Privacy in communication among pervasive objects: methodologies for securing the exchange of sensible information

Pier Luca Montessoro, Davide Pierattoni, Sandro Di Giusto
University of Udine, Department of Electrical, Management and Mechanical Engineering
I-33100 Via delle Scienze 208, Udine, Italy
{montessoro, pierattoni, sandro.digiusto}@uniud.it

ABSTRACT

This paper discusses the main issues regarding the reliability and the efficacy of standard cryptographic techniques applied to pervasive computing. First of all, we describe a set of scenarios, where social interactions are supported by small-sized pervasive devices. These are expected to manage and exchange personal and sensible information with context-specific requirements. We present a general methodology that helps identifying the most suitable model for securing data storage, processing and communication. Three protocols and systems are formally described and compared in this framework, in respect of model's complexity and security capabilities.

Keywords: pervasive computing, cryptography, privacy, social interactions.

1 INTRODUCTION

Trying to foresee how pervasive devices will share and exchange information with each other, there are important social influences that must be taken into account. Pervasive computers will often deal with personal information: what perception of safety do users expect from such applications? What are the requirements that pervasive systems [1] should guarantee in terms of confidentiality, data protection and privacy?

Historically, RFIDs have been the early devices presenting pervasive functionalities [2]. The spreading of this technology has some consequences in terms of security and privacy protection, which is the most common objection moved by its detractors [3]. However, such applications also raise significant privacy concerns in the minds of people, whether the concerns are grounded in reality or not. For example, individuals do not wish to have their movements available to everyone. Finding a solution to those concerns is of considerable importance if pervasive environments are to be widely used.

Nevertheless, there are various techniques that may enforce the security of such distributed applications, without affecting their flexibility. A fascinating paradigm is the delocalization of security agents, together with secure communication protocols.

2 THE CONTEXT

For a better comprehension of the main objectives of this work, it is useful to describe three application scenarios that would deal with sensible information, and thus could benefit from the proposed methodologies [4]. By the way, these scenarios are not the only considerable ones, but let us treat them as examples.

In the following sections, a *Pervasive Object* (PO) indicates a user-owned device, which performs some computation, has a local storage capability and may communicate wireless with other devices. For example, a PO will be a small-sized personal object, like a pendant, a key chain or a bangle, with limited processing power. Our prototype of Pervasive Object is shown in Figure 1: this is a tiny board with 8-bit microcontroller and IEEE 802.15.4-compliant wireless interface.

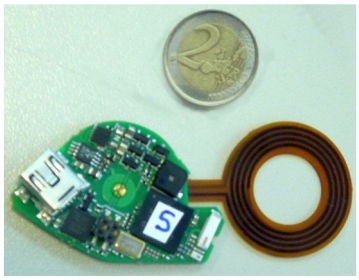

Figure 1: the prototype of small-sized pervasive device, with 8-bit microcontroller and IEEE 802.15.4-compliant wireless interface

Moreover, the *Certification Authority* (CA) represents an active entity or *agent*, either on a single node or distributed on a pool of servers, which is delegated to manage the security and to protect the privacy of the information exchanged among POs.

The fundamental assumption in this model of privacy management is that a user authorizes the divulgation of his/her personal data to another user only if the counterpart:

- is trusted
- is also divulgating his/her own data.

In general, both aspects are managed and guaranteed by the CA. Some applications could rely on personal data that are divulgated in a one-way communication; in such cases, the CA assures the identity and the authorizations of the persons to whom the collected information gets revealed.

2.1 Application scenarios

In our world, the use of communication technologies like phone, fax and email has become commonplace. Despite this fact, most human interactions still occur when people meet face-to-face. Frequently we exploit such meetings cooperating with other people and pursuing and advancing our own goals. For example, we purchase items from a salesperson, organize schedules with co-workers, or make travel arrangements with friends at home.

2.1.1 Car pooling

The car pooling principle is to optimize the utilization of the means of transport by the ride sharing among people who usually cover the same route [5]. Current car pooling management services are powered by centralized database systems and rely on Web-based interfaces to interact with users. This has some limitations: first, the scalability and the overall efficacy of a centralized system get lower when the number of users increases. Secondly, the user-driven approach in interfacing users to the management system is still tricky and few practical, since users spend time inserting their own data and verifying correspondences, if any. Last but not least, suspicion and mistrust are not so easy to defeat.

The basic idea for Pervasive Car Pooling is that those people who cover every day the same route at nearly the same time, have a good chance to meet close together sometimes.

This kind of neighborhood is often ineffective because people do not exchange information about their daily trip, and most of the time they do not communicate at all (think about an elevator, a canteen or a reception hall of a large enterprise). But a pervasive agent can automatically exchange the daily trip information and help discovering the possible affinities in an imperceptible and anonymous fashion.

Integrated in some little and portable thing – a trinket or a bracelet, for instance – POs are equipped with a short-range wireless communicator. POs can disseminate the average daily route covered and shareable by their owners: that is, the *datatrip*. At the same time, each PO collects from peer devices the same information disseminated by neighboring people. Pervasive car pooling is also easy-to-use: once data to be shared by a user's pendant has been set up, the user has nothing else to take care of. Setup can be performed both in an automated and manual fashion: in the former case, the datatrip can be evicted and refined automatically by the PO entity, collecting localization events from GPS or WiFi access points or special radio tags "on the road", and thus discovering the daily path. Alternatively, a user is also allowed to specify his/her preferred, "hand-made" datatrip to be shared among the service users, without requiring any additional infrastructure.

