Security for Context-Aware ad-hoc Networking Applications

Yeda Venturini, Vlad Coroama, Tereza C.M.B. Carvalho, Mats Naslund, and Makan Pourzandi

Abstract With the rapid spreading of ubiquitous computing applications, the importance of security concepts coping with their needs is also growing. While the possible application areas are so vast that one all-purpose security middleware fitting all the different needs seems impossible to realize, it is undoubtedly meaningful to have security frameworks covering the needs of as many applications as possible. In this paper, we thus discuss a security middleware for context-aware ad-hoc networking applications in home and work environments. The article focuses on two novel issues: it shows that the solution is particularly well-suited for context-aware applications, an often-encountered type of applications within home and work environments; and it discusses the encountered, non-trivial trade-offs between ad-hoc networking, context-awareness, and strong security.

1 Introduction

More than 15 years ago, in his seminal article "The Computer for the 21st Century" [13], Mark Weiser has been the first to foresee the spreading of sensing, computing, and communication technologies into everyday things, an evolution which he called "ubiquitous computing." The numerous technological progresses and societal changes that happened ever since, make his vision nowadays seem more realistic then ever before. On technological front, as Mattern [8] argues, we have witnessed the ongoing miniaturization of electronics; rapid progresses in wireless communication technologies such as WiFi, Bluetooth, or Near Field Communication (NFC);

Yeda Venturini · Vlad Coroama · Tereza C.M.B. Carvalho
USP – University of São Paulo, Brazil, e-mail: {yeda,vcoroama,carvalho}@larc.usp.br

Mats Naslund · Makan Pourzandi
Ericsson Research, e-mail: {mats.naslund,makan.pourzandi}@ericsson.com

the diversification and miniaturization of sensors; the emergence of identification technologies such as RFID tags and of indoor and outdoor positioning technologies.

Together with these technological advances, a large amount of applications for ubiquitous computing technologies has emerged inside academia and industry. Albeit not all, many of the envisioned mobile and ubiquitous computing applications involve the ad-hoc networking of devices that have been enhanced with computing and communication technologies in a similar manner to Weiser's vision. They encompass many fields, such as healthcare, commercial, military, home and work environments, and – for now – usually only exist as prototypes or as scenarios for the near future. Nevertheless, by looking at the existing scenarios and prototypes, it quickly becomes clear that virtually all of them will need to provide security and privacy mechanisms. However, the existing prototypes typically either do not provide any security mechanisms at all, or they do so as a proprietary, one-time solution. We have thus proposed such a generic and flexible security middleware. In the beginning, our project has been limited to the networking of devices belonging to one owner only [12]. Recognizing the added value obtained by networking devices of different owners, we have more recently extended the project to allow for the networking of such devices as well [9].

After presenting our middleware as a solution for ad-hoc networking applications in home and work environments, the present article addresses two novel issues: Firstly, it shows that our solution is particularly well-suited for context-aware applications. Secondly, since not all ad-hoc networking applications are context-aware (and, in fact, most are not), it discusses the trade-offs that exist for a security solution aiming at both ad-hoc networking and context-awareness. The remainder of the paper is thus organized as follows: Section 2 presents the application scenarios targeted by our work. Out of the various security requirements in ad-hoc networking, section 3 shows which security services are relevant for these applications. Section 4 introduces the concept of context-awareness, and shows how it has to be supported by the proposed solution. Finally, section 5 summarizes the architecture of our solution, highlighting the trade-offs between ad-hoc networking, context-awareness, and strong security.

2 Sharing of Services inside Familiar and Work Environments

If Weiser's vision is to become true – and, as argued, many technological trends point into this direction – more and more "smart" personal devices will belong to one person, silently bringing services to their owner. Since tiny, wirelessly interconnected computers can be embedded into and enhance almost any everyday object [8], the range of possible applications and services brought by such devices is virtually endless. The possible scenarios reach from the military warfare – where myriads of tiny, dust-sized particles that can sense movements would be spread in-

side the enemy territory[1] – to such mundane everyday household devices like coffee cups [4] or toothbrushes [6]. It is clear then that it would be a futile undertaking to try designing a security middleware that would be ideally suited for all possible kinds of applications. As Hubaux et. al put it, "clearly, security requirements depend very much on the kind of mission for which the mobile ad-hoc network has been conceived, and the environment in which it has to operate" [5]. We thus present in this section the application scenarios that our middleware has been built for.

