetrical and asymmetrical algorithms, including certificate authority functionality, and a command line interface for accessing them. It is freely available at http://www.openssl.org/.

tightly controlled servers, allowing to secure the key.

- The public key stored on O can not be changed by E. This is usually guaranteed with read-only objects that can be written only once (e.g. fuses in RFID tags) or by password-protected write access to the objects. Practically, this places no restriction on the possible scenarios because only the *physical* object needs to be read-only, not its virtual counterpart (represented by the remote proxy). With the virtual representation in the P2P environment, any interaction is possible, including operations that modify data on the remote proxy, associated with the physical object.

We want to point out that this solution will not be able to provide complete privacy and security on its own, it has to be seen as an addition to the standard methods described in section 4.2. However, without an addition, a secure integration of objects as defined in section 2.2 does not seem to be possible. When both layers (as laid out in this paper) are used, we are now able to provide a high level of privacy and security, even in this difficult environment.

6 Conclusions and Future Work

In the present paper, we have introduced a method to securely integrate passive objects into an ad-hoc P2P environment. After shortly introducing the SmartInteraction project and defining the term object in this context, a method for transparently interacting with objects which do not have own processing power has been given. The problem with this style of interaction via a remote proxy peer (responsible for a list of objects identified by their ID) is that three parties are involved in the authentication process: the object, the ordinary peer and the proxy peer. Because the object does not have the ability to actively participate in the authentication process, the ordinary peer must use locally available information to verify the authenticity of all proxy peers that claim to be responsible for the object. Our solution in the SmartInteraction project is to put public keys directly on the objects, which must be read-only to ordinary peers. With this technique, an attacker can no longer spoof responsibility for an object.

We have also shown by empirical performance evaluation that strong encryption is possible even on PDAs, giving a reason for our usage of standard cryptographic algorithms in the SmartInteraction project instead of special algorithms optimized for embedded devices.

Currently, we have proof-of-concept implementations of our methods and protocols, which we will integrate into our component-based framework in the next months. Formal reasoning on the protocol still has to be done. Furthermore, we want to work on another important aspect of privacy and security on mobile devices: configuration by end-users. Because the privacy and security component is very flexible, there are also many aspects that can be configured (e.g. details on certificate validity checks). End-users will

generally be unaware of those aspects and unable to set them properly, not knowing the consquences of each. Thus, we would like to implement the idea of a simple "slider" to set the desired level of privacy and security with explanations being displayed as the slider is moved. This slider will allow the user to define an appropriate compromise between privacy and convenience. However, as the default settings for the different slider positions will need to be defined in advance, we want to perform a field study to measure the actual usage of certain privacy and security features; this will be possible after the privacy and security module has been completely integrated into a new demonstration application, which will be presented in more detail in future work.

The topic of mobile, ad-hoc P2P interaction is still young – thus privacy and security concepts will need to mature and new concepts for other application areas will probably need to be developed.

References

[1] ***Elliptic curve digital signature algorithm (ECDSA)*, American Bankers Association (1999).**

[2] **Clarke, I., T. W. Hong, S. G. Miller, O. Sandberg and B. Wiley, *Protecting free expression online with Freenet*, IEEE Internet Computing 6 (2002), pp. 40–49.**

[3] **Courtois, N. and J. Pieprzyk, *Cryptoanalysis of block ciphers with overdefined systems of equations.***
URL `http://eprint.iacr.org/2002/044/`

[4] **Eastlake, D. E., J. M. Reagle and D. Solo, *XML-signature syntax and processing* (2002), W3C Recommendation, HTML version at http://www.w3.org/TR/xmldsig-core/.**

[5] **Ellison, C. and B. Schneier, *Ten risks of PKI: What you're not being told about public key infrastructure*, Computer Security Journal 6 (2000), pp. 1–7.**

[6] ***Records, computers, and the rights of citizens viii*, U.S. Dep't. of Health, Education and Welfare, Secretary's Advisory Committee on Automated Personal Data Systems (1973).**

[7] **Housley, R., W. Polk, W. Ford and D. Solo, *RFC3280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile* (2002).**
URL `http://RFC.net/rfc3280.html`

[8] **Ishii, H. and B. Ullmer, *Tangible bits: Towards seamless interfaces between people, bits and atoms*, in: *Proceedings of Conference on Human Factors in Computing Systems (CHI '97)* (1997), pp. 234–241.**

[9] ***ITU-T recommendation X.690: Information technology - ASN.1 encoding rules: Specification of basic encoding rules (BER), canonical encoding rules (CER) and distinguished encoding rules (DER)* (1997), also ISO/IEC 8825-1:1998.**

[10] **Koblitz, N., A. Menezes and S. Vanstone, *The state of elliptic curve cryptography*, Designs, Codes and Cryptography 9 (1994), pp. 173–193.**

[11] **Langheinrich, M., *A privacy awareness system for ubiquitous computing environments*, in: *UbiComp 2002: Ubiquitous Computing*, Lecture Notes in Computer Science 2498, 2002, p. 237ff.**