This silent information exchange produces in background a collection of messages inside each user's PO that are ready to be analyzed for affinity.

Controlled disclosure and privacy of exchanged information are guaranteed to the service subscribers by one or more trusted entities, named *Certification Authorities*. The problem of mistrust is solved by the fact that all the users that join the service are identified and recognized by a trusted corporate body, which owns and manages the CA and assigns POs to registered users.

As will be better described in the following chapters, communication among POs is anonymous, since data packets are encrypted according to a multi-layer enveloping scheme. In particular, each exchanged datatrip is signed and encrypted by the originating PO and may be decrypted only by the referring CA.

Communication with the CA happens from time to time and is necessary for the decryption of collected datatrips and their analysis, thanks to a trusted entity. Data transfer between a PO and the CA relies on a secure communication; for instance, it can be performed during battery recharge, through the PO's docking station connected to a networked PC.

CA searches for full or partial affinities, and performs the notification to those pairs or groups of subscribers, who matched the affinity criteria, In order to benefit of the trustworthy intermediation offered by the CA, the definitive arrangement of the *groups* of ride sharers should be done at the right time. For this reason, instead of sending back to the user a real-time notification that a match is found and what are the names of the candidates, the CA starts an off-line procedure of agreement. This way CA acts like a trusted agent and alerts via e-mail and/or SMS the involved subscribers to the opportunity of sharing the ride with someone else. Only after receiving the acceptance from all the concerned parties, CA discloses the minimal set of contact information to let people get in touch.

Each CA may also take part in a *federation* of trusted CAs that aggregates a large number of subscribers from different associated service providers. A safe procedure is also required for

managing groups when affine subscribers belong to distinct CAs in the federation.

Even if the number of members of a pervasive car pooling service increases, PO's short-range wireless communication confines the exchange of datatrips to those users, who find themselves in physical proximity at least one time. Let us observe that this condition acts like a constraint for reducing the space of events to be inspected for finding the local optimum solution for the affinity search problem.

2.1.2 Search for affinities

A further scenario, which generalizes somehow the previous one, involves the search for personal affinity among people. This could be the engine of various applications: for entertainment purposes, let us think of a discotheque or a club. But affinity might be discovered between a person's lifestyle and the books of a library.

Each user stores in his/her own pendant a set of personal information – a personal profile with hobbies, preferences, lifestyle, and so on – that are freely divulgated by this personal device. The other users' POs will receive the information spread by each neighboring PO; if a PO is in coverage of a CA (for instance, a WiFi Access Point) it sends to the CA its own personal information together with the harvested one. Being that information is encrypted, only the CA can decode PO-harvested data.

If the CA reveals significant affinities between two users, it sends a *match found message* to the PO of each corresponding user; even if match found messages were broadcasted, encryption allowed messages to be decoded only by the destination POs.

At this point, a matching user may decide to reveal his/her identity, informing the CA that he/she wants to meet his/her fellow. After receiving the authorization from both users, the CA informs them about the personal similarity and their respective contact information.

This scenario is not brand new [6][7][8], and some implementations of search for affinity have already been produced. Nevertheless, most existing systems do not deal enough with privacy and anonymity problems; further, they are based on pre-existent communication systems – like Bluetooth – whose security is inadequate for supporting this model of information exchange.

Let us observe that the information security level required in this scenario is lower than in the previous one. In fact, any personal and behavioral information is voluntarily selected by a user that decides to share something with the others; further, certain data are typically less critical from the point of view of the personal safety in comparison to the information on a user's own moves. On the other side, this scenario is more lightweight and suitable for entertainment applications.

2.1.3 Health and control

Much development work is ongoing addressing technologies and their application in the health domain [9], in order to achieve solutions that are non-invasive to everyday life and work. But by property of its confidentiality, the information exchanged by pervasive objects in this scenario must be kept under strict control.

Let us think about a network of wireless sensors that notice the pressure or the cardiac pulsation of a patient, or an accelerometer that monitors the mobility of a senior citizen in a nursing home. These POs must periodically communicate their readings to the CA – i.e., a centralized monitoring system – or even simply to another neighboring PO – for instance, the PO of a nurse or a physician, the PO embedded in medical equipment, and so on. It is expected that such information won't be transmitted in clear and therefore the system must guarantee the safety of communication. Moreover, it is necessary to certify the identity of the receiver that will decode the personal information.

In this scenario, many would be the advantages of a lightweight but robust system, in which PO sensors broadcast their own encrypted and digitally signed information. Each PO should also converse with other POs through the intermediation of an authentication agent in the CA, which is the only entity allowed to decrypt data sent from a PO.

2.2 General requirements

Starting from these examples, some fundamental requirements clearly emerge in the field of the communications among small pervasive devices.

Communication mode. In most cases connection-oriented protocols are too complex, since the amount of information to be exchanged is very small, it is spread in a short range transmission and it is not required to certainly reach a given destination.

RF and wireless technology. Protocols like Bluetooth, ZigBee, WiFi provide functionalities like peer pairing and connection control, requiring therefore large power consumption and sensitive loss of anonymity and safety. In the proposed scenarios instead a more simple RF communication is supposed: this could be based still on the physical layer of IEEE 802.15.4 but with ad-hoc layer-2 protocol implementation, giving just synchronization and collision detection.

Anonymity. It is often necessary to keep the anonymity of communication among pervasive devices, while features like authentication and intermediation (proxying) are delegated to a certified and trusted management entity.

