Trust and Authorization in Pervasive B2E Scenarios

Laurent Bussard[1], Yves Roudier[1], Roger Kilian-Kehr[2], and Stefano Crosta[1]

[1] Institut Eurecom, Network Security Team
2229, route des Crêtes BP 193
06904 Sophia Antipolis (France)
`http://www.eurecom.fr/∼nsteam/`

[2] SAP Labs France S.A., Corporate Research
805, Av Maurice Donat, Font de l'Orme
06250 Mougins (France)
`http://www.sap.com/research/`

Abstract. Many pervasive and ubiquitous application scenarios consider the interaction of users with surrounding devices offering services anywhere and anytime as one of the main future challenges. However, before this vision becomes reality, many security issues have to be solved. More specifically, the problem of trustworthiness of unknown devices is one of the major obstacles hindering the acceptance of pervasive applications. This paper focuses on solutions for *business-to-employee* scenarios, a particular sub-domain of the ubiquitous computing paradigm in which valuable *a priori* trust information is available. Mechanisms and protocols are introduced to assess the trustworthiness of devices federated around a mobile user, and to allow for the delegation of authorizations between such devices. The first results of a prototype implementation are finally presented.

1 Introduction

Many researchers consider the general progress in the development of microelectronics leading to small integrated devices, small high-resolution displays, wireless communication technologies, etc. as the driving force behind the ubiquitous wave [1]. It is commonly believed that computers are likely to disappear in many application contexts and that people will not carry around heavy-weight devices such as laptops, but instead use devices and user interfaces they encounter in their environment, e.g. cars, airport lounges, airplanes, hotels, public zones, etc. Nomadic users will take advantage of such devices to communicate with their companies, access information on the Net, and perform everyday tasks. However, this is where security problems immediately arise: who gives a device access to some data? Under which assumption? And for what purpose?

1.1 Security in Pervasive and Ubiquitous Applications

The literature generally envisions pervasive and ubiquitous computing in an application-centric manner that is focused on particular user application scenarios for which authors provide partial solutions. Most often, security aspects are neglected and considered as disturbing factors in these scenarios. In contrast, we are convinced that pervasive approaches will only be successful if the associated security problems are solved. The only ubiquitous or pervasive applications that will be successfully deployed and developed in the future will be those who will protect users' assets in terms of security and privacy. Of course, bridging the gap between security requirements on one hand and user-friendliness on the other hand is a specially challenging task.

Security is a major concern for employees and corporations, who may be the most interested in the spreading of pervasive technologies as described above. A large company with a mobile workforce has a natural interest in protecting its digital assets. Hence, if a mobile employee decides, for instance, to read email on a public display, there must be mechanisms available and enforced in order to prevent the leakage of company-confidential information via that display.

Put differently, an enterprise must be able to enforce its corporate security policy in a systematic manner, wherever company-related operations may take place. Pervasive computing environments specifically raise the problem of how information flow (e.g. displaying, printing, copying) can be reasonably controlled. The major issues to be solved are the following:

– Each mobile employee needs a certain *root of trust*, e.g. a user-specific identity and security module such as a smart card with some added functionality such as a trustworthy display and input channels.
– Interaction with surrounding devices requires the *delegation of rights* from the user to these devices, e.g. to access corporate systems on behalf of the user.
– Prior to delegation of rights, a device's *trust level* must be determined in order to decide upon the amount, type, and duration of the privileges granted.

1.2 B2E and B2B Environments

Solving these issues, especially determining or estimating the trustworthiness of the surrounding devices is in general a hard problem with many non-technical issues involved. However, this paper focuses on business-to-employee (B2E) and business-to-business (B2B) settings. In this context, these issues may be solved in a much simpler way since there is usually sufficient *a priori* trust information available related to the relationship between an employer and his employees, a company and its devices, or between business partners.

How this information can be structured and exploited to develop a user-friendly security framework is the main contribution of this paper, which is organized as follows: Section 2 provides an in-depth study of the context and requirements of B2E security. In particular, it discusses the trade-off between security and user-friendliness and the trust model underlying our approach. Section

3 describes a security architecture for *distributed access control*, *trust level verification*, *data distribution*, and *authorization*. A prototype implementation based on devices interconnected through short-distance wireless links is described in Section 4 together with a sample application. Section 5 concludes this paper and gives an outline on future work items.

2 Problem Statement

Pervasive computing architectures in which many devices cooperate to achieve some function are developing quite fast nowadays. This section presents the rationale for security in such architectures.