Within the ubiquitous computing research community, there has been a long tradition of scenarios and prototypes involving several devices belonging to the same person, which cooperate to bring services to their owner. Such services can reach from the simple synchronization of documents between the several information appliances of the user, to more ambitious visions and prototypes, closer to the vision of ubiquitous computing technology embedded into all sorts of everyday objects. In a project from the Japanese Waseda university [6], for example, the smart user's toothbrush (equipped with accelerometers and an RFID tag for identification) communicates with the likewise smart bathroom mirror, which then displays information such as weather forecast for the adults brushing teeth in the morning, or plays interactive games with the children. Going further, several projects have argued for the augmentation of entire houses with sensing, computing, and wireless communication facilities, thus realizing so-called "smart homes." Some of the best-known examples are the Philips HomeLab[2] and GeorgiaTech's "Aware Home" [7]. The prototypically realized services of smart homes include: a service for finding "Frequently Lost Objects" (FLOs), such as keys, wallets, or remote controls through the use of an indoor positioning system and the tagging of FLOs; or a service that automatically starts the air conditioned or the house heating when one of the residents approaches home (by using the vehicle's positioning system which communicates with the house via the driver's mobile phone).

Most of the above-mentioned applications need an underlying security concept, providing various security services, such as authentication and confidentiality. The wireless synchronization of documents between several devices of one person, for example, needs both. Likewise, it would be a bad idea for a thief outside the home to be able to use the service for finding lost objects to locate all the wallets inside the house, or to open the garage's door upon approaching the home. Nevertheless, most of the projects presented are prototypes, and typically either don't deploy any security at all, or use an ad-hoc proprietary security solution, which can have no impact on further research. Recognizing the need for a general-purpose, flexible security layer for such applications, we have proposed and implemented the concept of "Personal Security Domains" (PSDs) [12]. PSDs are a security middleware that can be used by such applications of cooperating smart home artefacts. Given their high abundance, we have focused the security middleware for ubiquitous computing applications inside the home environment and have from the beginning excluded military (and, to some extent, commercial) as target applications of the PSDs.

---

[1] See robotics.eecs.berkeley.edu/∼pister/SmartDust/.

[2] See www.research.philips.com/technologies/misc/homelab/

Nonetheless, devices belonging to one person and bringing services only to their owner, do not exploit the full potential that ubiquitous computing has to offer in home and familiar environments. As a recent study [2] shows, several of the technological devices existing nowadays in homes are shared among family members rather than being exclusively used – the authors talk about "shared ownership" versus "individual ownership." The survey refers to the usage patterns of different technologies inside households. While some of the technologies were individually owned (i.e., mobile phones and music players), others were shared among the residents. Computers and TV sets, for example, were usually used by more persons, although often their number exceeded the number of home residents. Furthermore, computers were not only shared as physical devices. The profiling allowed by the operating system to logically separate the different users was usually not used in this home setting. This leads to the question whether the sharing of future ubiquitous computing devices would not also induce a new quality to the services brought to a group of people – in family or friends settings, but also in work environments. And indeed – numerous ubiquitous computing projects emphasize such sharing and common use of devices and services provided by these. A smart room, for example, could autonomously derive a meeting going on (by noticing a gathering of smart coffee cups [4]). It could then modify its own behavior, or that of other entities (for example, turning the mobile phones of participants to silent mode [4]). A business traveler entering the office building of the company she is visiting, can be provided temporary access rights to parts of its computing infrastructure, such as the right to use projectors for slides, or printers for handouts. The sharing of documents or contacts among the devices of work colleagues can be as meaningful as sharing them with friends. However, a fine-granular content management system is obviously needed for such an application – the owner must be able to decide which data to share with any of these groups. Being able to grant access to a smart home for friends also seems a meaningful feature; as well as letting them operate the aircon. Similarly, one could grant temporary access to cleaning stuff or repairmen, without the need of meeting them or handing them an (easy to duplicate) physical key.

Obviously, all of these scenarios also need security concepts in order to control the access to data and services, and/or to ensure the privacy of communications. We have thus broaden our concept of PSDs to MPSDs ("Multiple Personal Security Domains"), as presented in [9] and summarized in section 5. An MPSD is realized by the (temporary) joining together of two or more PSDs (belonging to as many owners) with the aim of sharing some of the services offered by these.