Authentication. At the same time, the system is required to guarantee the sender's authenticity when a pervasive object receives the information. This is achieved through authentication performed by the CA, which is based on symmetric or asymmetric

cryptography, or both. The most effective example in this case is surely the health care scenario, where the control entity must not trust data received from those POs, whose authenticity is not guaranteed.

Latency. Some applications do not need real time communication and/or immediate responsiveness, being more focused to the intrinsic relevance of the content, rather than to the necessity of immediately know the results of the elaboration. When dealing with car pooling, the advantage of instantaneously identifying affine users and putting them into contact when they are still on the road is really less important than keeping high their sense of safety and anonymity. Time constraints instead are a little more critical for health care and entertainment purposes; anyhow, the system will not exceed a maximum latency, which could be a few minutes for affinity searches in entertainment applications, and a few seconds in health care and monitoring services.

Computation and complexity. From the user's point of view, everything should look as simple as possible. In fact, a citizen interested in the Pervasive Car Pooling service reaches the car pooling agency, signs a service acceptance agreement, and takes his/her own PO. Each Pervasive Object may be provided with a helper device, named *User Interface*: in the current vision, the UI is a docking station for the PO that gets connected with a Personal Computer – for instance, through the USB interface – plus dedicated software. UI recharges PO's internal battery. Nevertheless, UI's software assists the user during the PO setup, with wizards for creating, modifying and storing his/her encrypted information on the PO. The reason of such a UI in the model is not just to make things easier for the user: it also acts like a calculation helper for the PO. In fact, the UI could perform all the cryptographic features needed to encrypt user's information. This makes POs compatible with low-power, low-cost technological constraints.

3 THE PROPOSED TECHNOLOGY

To solve the problems previously described, we realized a model with two strata of communication: on the first stratum, Pervasive Objects – or *nodes* – are peers that may communicate with each other in unacknowledged mode, and without the possibility of decoding messages (then, without real interaction). On the second stratum, single POs converse with the certification entities (CAs) furnishing them the necessary functionalities of decoding and authentication, in order to access the information contents exchanged at stratum 1.

The system is thus based on a peer-to-peer, unidirectional infrastructure at stratum 1, and on a trusted hierarchy with bidirectional communication at stratum 2 [10][11]. The overall advantages of this model are undoubtedly notable, even if strictly related to the applications and scenarios presented in the previous section.

3.1 How the system works

The main scope of the system is to put the security management outside the POs; this is possible by means of a third party, the CA entity. At stratum 1 the system requires POs to communicate with each other in unacknowledged mode, sending the information they are allowed to distribute and eventually receiving information from other nodes. At this level, nodes are stores of collected information but they have no capability of decoding and interpreting its content.

We define at stratum 1 two communication phases: harvesting and dissemination. The former regards data collection, the latter regards data distribution among peer entities. Figure 2 shows how harvesting and dissemination are performed by neighbor POs in the pervasive car pooling scenario.

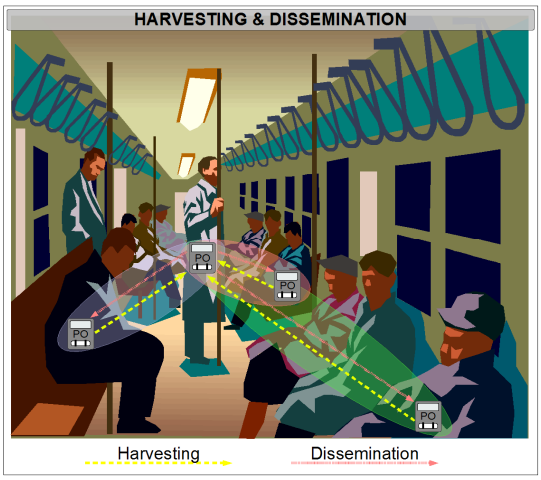

Figure 2: pervasive objects and stratum-1 harvesting and dissemination phases.

How CA agents intervene in decoding and managing information depends on the reference scenario. For instance, in Pervasive Car Pooling each PO sends to the CA agent the whole set of collected (*harvested*) messages; this stratum-2 exchange takes place with no time constraints, i.e., every time a communication channel is made available between the PO and the CA. On the other hand, low latency in health care applications requires the infrastructure to provide a diffuse and fast stratum-2 communication between POs and CA entities. Let us think of a large deployment of devices very similar to WiFi Access Points, whose scope were not just giving a mere connectivity towards POs, but providing them a network of *trusted* computation resources that works for them. The trusted resource deciphers, certifies and elaborates information collected by each PO in the system, and also sends back results and/or alerts, if needed.

3.2 The protocol stack

The heart of our system is a protocol stack for a multi-layered, secure communication among POs. Packets at each layer are transmitted and received in a completely anonymous and illegible form for those devices that are not expressly recognized and authorized in the trusted system [12].

The stack embeds double-key asymmetrical cryptography – e. g. ECC or RSA – and relies on digital signature techniques – e.g. ECC, RSA or DSA. Furthermore, cryptography [13] [14] and data hashing are applied in the system for data storage and management.

The main concept is thus forging anonymous, secure and authenticable packets. To achieve this, a *three-layer model* is required, as shown in Figure 3.

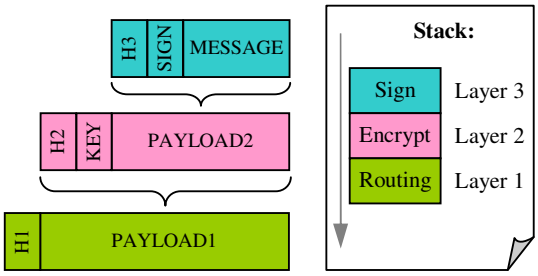

Figure 3: Protocol stack for trusted multi-layer communication.