2.1 The Pervasive Salesman Scenario

A salesman comes to visit commercial partners. He only carries a small trusted personal device (TPD) that could be a personal digital assistant (PDA), a watch, or even digital jewelry or some wearable computer. A positioning service provided by the building makes it possible for his TPD to guide him to the meeting room, opening the doors to the only rooms he is authorized to access. The visitor being alone in the meeting room, he can personalize the room illumination and have his favorite music played. While he is waiting for other participants to arrive, he wants to read his email on a workstation in the meeting room. He federates it with his personal device so that this terminal can have access to his corporate mailing box and show the list of emails received. The visitor reads a few non-classified emails, and then selects one email that is tagged as confidential. According to the security policy of the salesman's company, confidential data must not be displayed on any device not belonging to the company: the mail does appear on the smaller display of his TPD that is trustworthy. As other meeting participants arrive, their virtual business cards are displayed on the TPD. When the meeting starts, a space for securely sharing data is created between the meeting members. A slide show application in the visitor's TPD can then have all people in the meeting room sign an electronic non-disclosure agreement before it shows the visitor's corporate data on the local video projector. Even after the visitor has left the building, he has access to a list of the interactions performed with meeting participants. Back in his office, the salesman can browse classified data and emails on any corporate terminal. Because those displays are trusted, he can directly access confidential data.

2.2 Security Implications

The pervasive salesman scenario outlined in Section 2.1 shows that B2E applications can be expected to collaborate dynamically with surrounding devices. A federation of devices is defined as *a group of devices with different trust levels that collaborate towards the same application.* Security is an integral part of federations. By their very nature, federations of communicating devices are more exposed to attacks than plain mobile devices since an application running within a federation spans across devices owned by different entities.

This scenario entails multiple security implications. Users need some authorization to use local facilities. They also need some authorization to access corporate data remotely and a way to delegate specific rights to federated devices. Federations have to evaluate whether an execution environment is trustworthy. And last but not least, communication channels have to be protected, i.e. confidentiality, integrity, message authentication, and sometimes non-repudiation are necessary for applications used in a federation.

2.3 Trust Model

Corporate security policies define whether a given operation (e.g. dealing with confidential data) can be done by an appliance with an ascertained trust level.

Securing federations is about evaluating the trust level of each platform that takes part in the federation. This evaluation is not easy to achieve in general. If a person makes use of a terminal in a public place, it is impossible to assume that the terminal is trusted in any way without some additional information that makes up the trust model. In general, there is no relation that can be exploited between the user or his company and the owner of the terminal: this can be called an *open trust model* as opposed to an *a priori trust model*.

B2E context introduces assumptions that make it simpler to construct a workable trust model based on limited a priori trust information. First of all, public key based authentication is possible and meaningful since employees are directly managed by their corporation. In contrast, in an open trust model, there will generally be no authentication (or trust) infrastructure shared by all entities. Secondly, trust may be based on the partnership established between companies. The trust expectation regarding every partner's tasks and behaviors are contractual and can be translated into a security policy. Trust may also be based on the certification of devices, and specifically their level of tamper-resistance. Again compared with the open trust model, this assumption enables the automation of secure data distribution to different devices.

3 Security Architecture

This section presents the architecture developed to enable security features as described in Section 2.2. The notations used within this paper are defined in Tables 1 and 2.

3.1 Architecture Overview

Any employee has a trusted personal device that contains some credentials usable to prove his rights. Java enabled PDAs and cell-phones have been chosen for this purpose as this might become the platform of choice for developing on many appliances. This trusted personal device is part of a federation of surrounding devices that can be managed by other companies. The federation is used by the employee when accessing some resources protected by a corporate backend server. Access control is based on user's rights and trust level of each federated device.

Each device has its own asymmetric key pair and can be certified by its owner. Trust level evaluation and access control are based on those certificates. The backend and the TPD are in charge of enforcing the corporate security policy.

Access Control. Access to resources is protected at two levels (see Figure 1). The first access control layer verifies whether a given user is authorized to access the required resource. The user's rights are described in authorization certificates referencing the user's public key as an identifier. The corresponding private key is kept in the user's TPD or physically protected by a dedicated tamper-resistant module. SIM cards are specifically envisioned because of their ubiquity in mobile devices. Next, the second access control layer distributes data according to the trust level of each federated device. Each device has its own asymmetric key pair that may also be physically protected. Manufacturers, owners, or administrators of devices can install certificates providing useful information about the device trust level. When the trust level of a federated device is known, it becomes possible to decide whether it can access some confidential data or not.