3 Security for Shared Family and Work ad-hoc Networks

From the early stages of the project, the question of which security services are relevant inside shared family and work ad-hoc networks of personal devices – and should thus be offered by our middleware to its client applications – has thus been raised. In their seminal paper "Securing Ad Hoc Networks" [14], Zhou at al. define

following security requirements for ad-hoc networks: *availability*, *confidentiality*, *integrity*, *authentication*, *non-repudiation*, and *authorization*. While these requirements are general to any type of network, the authors identify three challenges that make them harder to achieve in an ad-hoc wireless network: *the use of wireless links*, the non-negligible *probability that nodes will be compromised* due their typically poor physical protection, and the *high network dynamics* with nodes frequently leaving and joining the network [14]. This facilitates, among others, following types of attacks: jamming attacks on the physical layer, disrupting the network protocol on network layer, and eavesdropping. Hubaux at al. [5], taking a slightly different approach, first differentiate between vulnerabilities of the basic and of the security mechanisms. Vulnerabilities of the basic mechanisms are similar to Zhou's approach, including eavesdropping, active interference and routing protocol attacks. As for the security mechanism, Hubaux considers the establishment of keys as the most critical and complex issue. The initialization phase, and the level of trust between the entities involved in it, are examples of issues that need to be answered for a good solution. Since Zhou at al. [14] also focus their proposed solution on how to secure routing and how to establish a secure key management service in an ad-hoc networking environment, both articles are concerned with key management as the main challenge in an ad-hoc network. Authentication, confidentiality, and integrity can all be achieved through a secure and robust key management.

As argued above, the security requirements depend on the network purpose and the environment in which it has to operate. Looking at the security requirements defined by Zhou [14] and Hubaux [5], it is easily noted that not all have the same importance within our target home and work environment scenarios. Denial-of-service attacks, for example, are more or less likely, depending on the layer the attack would occur on. Due to the typical small spread of the network, as well as the heterogenous physical transport protocols that will be used by most applications (e.g., WiFi, Bluetooth, GSM, etc.), we believe an attack on the physical layer to be unlikely, and will not pursue it further. A denial-of-service attack on the network layer would consist of the disruption of the routing protocol. This is a more likely attack in the target scenarios, and – as section 5 will show – one that is being considered by our solution which eliminates routing through the use of service discovery and then direct peer-to-peer communication. Finally, as Zhou et. al put it: "the adversary could bring down higher-level services like the key management layer" [14]. This is a more likely threat, that our solution has to account for.

Confidentiality and *authentication* are two security requirements at the very core of the target scenarios. In virtually all scenarios, the user's device needs to be sure of the authenticity of the other devices it is exchanging information with. While confidentiality is not as general a requirement as authentication, it is highly relevant to some of the scenarios, such as the document synchronization of possibly sensitive information. These two security requirements can be accomplished – as the countering of denial-of-service attacks on high layers – with a strong key management mechanism. *Non-repudiation*, on the other hand, is not of importance to our scenarios. Since they do not involve commercial applications (in which the sending or receiving of messages possibly needs to be proven in court), our middleware can

ignore non-repudiation mechanisms. Likewise, a lack of message *integrity* due to benign failures like bad connectivity, does not constitute a large problem in any of the scenarios. Even more, it is a problem that usually gets taken care of (if possible) on the lower layers.

Summarizing, due to the nature of the scenarios and from the considerations above, our middleware has to focus on providing the client applications with strong authentication and confidentiality mechanisms. It also needs to counter denial-of-service attacks on the routing mechanisms and on the services on higher layers.

4 Context-Aware ad-hoc Networking Applications

More often than not, ad-hoc networking applications are one-purpose only, and consist of a large amount of small nodes working together towards that task. Especially sensor networks applications fall under this category, such as the already mentioned military battlefield monitoring through "smart dust" particles, or the detection of oil spills or forest fires through a network of wireless sensors [1]. Such applications are characterized by a large amount of relatively small nodes, each with a limited amount of computing power, usually a limited amount of energy (since they are typically battery-operated), that are distributed into an often hostile environment (e.g., the enemy territory, or a forest with all the problems caused to the nodes by weather and wildlife). Three more factors make the algorithms for this kind of ad-hoc networks additionally challenging: the wireless communication ranges of the nodes are typically quite short, the network topology is a-priori unknown (since the nodes get typically distributed into the environment with a random pattern, e.g., by being thrown from an airplane), and the probability of any individual node to fail is relatively high due to the rough environment they operate in. For all these reasons, in such applications there's a large emphasis on the routing protocols, and on the security mechanisms, which have to function despite the failure of single nodes.