A basic condition is that each PO in the system has its own pair of asymmetric keys (a public key $PUK|_{PO}$ and a private key $PRK|_{PO}$) that are regularly provided by the trust management authority (ideally, who governs the application: for instance, the administrator of the car pooling service). $PUK|_{PO}$ is then saved in the management authority's database and also stored in the CA agent that will provide services to the particular PO by decoding its packets. $PRK|_{PO}$ is kept inside the PO or, alternatively, in a external software that could be used to program PO's functions (for instance, a software bundled with the UI device for managing and configuring the PO from a Personal Computer).

Similarly, each CA agent must obtain from the trust management authority its own pair of asymmetric keys (a public $PUK|_{CA}$ and a private one $PRK|_{CA}$). $PUK|_{CA}$ is public domain, and will be stored in each PO that will refer to that particular CA; $PRK|_{CA}$ is kept inside the agent device.

After these premises, the meaning of the various levels in the stack is briefly explained as follows.

Layer 3 – "Sign". At this layer, each PO entity deals with the authentication of its own message, i.e. of the concrete information shared by the pervasive application. This is achieved by a hash algorithm that summarizes the message; hash data is then digitally signed with a signing algorithm (S) by means of the private key ($PRK|_{PO}$) of the sender PO:

$$SIGN = S\left(INF_3, DATA_{Clear}\right)_{PRK}^{PO} \tag{1}$$

The signature ($SIGN$) is then appended to the payload, which is made of clear application data ($DATA_{Clear}$) and application protocol information (INF_3); together with a layer-3 header H_3, this becomes the layer-3 message:

$$DATA_3 = H_3, SIGN, INF_3, DATA_{Clear} \tag{2}$$

Moreover, H_3 contains the unique ID of the object, that is, the label that allows the decoding agent to recognize the sender and to get its public key from the CA's database.

Layer 2 – "Encrypt". This layer provides encryption and decryption features, making layer-3 packet contents completely anonymous and illegible. A symmetric cryptography – based on algorithms like AES, 3DES, Blowfish, … – is thus performed on the whole layer-3 message, encrypting it (SC) with a temporary session key $SK_{SESSION}|_{PO}$ which is random generated. The same key is then encrypted with an asymmetric algorithm (AC) using the CA agent's public key $PUK|_{CA}$ and appended to the SC-encrypted layer-3 message. Layer-2 packet is thus:

$$DATA_2 = H_2, AC\left(SK_{SESSION}|_{PO}\right)_{PUK}^{CA}, \ldots \\ \ldots, SC\left(DATA_3|_{PO}\right)_{SK_{SESSION}}^{PO} \tag{3}$$

where PO introduces some protocol information in the layer-2 header H_2.

Layer 1 – "Routing". This layer provides routing information to be used by intermediate agents. This is necessary for complex systems, where decoding agents may also act as intermediate entities, when they receive packets from POs that refer to another agent in a federation of CAs. This layer enforces scalability and flexibility of pervasive systems, supporting large numbers of POs and CA agents, and allowing hierarchical organization of services and trust management authorities (see Section 5).

In this approach, the anonymity of the original sender is conserved by upper-layer protocols. In order to route packets among CAs, if needed, layer 1 appends some clear – i.e., unencrypted and unsigned – information H_1 to the layer-2 message:

$$DATA_1 = H_1, INF_1, DATA_2 \quad (4)$$

where H_1 represents the layer-1 protocol header and carries routing information. Moreover, INF_1 is an optional field which is suitable for applications that could benefit from sharing part of the personal information in clear form – i.e., unencrypted and unsigned.

4 APPLYING THE METHODOLOGY

Figure 4 shows the structure of generic information packets and summarizes the encryption and decryption phases performed respectively by the PO (or the UI helper, if the PO had no computation power enough) and the CA.

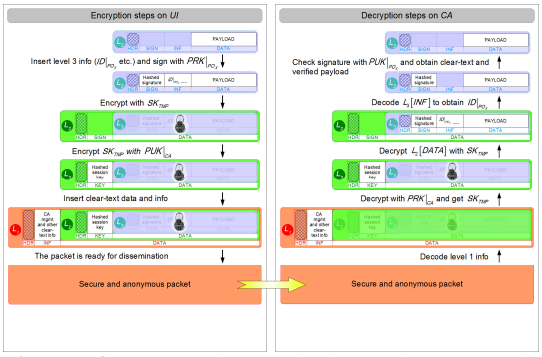

Figure 4: Encryption and decryption steps in pervasive, multi-layer trusted communication flows.

In order to satisfy both security and computation requirements of a large variety of applications, the proposed methodology provides three different versions of multi-layered, secure communication among pervasive objects: Robust, Flexible and Light.

4.1 Robust version

Robust version was designed with Pervasive Car Pooling in mind but it is also suitable for all those scenarios, where POs are not able to communicate frequently with a CA. In this case, assuring the highest security and privacy level is always more important than satisfying latency constraints. Let us note that sent packets are always anonymous and authenticable.

Communication between POs is unidirectional and offline, performing stratum-1 harvesting and dissemination phases. Stratum-2 communication is then represented by a secure communication between the PO and the CA; the endpoint of the stratum-2 communication may be the PO itself or an external helper device.