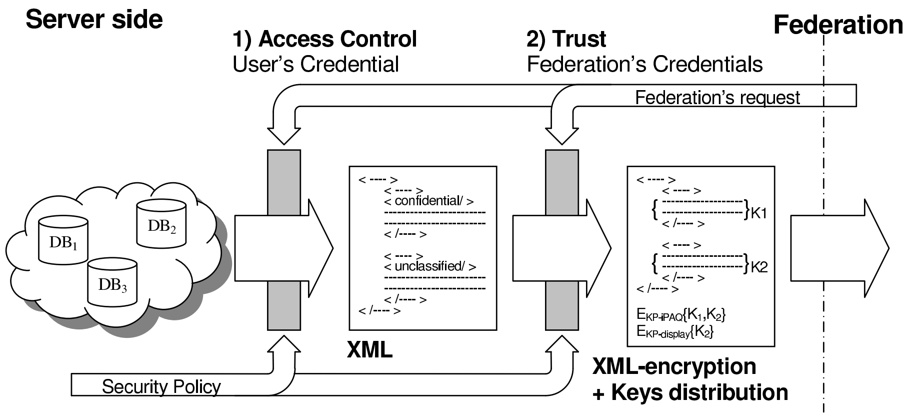


Fig. 1. Server side two stages access control based on employee's right and trust level of each federated device

Tamper-Resistant Module. Tamper-resistance of the TPD is also essential for enforcing the corporate security policy since it can hold private keys as well as the decryption keys of confidential data. It is very difficult to protect the execution of pieces of code [2] or even the evaluation of functions [8] in untrusted environments. A straightforward way to ensure that a device can be trusted is proposed by the Trusted Computing Platform Alliance (TCPA) [12]. The hardware is certified and can verify whether a certified kernel is running on top of it. This kernel controls the OS, which can check applications. This architecture makes it possible to prove that a given environment is running. As long as all layers are trusted (i.e. certified) and without implementation errors, it is possible to trust the environment. In other words, confidential data can be provided to any TCPA public terminal with the guarantee that those data will not be misused.

TCPA being unavailable as of now, a pragmatic approach relying on trust-based distribution has been chosen: data are distributed according to the trust level of each federated device. Trust level evaluation is possible because the trust model proposed in this paper is restricted to B2E scenarios. When B2B relationships exist, *a priori* trust is a realistic assumption. It is possible to base trust on the knowledge of the manager of a device. It is assumed that partner companies can be more or less trusted and that they correctly manage their devices (patches, antivirus, etc.) in order to protect themselves. Instead of proving the security of any involved device, data distribution is performed according to the trust that can be derived from established relationships.

Flexibility versus Security. On one hand, employees would like to use transparently any surrounding appliance such as a larger display embedded in a plane seat, a printer in an airport lounge, or location services offered by a building. On the other hand, the corporation has to protect its resources and prevent that an employee unintentionally reveals corporate data to untrusted and potentially malicious devices. The tradeoff between flexibility and security is based on the corporate security policy that defines whether a given device certified by an entity can be involved when getting access to non-public data. Such an open federation, in which devices owned by different entities are used, will have to be restricted to secure interactions, and yet remain fully usable.

In this architecture, access control must be enforced, devices have to be certified, and resources have to be tagged. Deploying such a solution is possible in a B2E application context where local PKI exists (without global CA) and where trust can be specified. This architecture offers a pragmatic approach to secure the use of potentially malicious environments in B2E applications. It ensures secure federations without loosing their flexibility and user-friendliness.

Notations. Table 1 defines the involved actors. E_2 is an employee of company C_2. He is visiting company C_1 and would like to federate his cell-phone D_2 with the partially trusted display D_1 in front of him in order to access corporate data R_2. E_2 must be granted the right to use D_1 and to access R_2 (Section 3.2). The trust level of D_1 is evaluated in order to distribute data to the federation $\langle\, D_2\ ,\ D_1\,\rangle$ (Section 3.3).

C_1	a company
C_2	another company working with C_1
E_1	an employee of C_1
E_2	an employee of C_2. He is called visitor when working in C_1
D_1	a device (e.g. wall display, printer, terminal, coffee machine) of C_1
D_2	a trusted personal device (e.g. PDA, cell-phone, smart card) of C_2

Table 1. List of actors involved in the protocols of this paper.

It is also possible to define subject roles E corresponding to a set of employees and object roles D corresponding to a set of devices [3].

3.2 User-level Access Control Infrastructure

In order to enforce security, an access control system to corporate data has to be set up. In this system, each employee receives a security profile (set of rights) to access resources or use other devices.