By contrast, the applications our security middleware is being built for, operate under much friendlier constraints. The nodes envisioned are either electronic devices (e.g., PDAs, smart phones), or large and typically immobile devices, such as garage doors, air conditioned devices, or vehicles. They thus either inherently possess strong computing capabilities, or plenty of space for these to be built in. Energy supply is not an issue either – such devices are either already connected to the power grid (aircon, garage door), or can generate a virtually unlimited amount of energy by burning fuel (vehicles), or have strong rechargeable batteries, that users are already used to charge on a regular basis (PDAs, phones). The supplemental power needed for their "smartness" and for the wireless communication thus represents no obstacle. Furthermore, the devices in our target scenarios are collocated in close physical proximity. The information does not have to be routed over large distances.

Obviously, these constraints don't pose for a security middleware the kind of challenges that arise from application domains such as those described above. However, there is an entirely different range of characteristics that makes other aspects of

the design rather challenging. Firstly, as argued in the last section, the applications often need stronger authentication and confidentiality services as compared to such ad-hoc networking applications as presented above. Secondly, not only must the middleware allow for all sorts of different applications to be built on top of it; most envisioned devices will typically run an entire collection of heterogenous services in parallel, which all have to be supported by the framework. Thirdly, these services must be able to exhibit a refined and fine-granular model of rights and authorizations, as the examples of the repairman being granted temporary access to a house, or of the businesswoman being granted the rights to use part of the computing infrastructure of a different company show. This sort of granularity is never encountered in the typical sensor networks applications. Fourthly, and arguably most challenging, many of the applications using our middleware will be context-aware, a central concept to our solution, and thus presented in the current section.

Being able to sense their surroundings and often being mobile, an outstanding attribute of ubiquitous computing applications is their capability to adapt to changes in the environment. This feature is called *context-awareness*. The term has first been defined by Shilit as "software that adapts itself to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time" [11]. The definition has later shifted towards a more user-centric view, and has also become more general. The definition widely accepted nowadays has been provided by Dey and Abowd: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves" [3]. Several other contributions, such as Schilit's [11], have listed the most important aspects of context: *user environment* (location, collection of people nearby, social situation), *physical environment* (lightning conditions or acceleration of a vehicle), *computing environment* (available processing power, communication protocols). Such definitions, although not as general and thus sometimes too constraining, help to identify the most common types of context. Dey and Abowd further differentiate between *primary* and *secondary* types of context: "Location, identity, time, and activity are the primary context types for characterizing the situation of a particular entity" [3]. All secondary types of context can be derived from these fundamental ones: the email address, for example, from the identity of an entity, or the people nearby from its location.

For the scenarios presented in section 2, following primary types of context are relevant: identity, location, time, and computing resources. The user's *identity* is relevant to virtually all envisioned applications. From it, the key security issue of *access rights* (both physical and logical access) is derived. From the user's *location*, several types of secondary context can be derived. Access rights can depend on the user's location as well, not only on her identity. In our example of the businesswoman visiting for a few days a foreign company, she is granted access to parts of the building's computing infrastructure. However, these rights could be restricted to only the times she resides inside the building (and some social control over the way she uses the infrastructure exists), and not when she is in the hotel, for example. Furthermore, location also implies *proximity*: which other personal devices are

in the vicinity, and which devices belonging to others. Likewise, the *time* also determines access rights. A repairman could be granted a two-hour timeslot to enter the home without the residents being present, for example, or a cleaning aid could be granted access every Tuesday. The available *computing and communication resources* can be highly relevant for security-sensitive applications (and the middleware supporting them), especially for remotely-executed services. Depending on the kind of communication protocols available, such services could be executed or not.

5 Multiple Personal Security Domains and Context-Awareness

According to ISO/IEC 10181-1, a security domain is "a set of elements, a security policy, a security authority and a set of security relevant activities in which the set of elements are subject to the security policy, administered by the security authority, for the specified activities." This concept is applied in some commercial products like the Java Micro Edition architecture (J2ME), where the elements subject to the security policy are Java applications running on the device's Java Virtual Machine (JVM). Our implementation of the security domain concept is different from J2ME. The elements can be applications hosted or distributed in different mobile devices. The granularity of the security policies is much higher: It is possible to define rules for devices, applications, applications hosted on a specific device, or even for a particular operation of an application. We summarize here concepts and architecture of our middleware, more details can be find in [9] and [12].