In fact, the Robust version is suited for systems with soft speed requirements, allowing minimizing PO's computational and hardware requirements. In particular, this version allows cryptographic and stratum-2 functions to be performed on external helper devices, instead of directly on POs. This is the case of a Personal Computer having a PO docking station with communication interface.

A data packet to be disseminated by the PO is encrypted once in a time by the software on the PC, according to the multi-layer protocol scheme (see previous Section). Then the packet is stored in PO's memory. On the other hand, packets collected from the PO are downloaded to the PC via the UI for further stratum-2 communication with the CA, each time PO's battery is recharged.

From the CA perspective, when receiving a collection of packets from a given PO, it must perform a preliminary check, decoding H_1 header and reading the $ID|_{CA}$. If this value doesn't match its own ID, CA could route the involved packets to the legitimate agent in a CA federation (see Section 5). Otherwise, it performs the following operations:

1. First of all, CA decrypts the second part of the message by using its own $PRK|_{CA}$.
2. Then it decodes information stored in H_1 and identifies the PO by the $ID|_{PO}$ field.
3. Having the $ID|_{PO}$, it interrogates its own user database and gets the related $PUK|_{PO}$. The public key of the PO is used by the CA to check the layer-3 signature.
4. If also this step yields positive result, the CA elaborates received data – for instance, finding affinities, recording health details, and so on - according to the application requirements.

4.2 Flexible version

If we think on a search for affinity scenario or on a sensor-based health care system, it is necessary POs to communicate frequently and rapidly with a CA.

This version is hence more lightweight and requires less computation on a PO, performing also a faster communication with the CA.

To obtain this, the Flexible version introduces a preliminary phase, named *discovery*: in this phase, every PO and CA that are meeting each other for the first time must perform a mutual authentication procedure.

To ensure the maximum flexibility, we have defined three protocol flavors for the discovery phase.

4.2.1 Mutual authentication

This is the most complicated flavor, but it gives the highest level of security and anonymity.

This protocol requires that every CA_{LOCAL} entity has an active network link to communicate with the global reference server CA_{RIF}. Then:

1. After a pervasive object PO_X receives a

Hello packet from CA_{LOCAL} (from which it also discovers $ID|_{CA_{LOCAL}}$ and the related public key $PUK|_{CA_{LOCAL}}$), it generates two random values: $ID_{TMP}|_{PO_X}$ and $SK_{DISCOVERY}|_{PO_X}$.

2. PO_X generates an *AuthReq* packet (Authentication Request) to the local CA entity:

$$AuthReq = H_{Clear}, \underbrace{AC\left(SK_{DISCOVERY}|_{PO_X}\right)_{PUK}^{CA_{RIF}}}_{SK^{\otimes}_{DISCOVERY}}, \\ SC\left(SIGN_{PO}, ID|_{PO_X}, ID_{TMP}|_{PO_X}, \right. \\ \left., ID|_{CA_{LOCAL}}, PUK|_{CA_{LOCAL}}\right)_{SK_{DISCOVERY}}^{PO_X} \quad (5)$$

where in particular $SIGN_{PO}$ is obtained from:

$$SIGN_{PO} = S\left(ID|_{PO_X}, ID_{TMP}|_{PO_X}, ID|_{CA_{LOCAL}}, \right. \\ \left., PUK|_{CA_{LOCAL}}\right)_{PRK}^{PO_X} \quad (6)$$

that is a typical digital signature. This packet is then broadcasted.

3. When CA_{LOCAL} receives the packet, it decodes the H_{Clear} field and forwards the packet to CA_{RIF}

4. CA_{RIF} receives the packet and decodes H_{Clear} field, realizing that it has to validate the CA_{LOCAL} - PO_X pair. It decrypts the second field of the message, $SK^{\otimes}_{DISCOVERY}$, using his own private key $PRK|_{CA_{RIF}}$:

$$SK_{DISCOVERY} = AD\left(SK^{\otimes}_{DISCOVERY}\right)_{PRK}^{CA_{RIF}} \quad (7)$$

and with the obtained $SK_{DISCOVERY}$ decrypts all the rest of the message:

$$AuthReq_{DATA} = SD\left(AuthReq^{\otimes}_{DATA}\right)_{SK_{DISCOVERY}}^{PO_X} \quad (8)$$

5. CA_{RIF} searches in its own user database the PO_X registration data. If a match is found, CA_{RIF} checks the integrity of the message, performing the sign-check on $SIGN_{PO}$ with the $PUK|_{PO_X}$ obtained from a query in the user database. CA_{RIF} then starts the second authentication phase, where it checks $ID|_{CA_{LOCAL}}$ and $PUK|_{CA_{LOCAL}}$ coherency. If also this step gives a positive result, the CA_{LOCAL} - PO_X pair is now certified.

6. CA_{RIF} generates a first reply packet for PO_X:

$$AuthAck_{PO} = H_{AuthAck_{PO}}, ID_{TMP}|_{PO_X}, SC\left(ID|_{PO_X}, \right. \\ \left., SK_{SESSION}|_{CA_{RIF}}, Message|_{CA_{RIF}}\right)_{SK_{DISCOVERY}}^{PO_X} \quad (9)$$

where some important values are stored, like $ID|_{PO_X}$, a new random symmetric key $SK_{SESSION}|_{CA_{RIF}}$ and an optional *Message* for PO_X. This packet is sent to CA_{LOCAL}, which broadcasts it to the reachable POs.