Access control is a classical problem that becomes complex when delegation (e.g. a device from the visited company) is required. To be compatible with the issuance of local rights, this paper proposes to use authorization certificates (similar to SPKI [6]). Delegation happens for instance when a secretary gives a visitor access to a meeting room for a few hours, as well as when a visitor authorizes a wall display to access some specific corporate data in order to display them. Short term and application specific rights may also be required.

PK_A	public key of entity A
SK_A	private key of entity A
$\mathrm{E_K}(m)$	plaintext m encrypted with key K
$\mathrm{h}(m)$	digest of data m using hash function h applied
$\mathrm{SIGN_A}(m)$	plaintext m signed with the private key of A i.e. $\mathrm{SIGN_A}(m) = \{m, E_{SK_A}(\mathrm{h}(m))\}$.
$cert_{CA \to A}$	certification authority CA certifies that the entity A has some attributes (e.g. rights, identity, owner). $\mathrm{CERT_A}(m) = \mathrm{SIGN_A}(m)$

Table 2. Cryptographic notations used in this paper.

Case 1: Accessing Corporate Database. Any employee E of company C receives an authorization certificate $cert_{C \to E} = \mathrm{CERT_C}(PK_E, R_E, deleg)$ where $R_E = \langle r_1, r_2, \cdots, r_n \rangle$ is a set of rights and *deleg* is the ability to delegate those rights. It is necessary to distinguish between delegation to other users (*deleg* tag) and delegation to devices. Indeed, when employee E wants to let a federated device D access a resource, he has to delegate some specific short-term rights. For instance, E will browse a file server on his cell-phone and authorize the terminal in front of him to access a given file in order to open it for editing. It does not mean that E is allowed to delegate rights to another employee. E provides $cert_{E \to D} = \mathrm{CERT_E}(PK_D, R_D)$ where $R_D \subseteq R_E$ is a subset of the employee's rights. It is important to define R_D as precisely as possible and to set a short validity in order to avoid unexpected access from federated devices.

When D accesses the resource r, it has to provide the following certificate chain:

$$\langle\ \underbrace{\mathrm{CERT_C}(PK_E, R_E)}_{\text{authorization}}\ ,\ \underbrace{\mathrm{CERT_E}(PK_D, R_D)}_{\text{auth. (delegation)}}\ \rangle \qquad \text{with} \qquad R_D \subseteq R_E$$

During access control, the links between the components of the chain are verified, the signature and validity of certificates are controlled, a challenge-response protocol is used to check whether D knows the private key SK_D corresponding to the public key PK_D embedded in the certificate $cert_{E \to D}$, and the server finally verifies that $r \in R_D$. When the whole verification has succeeded, trust level evaluation and trust-based distribution can start.

Case 2: Using a Surrounding Device. Authorization certificates are similarly defined to enable the use of surrounding devices that can be owned by another company. When a visitor E_2 enters the building of the partner company C_1, he needs some rights to benefit from the services offered by the environment (like using wall displays and printers, getting a map of a building and his location, opening a door to access a room, getting a lunch, etc.). Those rights can be based on an agreement between the visitor's company and the visited company, or can be delivered when the visitor enters the building. Depending on the security policy, rights can be provided automatically or require that a human being E_1 be involved. For instance, the policy may specify that anybody physically in the building can use the wall display but it is necessary to register to get access to some restricted areas. Different chains are possible. For instance, company C_1 authorizes company C_2 to use its wall displays D_1. This right can be delegated to employees:

$$\langle\ \underbrace{\mathrm{CERT_{C_1}}(PK_{C_2}, D_1)}_{\text{authorization}}\ ,\ \underbrace{\mathrm{CERT_{C_2}}(PK_{E_2}, D_1)}_{\text{auth. (delegation)}}\ \rangle$$

Or, company C_1 authorizes a secretary E_1 to delegate rights to registered visitors. E_2 receives some rights:

$$\langle\ \underbrace{\mathrm{CERT_{C_1}}(PK_{E_1}, D_1, deleg.)}_{\text{authorization}}\ ,\ \underbrace{\mathrm{CERT_{E_1}}(PK_{E_2}, D_1)}_{\text{auth. (delegation)}}\ \rangle$$

The certificate chain can be longer but it can be assumed in B2E environments that certificate chains remain short enough. Their verification can thus be simplified by providing well-formed chains instead of individual certificates that would have to be assembled by the verifier on the corporate side.

3.3 Device-Level Access Control Infrastructure

It is not sufficient to base access control on users' rights: characteristics of federated devices have to be taken into account.