In our terminology, a *Personal Security Domain* (PSD) is a group of fixed and/or mobile components, where each component can be authenticated, trusted and securely communicated with through a common security association, subject to a security policy. The PSD must have an owner, called *controller*, who creates the PSD with the components that he owns or is responsible for. PSD components can be geographically distributed. A *PSD partition* is a subdivision of a PSD formed by all PSD components that are at the same location and can thus communicate directly (layer 2 of the OSI model). PSDs can be joined together, forming a *Multiple PSD* (MPSD). The controller role for an MPSD can be either shared among the PSD controllers or one PSD controller alone takes this role. The MPSD controller creates the MPSD with the desired PSDs, while the security policy for PSD components is still defined by the respective PSD controller. While PSDs typically have long lifetimes, MPSDs are temporary security associations with a shorter lifespan.

Figure 1 shows the layer structure of the PSD system, consisting of the Security Enforcement Layer (SEL) and the Security Domain Layer (SDL). While the SDL implements the PSD control and management APIs, the SEL provides secure communication between entities of the (M)PSDs, ensures entities authentication, communication security (confidentiality and integrity), and the applications access control (authorization). It enforces dynamically the security policy for the (M)PSD. It is also responsible for the management of the entities' trust values, which are used in a trust based authorization model.

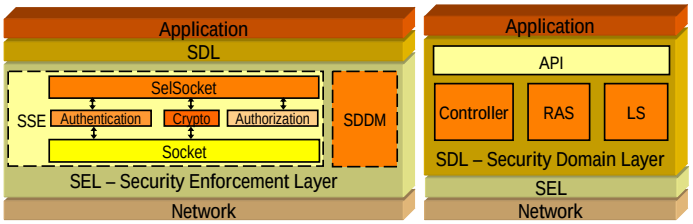

Fig. 1 Layers of the Security Domain Architecture.

The SEL has two basic components: the SelSocket Encapsulation (SSE) and the Security Domain Data Manager (SDDM). The SSE enforces the security based on the security data provided by the SDDM. The SDDM manages security data, provided by the Security Domain Layer (SDL), which will be presented shortly. To make its usage easier and more transparent for application developers, the SelSocket works similar to a standard socket. However, before releasing the connection for an application, the SelSocket authenticates the entities and verifies the authorization for communication. After authorization, the application can send messages that get encrypted by the SelSocket with the session key, which has been previously exchanged during authentication. The SelSocket enforces authentication and authorization based on the security data provided by the SDDM for the requesting application. The SEL has two special features that contribute for security in ad-hoc networks: domain authentication and trust-based authorization. The domain authentication is based on a (renewable) key shared among the entities of a PSD. This authentication is enforced before the mutual entity authentication. It aims at an early identification of the PSD entities in a densely populated medium, as in wireless communication. Additionally, the domain authentication provides the privacy needed by the entity authentication. Besides the fine-granularity for rules, the authorization service relies on a trust concept to face the possibility that some PSD entity has been compromised. Suspicious behavior of entities generates security events, which dynamically reduce the trust of that PSD entity. Authorization rules for applications, services, or devices with critical security requirements can be based on minimal trust, so that any misbehavior of some PSD entity causes its disallowance. These features contribute towards overall availability, through the early disconnection of non-PSD entities or of compromised ones, avoiding hence denial-of-service attacks, as well as eavesdropping attacks during entity authentication.

The Security Domains Layer (SDL), on the other hand, provides an API for PSD management and for the development of applications on top of the MPSD, as well as three basic services: *controller*, *lookup* and *remote access*. The PSD creation, entity joining or leaving, as well as security policy definition, are operations performed by the *controller service*. A PSD possesses a symmetric key and an auto-signed certificate (root certificate), which are known by any PSD entity. The PSD controller is thus the personal Certificate Authority (CA) for the respective PSD, issuing certificates for PSD entities during the joining process and revoking certificates for the