7. When PO_X receives the $AuthAck_{PO}$ packet, it simply realizes to be the right receiver by reading the $ID_{TMP}|_{PO_X}$ clear field: this is possible because this field is not encrypted and its value matches the original copy of the random variable. PO_X then uses its own $SK_{DISCOVERY}|_{PO_X}$ to decrypt all the rest of the message. This way, PO_X also learns that CA_{LOCAL} is a certified local server for the requested application. From now on, PO_X can directly communicate with CA_{LOCAL} only by using $ID_{TMP}|_{PO_X}$ as identifier and $SK_{SESSION}|_{CA_{RIF}}$ as symmetric key for encryption.

8. CA_{RIF} generates also a reply packet for CA_{LOCAL}:

$$AuthAck_{CA} = H_{AuthAck_{CA}}, CA\left(ID_{TMP}|_{PO_X}, \right. \\ \left., SK_{DISCOVERY}|_{CA_{RIF}}\right)_{PUK}^{CA_{LOCAL}} \quad (10)$$

confirming that PO_X is a trusted and registered object for the requested application. In this message CA_{RIF} also informs CA_{LOCAL} about the details of the connection with PO_X, that are $ID_{TMP}|_{PO_X}$ and $SK_{SESSION}|_{CA_{RIF}}$.

9. From now on, PO_X and CA_{LOCAL} can simply communicate each other using only symmetric cryptography with $SK_{SESSION}|_{CA_{RIF}}$.

4.2.2 One-way explicit authentication

This protocol flavor was studied for

environments and applications where it might be difficult to set up a permanently active network link among CAs. Perhaps, a side effect of relaxing this constraint is that it becomes impossible to obtain a mutual form of authentication, since CA_{LOCAL} cannot own the entire user database. For this reason, only one way-authentication is performed: in particular, only CA_{LOCAL} is authenticated by PO_X.

The basic idea of this protocol is to define a specific role, $CA_{LOCAL-PROX}$, which consist of a small device with a very short-range wireless communication capability (about few centimeters, like in the case of proximity sensors) and could be located in a specific position, for instance, at the checkpoint entrance of a building. This protocol flavor works as follows:

1. When the user gets into the range of the $CA_{LOCAL-PROX}$, his/her PO_X receives a *Hello* packet from it. In this advertisement, each PO_X discovers $ID|_{CA_{LOCAL}}$ and $PUK|_{CA_{LOCAL}}$.

2. PO_X generates an *AuthReq* packet (Authentication Request):

$$AuthReq = H_{Clear}, AC\left(SK_{TMP}|_{PO_X}\right)_{PUK}^{CA_{LOCAL}}, \\ , SC\left(ID_{TMP}|_{PO_X}, ID|_{PO_X}, Message|_{CA_{LOCAL}}\right)_{SK_{TMP}}^{PO_X} \quad (11)$$

and sends it to $CA_{LOCAL-PROX}$. In this packet, PO_X inserts two new random values: $ID_{TMP}|_{PO_X}$ and $SK_{TMP}|_{PO_X}$. Furthermore, it appends an optional *Message* field to $CA_{LOCAL-PROX}$ containing piggybacked information, if needed, according to the application requirements.

3. When CA_{LOCAL_PROX} receives the *AuthReq* packet, it reads H_{Clear} header and obtains $SK_{TMP}|_{PO_X}$ by decrypting the second field through its own $PRK|_{CA_{LOCAL}}$. After that, CA_{LOCAL_PROX} stores the extracted details about PO_X in the user database, which is shared with all CA_{LOCAL} entities in the local system. In particular, stored information are $ID_{TMP}|_{PO_X}$, $SK_{TMP}|_{PO_X}$, $ID|_{PO_X}$ and *Message*.

4. As last operation performed during the discovery phase, CA_{LOCAL_PROX} informs the user (but not necessarily through the PO_X) about the completion of the discovery phase, for example with a text on a display, a green light on the turnstile or an acoustic signal.

4.2.3 One-way implicit authentication

The last protocol flavor is suitable for scenarios where security and anonymity requirements aren't so pressing, preferring instead a lightweight and faster infrastructure.

In this case, the basic idea is to share the same, "universal" key pair $PUK|_{CA_{UNIV}}$ - $PRK|_{CA_{UNIV}}$ among all the CA_{LOCAL} authorities belonging to the same service. Nevertheless, every PO_X stores in its local memory the $PUK|_{CA_{UNIV}}$. To enforce security, the system could perform a key update procedure with a periodic generation and distribution of a new key pair, reducing the risk of corrupted or stolen keys.

This approach may be described as follows:

1. When PO_X enters in a service area, it receives a *Hello* packet from the nearest CA_{LOCAL}, from which it discovers the $ID|_{CA_{LOCAL}}$ value that identifies the local CA.

2. PO_X generates an *AuthReq* packet in the following manner:

$$AuthReq = H_{Clear}, CA\left(SK_{DISCOVERY}|_{PO_X}\right)_{PUK}^{CA_{UNIV}}, \\ , SC\left(ID_{TMP}|_{PO_X}, ID|_{PO_X}, Message|_{PO_X}\right)_{SK_{DISCOVERY}}^{PO_X} \quad (12)$$

This step is similar to the explicit authentication.

3. When CA_{LOCAL} receives the packet, it decrypts the packet using the $PRK|_{CA_{UNIV}}$ shared among CAs, and stores the extracted details about PO_X in its local user database.