Trust Level Verification. New types of capabilities have been defined to evaluate the trust level of federated devices. Device D is owned and managed by company C (say for instance a wall-display in a meeting room that is physically protected). Company C provides an ownership capability $cert_{C \to D} = \mathrm{CERT_C}(PK_D, \text{ownership})$ to device D.

Agreements between companies are necessary to formally define trust relationships. Such agreements are also defined by certificates. Company C_2 is a partner of C_1. Employees of C_2 frequently need to work in C_1's offices and use local facilities D_1. Because C_2 trusts C_1, they can have the following agreement: $cert_{C_2 \to C_1} = \mathrm{CERT_{C_2}}(PK_{C_1}, \text{trust=confidential})$ meaning that devices owned by C_1 can be used to deal with confidential and unclassified data.

The trust level of a federated device is evaluated thanks to the chain:

$$\langle\ \underbrace{\mathrm{CERT_{C_2}}(PK_{C_1}, \text{trust=confidential})}_{\text{agreement}}\ ,\ \underbrace{\mathrm{CERT_{C_1}}(PK_D, \text{ownership})}_{\text{ownership}}\ \rangle$$

Here, it is assumed that all devices managed by a specific entity have the same trust level. It is possible to define more precise trust relationships involving other parameters such as tamper-resistance, location, etc.

Data and Key Distribution. Data distribution is based on the rights of the requestor and on the trust level of the devices involved. Suppose that a set of resources $R_{req} = \langle r_1, \cdots, r_2 \rangle$ are requested from C_1's server by an employee E_1 using a federation $F = \langle D_1, D_2, \cdots \rangle$. As shown in Figure 1, the access control is done in two stages: user's authorization defines which data can be retrieved and the trust level of each federated device combined with the classification tags of resources defines which device can deal with given data. The first access control layer requires the user's authorization chain and delivers authorized resources $R_{AC} = R_{E_1} \cap R_{req}$.

Each device can receive any data in an encrypted form but can only retrieve keys corresponding to its trust level. Resources R_{AC} are encrypted according to their classification $cl \in CL$. For instance, $CL = \langle \text{unclassified} = 0, \text{confidential} = 1, \text{secret} = 2 \rangle$. The classification of a resource r is defined as $cl(r)$. Tags are associated to resources in order to specify their classification. The trust level of a device d is defined as $cl(d)$ (see Section 2.3). A chain of certificates has to be resolved for each device in order to verify their trust level. When $cl(d) \geq cl(r)$, device d is enabled to deal with resource r. A symmetric key K_{cl} has to be defined for each classification cl:

$$\forall cl \in CL \quad : \text{Server generates symmetric key } K_{cl}$$

Each resource r has to be encrypted with the symmetric key K_{cl} corresponding to its classification $cl(r)$:

$$\forall r \in R_{AC} \quad : \text{Server computes encrypted resource } \widehat{r} = E_{K_{cl(r)}}(r)$$

The set of encrypted resources is $\widehat{R}_{AC} = \langle \widehat{r} \mid r \in R_{AC} \rangle$. Before sending it to the federation, the devices' trust chains are necessary for discovering trust levels in order to distribute keys:

$$\forall d \in F \quad \text{and} \quad \forall cl \in CL \quad : \text{if } cl(d) \geq cl : \text{Server computes } E_{PK_d}(K_{cl(d)})$$

The result is that each federated device can receive any encrypted resource but only keys for decrypting the parts it is authorized to deal with. For instance, a terminal that is trusted enough to deal with confidential data will receive $K_{confidential}$ and $K_{unclassified}$ but will not receive K_{secret}.

3.4 Management of Security Data

The natural choice for the format of both attribute certificates and the security policy markup that serves as the reference for data and key distribution has been XML [4]. The set of resources R is an XML document with different parts $r_1, \cdots, r_n$. Tags are used to define the classification of each part. A new document based on XML-encryption [13] can be generated for distribution.

As a matter of fact, XML is becoming the standard format for any data transaction on the Internet, due to its well known benefits: it is text based, human legible, self describing, structured, easy to treat, modular, and object oriented. Associating XML with one of the many existing libraries for parsing, displaying, transforming documents, and standards for signatures (xml-dsig) [14], encryption (xml-encrypt) [13], remote procedure call (SOAP), user data exchange (SAML) [11], access control (XACML) [15] deliver a powerful combination. Using XML as the format for secure attributes instead of S-expressions (SPKI) or ASN.1 (X.509) is not a novelty, and references can be found in an expired draft from Otani [9] and project Akenti [10]. However, the framework described in this paper deals with different problems and environments, which cannot be fulfilled with existing standards or applications.