leaving process. The symmetric key is shared among all PSD entities and used for domain authentication. This key is also used to generate a pseudo-random number [12] for multicast messages, so that only entities sharing the same key answer to the originator. For example, home devices will not answer to a location message if the requestor is not a home device. The *remote access service* (RAS) enables access to resources across partitions, which can be independent networks, thus dealing with conflicting addresses in different partitions and enforcing the security between them. The device that hosts the RAS and receives the remote connection requests needs to have a valid IP address. The *lookup service* (LS) allows devices to discover other devices and services available. The LS allows a service-based network formation, in which the user does not need to know in advance the available services. When a service starts, it searches for the LSs in its reach area and announces itself to all of them. For service use, the user searches for available LSs, requests a service, receives the available service list, and selects a service to use. The list of available services has all information required for service connection. We developed a novel service location protocol, with additional functionalities (as compared to SLP or JINI, for example), which are related to the available remote connections and security. Firstly, the LS answers only to entities from the PSDs it trusts. Secondly, it supports customized attributes for entities' registration, which can be primary context information with different interpretation depending of the device or the service type. Finally, it is possible to discover services in a remote partition through the RAS service.

Taken together, all these concepts offer the strong security needed by applications, while allowing at the same time the flexibility needed for their context-awareness. The authorization-based service discovery offered by the LSs (differentiating between services offered within the own PSD and services from affiliated MPSDs), for example, provides fine-granular contextual information about the local computing environment: the locally available own and foreign devices, their services, as well as the communication protocols available for accessing them. Through the transparent communication between RAS and LS, the application can also gain detailed information about physically remote environments and decide upon accessing services there. To test context-awareness inside MPSD, we have implemented a prototypical file and address book sharing application and tested it in a setting with seven MPSDs. The promising results, however, lie outside the scope of this paper.

Some ad-hoc networking projects go even further and allow for very subtle ways of taking into account the physical context surrounding the application. Robinson and Beigl [10], for example, in a solution similar to ours (but for a different class of applications, as will be shown shortly), let devices that are in close physical proximity form a so-called "Trust Context Space." They assume that, as in real-life people sharing the same physical boundaries have a certain level of trust, devices sharing the same sort of common space should also automatically share trust. They implemented a prototypical application where all devices within the same walls, independently, derive a common key from the sounds within the physical space they share.

While the beauty of this solution lies in its absolute lack of any explicit user intervention (such as pairing of devices), for our class of applications such a solution

would not work for two reasons – one of fundamental, the other of pragmatic nature. The fundamental problem is the fine-granularity needed by our applications. Not all people in the same office should share the same secrets (e.g., the visiting businesswoman will only have restricted access), and not all of a device's secrets should be shared with everyone (e.g., the private contacts or calendar items should not be shared with work colleagues). This sort of semantic knowledge cannot be automatically derived by the middleware, but has to be explicitly defined by the owner of the device. The pragmatic problem lies in the imprecision of the sensors detecting environmental attributes such as sound. This limits the length of the key that can be derived, while keeping the error rate low enough for the solution to still be practical.

Due to both reasons, a solution such as Robinson's and Beigl's, while allowing for a very spontaneous ad-hoc networking, with no previous knowledge about other devices or any human intervention, can only be used by applications needing no more than a basic level of security. It actually seems that there is an inherent trade-off between ad-hoc networking and context-awareness, when looking at the security features. If a context-aware application should allow a very spontaneous ad-hoc networking, with no previous device pairing or knowledge about other device's existence, nor any human intervention, the security level that can be guaranteed is relatively low. If, as in our case, the security level must be relatively high, some compromises have to be done in terms of the spontaneity of the networking allowed. This trade-off has governed the design of our middleware.

6 Conclusions

In the present article, we have summed up the experiences from the project "Multiple Personal Security Domains," a security middleware focused on a specific class of ubiquitous and mobile computing applications. After having first argued why we have limited the class of targeted applications to ad-hoc networks of smart devices in home and work environments, we have shown how these applications differ from most other applications within the ad-hoc networks and especially the sensor networks research. We have then presented the specific needs of such applications: a strong security model for authentication and confidentiality, but also the means to fine-granularly distribute access rights (both physical and logical), which raises from the context-awareness of many of the target applications. We have shown how our middleware copes with both these major needs, and how it is thus well-suited for ad-hoc networking, context-aware applications in home and work scenarios.