[12] **Lenstra, A. K. and E. R. Verheul, *Selecting cryptographic key sizes*, Journal of Cryptology: the journal of the International Association for Cryptologic Research 14 (2001), pp. 255–293.**

[13] **Perrig, A., R. Szewczyk, V. Wen, D. E. Culler and J. D. Tygar, *SPINS: security protocols for sensor netowrks*, in: *Mobile Computing and Networking*, 2001, pp. 189–199.**

[14] **Sarma, S. E., S. A. Weis and D. W. Engels, *Radio frequency identification: Risks and challenges*, CryptoBytes (RSA Laboratories) 6 (2003).**

[15] **Satyanarayanan, M., *Privacy: The achilles heel of pervasive computing?*, IEEE Pervasive Computing 2 (2003), pp. 4–5.**

[16] **Stajano, F., "Security for Ubiquitous Computing," Wiley Series in Communications Networking & Distributed Systems, John Wiley & Sons, 2002.**

[17] **Want, R. and A. Hopper, *Active badges and personal interactive computing objects*, IEEE Transactions on Consumer Electronics 38 (1992), pp. 10–20, first published as ORL technical report 92.1: "The Active Badge Location System".**

A Performance evaluation

Before defining an architecture that fulfills our requirements, we have to study which techniques are feasible on the described devices. The following performance data was obtained on an Athlon 1,8 GHz PC, on a Fujitsu-Siemens Pocket Loox (with a 400 MHz XScale ARM processor and 64 MB RAM) and on a Compaq Ipaq 3870 (with a 206 MHz StrongARM processor and 64 MB RAM) with a small test program utilizing the freely available BouncyCastle

Java cryptography library [5] . These values should only give an overview as the current implementation is not optimized for performance. Additionally, these values also include some processor time for console output, which is slow on PocketPCs (without performance output, the test program should be faster). On the PC, 5 test runs were done to obtain the average values while on the PocketPCs 10 test runs were performed. All values are in milliseconds (ms).

Symmetric encryption and decryption as well as digest generation were performed on a typical XML message, which had a size of 1768 Byte. Asymmetric signatures and RSA encryption were performed on the 256 Bit digest generated from that message. Key lengths were 128 Bit for symmetric and 1024 Bit respectively 163 Bit for asymmetric encryption. These tests should reflect the typical operations. For EC, we could only test the ECDSA (signature generation and verification) part, because there seems to be no implementation of EC ElGamal available at the moment. We will implement EC ElGamal with BouncyCastly and make it publically available as the SmartInteraction project progresses. The higher variance in the PC test runs can be explained by concurrently running programs.

As can be seen from table A.1, high security encryption and signatures are possible on current PDAs – a typical message can be encrypted with AES/RSA and signed in less than 1300 ms and decrypted and verified in less than 1200 ms. The high values for generation of RSA key pairs do not influence the intended security architecture because keys would be generated on more powerful external systems and transferred to the mobile devices in a bootstrap phase. However, on the PocketPC platform, EC operations (signature generation and verification) take significantly longer than their RSA counterparts, most probably because of a weak floating point unit; on the PC, EC signature generation and verification have a run-time comparable to RSA. Currently, we will use RSA as asymmetric algorithm whenever possible and only employ EC when key size is the limiting factor.

[5] The library, including API documentation, can be downloaded freely from http://www.bouncycastle.org/

	PC mean	PC std. dev.	Loox mean	Loox std. dev.	Ipaq mean	Ipaq std. dev.
AES: init. for encr.	3	0,00	172	60,48	200,5	70,73
AES: encryption	20	15,53	402,7	115,21	479,6	86,50
AES: init. for decr.	2,2	0,98	72,3	39,03	167,9	97,12
AES: decryption	24,2	16,17	493,3	151,47	373,9	116,43
RC6: init. for encr.	3,8	3,60	90	36,52	218,4	146,83
RC6: encryption	6,6	0,80	215,8	87,14	447,5	163,96
RC6: init. for decr.	2,4	0,80	48,2	8,85	161,4	48,42
RC6: decryption	13,8	16,12	258,8	68,48	377,1	62,92
SHA256: digest	16,4	4,84	306,9	101,76	770,3	507,15
RSA: param. gen.	2,4	0,49	85,3	123,31	45,5	49,47
RSA: keypair gen.	3057,8	2375,87	10328,5	5043,77	15886,8	7164,11
RSA: signature gen.	73,8	16,23	360,2	19,84	421,6	115,54
RSA: signature verify	2,2	0,40	142,5	27,49	187,8	71,58
RSA: init. for encr.	4,4	4,32	4,8	1,17	3,5	0,50
RSA: encryption	6	8,00	34,6	4,80	151,8	39,21
RSA: init. for decr.	2,6	0,80	4,6	0,49	392,6	85,01
RSA: decryption	48	16,98	123,5	2,46	254,3	30,51
EC param. gen.	9,4	4,59	748,6	154,57	514	60,37
EC keypair gen.	61,2	22,48	9194,2	453,19	5998,2	492,87
EC signature gen.	35,8	8,61	8959,3	444,01	6622	378,30
EC signature verify	51,6	4,22	17138	769,20	11389,6	667,37

Table A.1
Performance measurements