XML Attribute Certificates. The proposed framework uses a proprietary format of XML attribute certificates, which allows to securely associate a capability to an employee or device. A public key is associated with each certificate, and only the user or device that is in possession of the corresponding private key can use the capability. Rights can be delegated if the certificate allows so and delegation can be performed in a local way without the need to connect to a centralized authority: each user behaves as a local authority for attribute certificates. Delegated credentials have a short lifetime, thus rendering the use of centralized revocation lists unnecessary, and permitting a local validation of the certificate chain. For long-lasting capabilities, revocation lists are envisaged. In this way, a trusted personal device can empower another device in a federation for acting in his place, for instance to retrieve a confidential document. Attribute certificates are used to store a different type of information: for an employee, it can consist of his role or personal rights, along with some useful personal data; for a device, information about its trust level, the company he belongs to, and its properties may be provided.

XML Documents. XML has also been chosen as the format for document storage. XML is actually becoming the natural format for many kinds of documents, be they in pure text, vector graphics, or more complex data formats; XML databases are spreading. Adding trust and security information to an XML document is quite straightforward, because XML has been conceived to be naturally extensible. In this work, there are two different kinds of XML documents. One is the *stored* document, which can be seen in the middle of Figure 1. The other one is the *transmitted* document, on the right part of Figure 1.

Upon a user request to the server for a resource stored in the backend database, an XML Parser generates a document depending on security policies, user credentials, and federated devices credentials. This document contains some cleartext parts, some encrypted parts, and an additional section to distribute symmetric keys for the encrypted sections. The receiving device will be able to show the encrypted parts if it can retrieve the corresponding key.

4 Implementation

The development of a full-fledged application using the security concepts presented in this paper is under way. A first prototype showing in what way secure federations are required in B2E has already been deployed and demonstrated.

4.1 Prototype Scenario

A salesman travels with his trusted personal device. He can use this one to access corporate emails but reading them on a small display is not always user-friendly. He uses surrounding public terminals, laptops, or video projectors to enlarge his display. He selects an email on his TPD and delegates it to a discovered device. Depending on trust levels, this device can be allowed to retrieve the email and to display it.

The prototype implements access control and trust based distribution for a specific service, displaying Web pages on surrounding devices. It is possible to extend this concept to other applications (e.g. signature or mobile code protection).

The corporate security policy defines which data can be accessed by the salesman, i.e. his emails (see Section 3.2) and the classification of data that can be handled by federated devices (see Section 3.3). For instance, some emails or attachments can be tagged as confidential.

- **Case 1**: User not authorized. Access control ensures that the salesman can only access the documents that he is authorized to.
- **Case 2**: User authorized but untrusted terminal. Using a public terminal, the emails that are confidential will not appear but will be replaced by a hypertext link. Upon selecting this link, the email is displayed by the salesman's personal device that is a trusted enough member of the federation.
- **Case 3**: User authorized and trusted terminal. When the salesman uses a terminal of a partner company, the confidential document is displayed.

Federations allow user-friendly interactions with surrounding artifacts and security is ensured by combining authorization (i.e. what a user is authorized to do) and trust levels (i.e. what kind of data can be securely handled by a federated device).

Next sections describe the platform used to implement a prototype of the system which permitted some early experiments.

4.2 Communication Technology

PAN Communications. Federations require local communications and discovery mechanisms to work with surrounding devices. Bluetooth is appropriate to implement those concepts and thus has been preferred to WLAN. Moreover, it is planed to run the application on smart-phones that do not offer WLAN interfaces. The access to Bluetooth from Java has been specified [5] but there is no available implementation. A Dynamic Link Library has been implemented to connect devices through the Bluetooth serial profile. Bluetooth connections can be piloted from applications thanks to Java Native Interface (JNI).

Web-based application. Figure 2 gives an overview of the architecture. A Web-mail interface was adopted that makes it easy to integrate XML transformations. On each machine, a local HTTP server acts as a proxy when accessing corporate intranet. This proxy is in charge of transforming XML into HTML, decrypting encrypted parts when the key is available or encapsulating them in applets when the key is unknown, and pushing data to the browser. It has been chosen to use a browser for the graphical user interface because it is standardized and available on all considered platforms. The browser loads some applets that communicate with the proxy so that pages can be pushed from browsers of other devices. Moreover, a toolbar allows to select discovered devices and to push them pages or URL (i.e. delegate access right).

XML has been chosen for exchanging data between proxies and the backend and for defining attribute certificates.