4. The CA_{LOCAL} generates an *AuthAck* (Authentication Acknowledgement) and broadcasts it to POs with the following content:

$$AuthAck_{PO} = H_{AuthAck_{PO}}, ID_{TMP}|_{PO_X}, SC\left(ID|_{PO_X}\right. \\ \left., SK_{SESSION}|_{CA_{LOCAL}}, Message|_{CA_{LOCAL}}\right)_{SK_{DISCOVERY}}^{PO_X} \quad (13)$$

5. PO_X receives the packet from CA_{LOCAL} and, from now on, it will communicate with CA_{LOCAL} using symmetric cryptography with the couple of values: $SK_{SESSION}|_{CA_{LOCAL}}$ and $ID_{TMP}|_{PO_X}$.

4.3 Light version

Light version aims at the minimization of the overall communication and protocol overhead; at the same time, it increases the computational performance and reduces the latency of the system.

The basic idea is to simplify the *discovery* phase of the Flexible protocol version by removing the

authentication procedure. This means that PO_X is not really able to authenticate a CA that sends him a *Hello* packet; nevertheless, it can perform anyway a secure and virtually anonymous data exchange with the CA.

Basically, the Light version starts from the usual *Hello* packet from the CA_{LOCAL}, which advertises its $PUK|_{CA_{LOCAL}}$. When PO_X receives this packet, it generates a new packet with $ID|_{PO_X}$, and two new random values: $SK_{TMP}|_{PO_X}$ and $ID_{TMP}|_{PO_X}$. This information is then encrypted with $PUK|_{CA_{LOCAL}}$ and broadcasted. When CA_{LOCAL} receives the message, it decodes and decrypts it with its own $PRK|_{CA_{LOCAL}}$ and then stores the extracted details about PO_X in its local user database.

From now on, CA_{LOCAL} and PO_X can communicate using symmetric cryptography with the values $ID_{TMP}|_{PO_X}$ and $SK_{TMP}|_{PO_X}$.

This model is suitable for those applications, whose constraints in terms of mutual trust between provider and consumer are not so critical, like in the case of "rough" systems, such as search for affinity in a discotheque or for other entertainment purposes.

5 COMPLEX SCENARIOS

As previously described, Robust and Flexible variants require PO authentication; nevertheless, there exist some scenarios where the decoding agent for a given PO could be not unique. In the pervasive car pooling, for instance, a single server acting as decoding agent for a huge number of POs would become unmanageable. On the other hand, for vulnerability reasons it would be unsafe to replicate and thus to share the same private key $PRK|_{CA}$ among the CA agents in a pool. A suitable solution is a distributed approach, with a protocol for managing agents and secure information exchange in a *federation* of Certification Authorities.

Current implementation is based on level-1 information in H_1 header: this identifies the decoding agent — i.e., the CA — that sends the message or is expected to be the receiver by the sending PO. When a CA agent receives a packet, it checks the CA identifier ($ID|_{CA}$) in H_1. If the $ID|_{CA}$ in the packet doesn't correspond with its own $ID|_{CA}$, the CA engages the *CA-discovery* and *CA-routing* phases.

First, *CA-discovery* requires the local CA agent to verify if the packet's $ID|_{CA}$ is in a local cache of the $ID|_{CA}$ keys of previously known, certified CAs in the federation. If not, the agent sends a request to the $CA_{MANAGER}$ asking if the $ID|_{CA}$ belongs to a valid federate, and requesting the contact data (like network address, hostname, digital certificate, …) of the federate CA. If the $CA_{MANAGER}$ answers positively, the local agent is able to contact the federate CA corresponding to the $ID|_{CA}$ through a secure network connection – for instance, a SSL-encrypted TCP connection. The local agent requests the federate $ID|_{CA}$ to send information for remote decoding. After confirm, local CA performs *CA-routing* phase by sending the PO-generated information on the secure channel to the counterpart $ID|_{CA}$, which decodes data following the model described in previous section. After processing, $ID|_{CA}$ may send back the results to the local agent, if needed by the application.

In particular cases, it could be necessary to enforce security and robustness of this scheme, so that information about POs would not be exchanged among federate CAs. In the car pooling scenario, for privacy reasons the federate $ID|_{CA}$ could send back to the local CA agent just the result of its elaboration, but never inform it about the personal information of the user referring to the served $ID|_{PO}$. To achieve this, status information or timestamps could be added in clear as part of the layer-1 protocol, to be managed during *CA-routing* phase. This would help local CAs tracking requests regarding foreign POs, and also taking countermeasures in case of communication failure, latency exceeded or timeout.

6 PERFORMANCE OF THE SYSTEM

To analyze more in detail the proposed system, we report some quantitative considerations regarding the trade off between security and overall performance.

As evicted before, the proposed methodology and all its protocol variants are very generic and can be applied on many different scenarios. This fact makes difficult to define an overall performance indicator, since it varies from application to application. Nevertheless, we try to address an evaluation of the overall performance in the very particular scenario of pervasive car pooling.

We can assume that a reasonable data packet dimension is about few hundreds of bytes: depending on the server federation complexity and the datatrip profiling and coding system, application-level data could reasonably vary between 200 and 500 byte. Considering the smallest data size, multi-layer trusted communication adds the following protocol

information:

- Level 3: in addition to application-level payload, we have to consider the SIGN field (32 bytes using standard SHA-256 digital signature), the INF field with 6 bytes (since it contains also the PO's ID) and the HDR field of 2 bytes. Level-3 packet amounts to 240 bytes.
- Level 2: the entire level-3 packet is encrypted using AES-128, obtaining a level-2 payload of 240 bytes; we have also the KEY field of about 32 bytes (using ECC-256) and an HDR of 2 bytes. Level-2 packet is thus of 274 bytes.
- Level 1: the payload is the entire level-2 packet; level 1 adds also an INF field of about 8 bytes and an HDR of about 2 bytes. The final packet amounts to 284 bytes.