We have further presented, from our experience in developing the middleware's architecture, the trade-off existing between ad-hoc networking, context-awareness, and fine-granularity of strong access rights. Numerous applications, such as several sensor networks applications, rely on a very pure form of ad-hoc networking, which comes along with various specific challenges for the networking and authentication algorithms. However, such applications do not require any form of context awareness, nor do they typically have to ensure the confidentiality of communication.

If, as in the case of our target applications, a high degree of fine-granularity is required, some compromises have to be made in terms of the spontaneity of the ad-hoc networking. We have solved this dilemma by allowing for a first step in which some of the devices define trust relationships among them, relationships that get then spread throughout the network and also autonomously influence the levels of trust when combining more such personal domain networks into a "multiple personal security domain." When, on the other hand, a high degree of security is needed, a compromise has to be made in terms of the context-awareness of the middleware itself. In our case, the middleware does not automatically generate keys from the environment, but relies on the strong keys provided by a certification authority. To put it bluntly, we have thereby sacrificed some of the possible context-awareness of the middleware itself, in order to provide the applications with a strong security, while at the same time allowing them to keep a high degree of context-awareness. We believe these insights to have an impact beyond our middleware solution, for further ad-hoc networking and context-awareness research in general.

References

1. Bohn, J., Coroama, V., Langheinrich, M., Mattern, F., Rohs, M.: Living in a world of smart everyday objects – social, economic, and ethical implications. HERA **10**(5), 763–786 (2004)
2. Brush, A.J.B., Inkpe, K.M.: Yours, mine and ours? Sharing and use of technology in domestic environments. In: J. Krumm, G.D. Abowd, A. Seneviratne, T. Strang (eds.) Proc. of UbiComp 2007, *LNCS*, vol. 4717, pp. 109–126. Springer, Innsbruck, Austria (2007)
3. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: Proc. of the CHI 2000 Workshop on Context-Awareness (2000)
4. Gellersen, H.W., Schmidt, A., Beigl, M.: Adding some smartness to devices and everyday things. In: Proc. of WMCSA, pp. 3–10. Los Alamitos, CA, USA (2000)
5. Hubaux, J.P., Buttyan, L., Capkun, S.: The quest for security in mobile ad hoc networks. In: N.H. Vaidya, M.S. Corson, S.R. Das (eds.) Proc. of MobiHOC'01, pp. 146–155 (2001)
6. Kawsar, F., Fujinami, K., Nakajima, T.: Augmenting everyday life with sentient artefacts. In: G. Bailly, J.L. Crowley (eds.) Proceedings of the 2005 joint Conference on Smart objects and Ambient Intelligence (sOc-EUSAI), pp. 141–146. ACM, Grenoble, France (2005)
7. Kidd, C.D., Orr, R., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E., Starner, T.E., Newstetter, W.: The aware home. In: N. Streitz, J. Siegel, V. Hartkopf, S. Konomi (eds.) CoBuild'99, *LNCS*, vol. 1670, pp. 191–198. Springer, Pittsburgh, PA, USA (1999)
8. Mattern, F.: The vision and technical foundations of ubiquitous computing. Upgrade **2**(5), 2–6 (2001)
9. Matushima, R., Venturini, Y.R., Sakuragui, R.R.M., Carvalho, T.C.M.B., Ruggiero, W.V., Naslund, M., Pourzandi, M.: Multiple personal security domains. In: S. Onoe, M. Guizani, H.H. Chen, M. Sawahashi (eds.) Proc. of IWCMC, pp. 361–366. Vancouver, Canada (2006)
10. Robinson, P., Beigl, M.: Trust context spaces. In: D. Hutter, G. Mller, W. Stephan, M. Ullmann (eds.) Proc. of Sec. in Pervasive Computing, *LNCS*, vol. 2802, pp. 157–172. Springer (2003)
11. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Proc. of WMCSA, pp. 85–90. Santa Cruz, CA, US (1994)
12. Venturini, Y.R., Sakuragui, R.M., Matushima, R., Carvalho, T.C.M.B., Ruggiero, W.V., Naslund, M., Pourzandi, M.: Security enforcement layer for security domain. In: Proc. of I2TS'2005, pp. 19–26. Florianopolis, SC, Brazil (2005)
13. Weiser, M.: The computer for the 21st century. Scientific American **265**(3), 66–75 (1991)
14. Zhou, L., Haas, Z.J.: Securing ad hoc networks. IEEE Network **13**(6), 24–30 (1999)