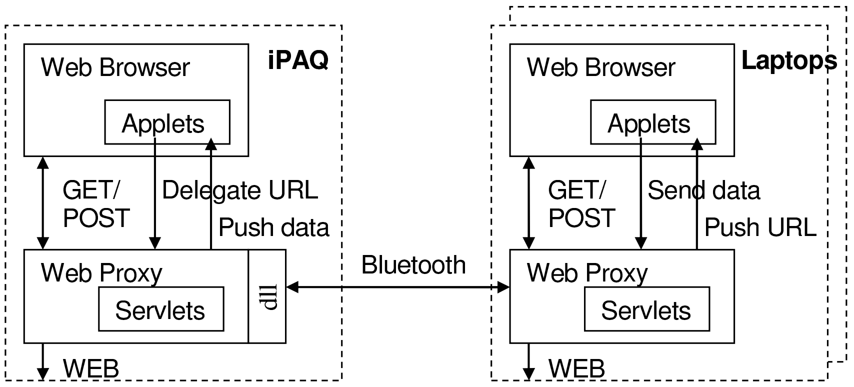


Fig. 2. Prototype architecture

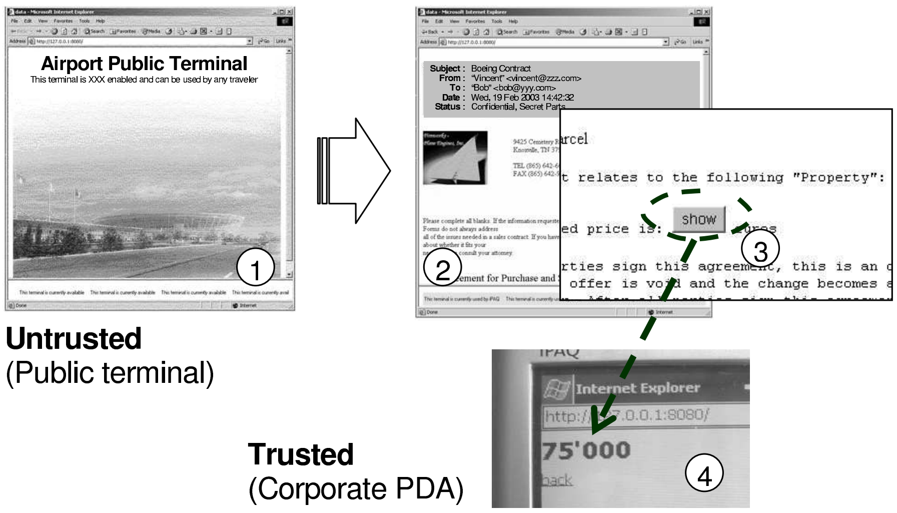
4.3 Platform

Hardware. Personal devices need short range wireless communication. Moreover, an important computational power can be required when signing certificates or parsing XML data. The iPAQ 3870 was chosen for implementing a personal device and laptops for federated devices (terminals). Tamper-resistant modules can be used to handle critical data such as private key: it has been chosen to use a SIM card to protect the user's private key. This module has been selected because of its ubiquity in mobile environments. The API to access a SIM card from Java is under way [7].

Software Environment. In order to reuse results of this work, it has been chosen to work on a Pocket PC operating system. This choice has a strong impact on the Java virtual machines supported and Bluetooth profiles available. Only Personal Java (Jeode) was offering Java Native Interfaces for the Pocket PC. JNI was a mandatory feature to be able to access Bluetooth from Java without JSR-82 implementation [5], which were non-existent when the development was started. Personal Java is a lightweight Java for PDAs that offers an extension of JDK1.1.8 with Java 2 security. Those restrictions limit the libraries available to parse and transform XML data. For application level security, the Bouncy Castle cryptographic library was chosen. It provides a Java Cryptographic Extension (JCE) that runs in JDK1.1.8 and offers all required cryptographic tools.

4.4 Prototype

Figure 3 shows some snapshots of the prototype. The Web browser, which runs on a public terminal in an airport, displays a welcome message (see Label 1). When the pervasive salesman is close enough, the terminal is discovered and appears in a toolbar of his PDA browser. He can select a link to an email and delegate it to



Subject : Boeing Contract
From : "Vincent" <vincent@zzz.com>
To : "Bob" <bob@yyy.com>
Date : Wed, 19 Feb 2003 14:42:32
Status : Confidential, Secret Parts

rcel

t relates to the following "Property":

ed price is: show euros

rties sign this agreement, this is an
offer is void and the change becomes

Fig. 3. example of trust-based distribution

the discovered terminal (see Label 2). When a part of the document cannot be decrypted by a terminal that is not trusted enough, this part is replaced by an applet displaying a button (see Label 3). Here, the negotiated price of a contract is tagged as confidential and cannot appear on a public terminal: security policies enforcement ensures that a public terminal is not able do decrypt confidential data. The user can press the button to find a member of the federation that is trusted enough. For instance, the confidential part of the document will appear on his PDA (see Label 4). When the same operation is done on a corporate terminal, the whole email (including confidential data) appears.