This rough analysis provides a reasonable accuracy when considering the network overhead, i.e. the overhead introduced by the security- and trust-oriented communication. The additional cost in terms of packet dimension is of about 42% in the worst case. This is quite inexpensive if we consider that we are using a medium-high security-level (a 128 bit strong equivalent security [15]). Smaller dimensions of INF and HDR protocol fields are also possible in other application scenarios, reducing network overhead to lower values but maintaining the same security level.

7 CONCLUSIONS

The focus of this work is to present a set of scenarios where small-sized pervasive entities are expected to manage and exchange personal and sensible information.

Three main approaches with some flexible variants for securing data storage, processing and communication are thus formally described and compared, in respect of model's complexity and security expectations.

Table 1 summarizes the main features and requirements of each version of the proposed methodology:

		Version				
		Robust	***Flexible***			***Light***
			Mutual authentication	Explicit authentication	Implicit authentication	
Features	Active network link	Yes, but not permanently	Yes	No	No	No
	Authenticates *PO*s	Yes	Yes	No	No	No
	Authenticates CA_{LOCAL}	Yes	Yes	Yes, by proximity	Yes, by universal keys	No
	Requires a *discovery* phase	No	Yes	Yes	Yes	Yes
	Uses signatures during *data session*	Yes	No	No	No	No
	User anonymity level	Total	Almost total	Almost total	Almost total	Almost total
	Security level	Total	Total	Good	Fair	Partial
	Notes	It requires more computational resources than others (suitable for pervasive objects with a PC or PDA as helper devices)	Requires still a permanently active network link for CA_{LOCAL}	Less transparent to the user, until the completion of the discovery phase	The overall security level is strictly related to the security of the *CA* devices (because keys are universal)	It cannot offer strong authentication techniques (potentially exposed to aliasing attacks)

Table 1: Comparison among the different versions of the methodology

The affinity search methodology described here is part of an italian patent [16] and a pending application of international patent [17].

ACKNOWLEDGMENTS

Authors wish to thank Dr. Gianpietro Tecchiolli from Exadron - a subsidiary of Eurotech S.p.A., Amaro (UD), Italy - for his contribution in stimulating and supporting this research activity and for the development of the prototype.

REFERENCES

1. Davies, N.; Gellersen, H.-W.: Beyond prototypes: challenges in deploying ubiquitous systems. Pervasive Computing, IEEE Volume 1, Issue 1, Jan.-March 2002 Page(s):26 – 35

2. Riekki, J., Salminen, T., Alakärppä, I.: Requesting Pervasive Services by Touching RFID Tags, IEEE Pervasive Computing, vol. 5, no. 1, pp. 40-46, Jan-Mar, 2006.

3. Rieback, M.R., Crispo, B., Tanenbaum, A.S.: The Evolution of RFID Security, IEEE Pervasive Computing, vol. 5, no. 1, pp. 62-69, Jan-Mar, 2006.

4. Estrin, D., Culler, D., Pister, K., Sukhatme, G.: Connecting the Physical World with Pervasive

Networks, IEEE Pervasive Computing, vol. 1, no. 1, pp. 59-69, Jan-Mar, 2002.

5. Montessoro, P.L., Pierattoni, D., Di Giusto, S.: Designing a Pervasive Architecture for Car Pooling Services, Proceedings of the International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering. CISSE 2005.

6. Teng, C.-M., Chu, H.-H., Hsu, J. Y.-j.: Making Use Of Serendipity: A New Direction For Pervasive Computing From A Sociological View. Department of CSIE, National Taiwan University.

7. Beresford, A.R., Stajano, F.: Location Privacy in Pervasive Computing, IEEE Pervasive Computing, vol. 02, no. 1, pp. 46-55, Jan-Mar, 2003.

8. Bessler, S., Jorns, O.: A Privacy Enhanced Service Architecture for Mobile Users, percomw, pp. 125-129, Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05), 2005.

9. Drugge, M., Hallberg, J., Parnes, P., Synnes, K.: Wearable Systems in Nursing Home Care: Prototyping Experience, IEEE Pervasive Computing, vol. 5, no. 1, pp. 86-91, Jan-Mar, 2006.

10. Shih, E., Bahl, P., Sinclair, M.J.: Wake on wireless: an event driven energy saving strategy for battery operated devices. In Proceedings of the eighth annual international conference on Mobile computing and networking, pages 160--171. ACM Press, 2002.

11. Colon Osorio, F.C., Agu, E., McKay, K.: Tradeoffs Between Energy and Security in Wireless Networks.

12. Anderson, R., Needham, R.: Robustness principles for public key protocols. Cambridge University Computer Laboratory

13. Kakkar, P., Gunter, C.A., Cryptographic Reflection, Department of Computer and Information Science University of Pennsylvania

14. Network Associates, Inc.: Introduction to Cryptography

15. Barker et al.: NIST Special Publication 800-57, Recommendation for Key Management, January 2003, Revised on March 2007.

16. Montessoro, P.L., Pierattoni, D., Di Giusto, S.: Metodo per l'individuazione di affinità fra soggetti e relativo apparato, Italian Patent Application UD2005A000209

17. Montessoro, P.L., Pierattoni, D., Di Giusto, S.: Method to search for affinities between subjects and relative apparatus, International Patent Application WO2007065931