This approach offers a simple mechanism to adapt a content to the trust level. User-friendliness is not sacrificed for security: the salesman does not have to care when delegating the presentation of information to another device because the mandatory security model of his corporation ensures that classified data will never be delivered to an entity untrusted by his company.

5 Conclusion and Future Work

Solutions to the problem of trust assignment for devices are hard to develop in an open trust model for pervasive application scenarios. In contrast, the scheme proposed in this paper addresses the business-to-employee and business-to-business domains that make it possible to exploit *a priori* defined trust knowledge. Such knowledge covers the state of the relationships between a company, its employees, and devices on one hand, and the company's business partner correspondents on the other hand. In such a setting, reasonable assumptions about the trustworthiness of devices can be made. This permits the specification and enforcement of the corporate security policy required for letting an employee take advantage of any device available in his surroundings, be it his companies' or another one's, and yet protect corporate assets. This paper contributes with protocols supporting distributed access control, trust level verification, data distribution, and authorization within a set of devices federated to provide higher-level services for a company's mobile work-force.

A prototype implementation for parts of this architecture was developed based on personal digital assistants acting as and employees' trusted personal device. This prototype is currently being integrated into a larger framework for securing mobile business applications. The use of GSM SIM cards equipped with dedicated software as tamper-resistant security modules embedded into a PDA is currently under work. Such a trusted device will be able to perform all security-critical operations such as issuing delegation certificates and transferring access rights of particular resources to federated devices. The XML markup is also under development at the time of this writing.

We further plan to investigate how the security architecture presented in this paper can be adapted to scenarios where even less *a priori* information is available. More specifically, we are interested in how to assess the trust level of devices in a federation when connection to corporate back-end services is not permanently available.

Acknowledgement

This research was supported by European Union Project "*WiTness – Wireless Trust for Mobile Business*" (FP5 IST-2001-32275), by SAP, and by Institut Eurécom. MERCI A LAURENT GOMEZ !!!!!!!!!!!!

References

1. J. Bohn, V. Coroama, M. Langheinrich, F. Mattern, M. Rohs. *Disappearing Computers Everywhere - Living in a World of Smart Everyday Objects.* Proc. of New Media, Technology and Everyday Life in Europe Conference, London, UK, April 2003
2. C. Collberg, C. Thomborson, and D. Low. *A taxonomy of obfuscating transformations*, Technical Report 148, Department of Computer Science, University of Auckland, 1996.
3. M.J. Covington, M.J. Moyer, and M. Ahamad. *Generalized Role-Based Access Control for Securing Future Applications.* In 23rd National Information Systems Security Conference, 2000.
4. *Extensible Markup Language (XML) 1.0 (Second Edition).* W3C Recommendation, 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006
5. JSR 82: Java APIs for Bluetooth, http://www.jcp.org/en/jsr/detail?id=82
6. Network Working Group, Request for Comments 2693: *SPKI Certificate Theory*, September 1999.
7. JSR 177 : Security and Trust Services API for J2ME, http://www.jcp.org/en/jsr/detail?id=177

8. T. Sander and C. Tschudin. *On software protection via function hiding.* In Proceedings of the Second Workshop on Information Hiding, Portland, Oregon, USA, April 1998.
9. *Capability Card: An Attribute Certificate in XML*, Expired Internet Draft draft-otani-ccard-00.txt, 18 Nov. 1998
10. Akenti: a security model and architecture to provide scalable security services in highly distributed network environments http://www-itg.lbl.gov/Akenti/
11. *Security Assertion Markup Language (SAML 1.0).* OASIS standard, 5-Nov-2002, http://www.oasis-open.org/committees/security/
12. The Trusted Computing Platform Alliance. *Building A Foundation of Trust in the PC.* White paper, January 2000. http://www.trustedcomputing.org/
13. *XML Encryption.* W3C Recommendation, 10 December 2002, http://www.w3.org/Encryption/
14. *XML Digital Signature.* W3C Recommendation, 12 February 2002, http://www.w3.org/Signature/
15. *eXtensible Access Control Markup (XACML 1.0).* OASIS Standard , 6 Feb. 2003, www.oasis-open.org/committees/xacml/