One-Time Capabilities for Authorizations without Trust

Laurent Bussard and Refik Molva
Institut Eurécom[1], Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis (France)
{bussard, molva}@eurecom.fr

Abstract

This paper introduces and solves a security problem of pervasive computing: how to define authorizations for off-line interactions when trust relationships among entities do not exist. A new type of capability is proposed to assure that user interactions are not traceable and that misbehaving users automatically lose some money they deposited as a guarantee of their loyalty. The solution inspired by electronic cash is based on a new version of the cut-and-choose protocol adapted for open environments.

1. Introduction

The pervasive computing paradigm is exploding in popularity as one of the main applications taking advantage of advanced mobile and wireless communication technologies. Large scale deployment of pervasive computer applications still heavily depends on the assurance of essential security properties for users and service providers. Apart from security exposures due to the underlying mobile and wireless communications, pervasive computing inherently brings up some issues that are relevant to security:

1. Lack of infrastructure or off-line operation with respect to the infrastructure.
2. Lack of organization or self-organization, hence lack of a priori trust among parties.

In this paper we tackle both problems through the design of an access control scheme suited to pervasive computing. We focus on limiting the access to low-value services (printers, coffee machines, etc.) that are freely offered to users visiting some environment. In line with requirement 2, a deposit is used to guarantee the correct behavior of untrusted users. Money is envisioned as a universal enforcement mechanism when there is no trust relationship. Protocols ensure that service providers can only cash the deposit in case of misbehavior and that users are not traceable. As stated in requirement 1, the verification of users' rights can be performed without any communication with a third party system.

Section 2 gives a precise description of the scenario with respect to the problem and analyzes the limitations of existing access control solutions in the light of this scenario. The solution is described in Section 3 in terms of a protocol in three phases. The security of this scheme is discussed in Section 4. More details on this work are available in a research report [2].

2. Pervasive Computing Scenario

The scenario envisioned in this paper consists of several pervasive computing environments (PCE) as shown in Figure 1. Each PCE includes a set of appliance servers ($S_1, S_2 \cdots, S_z$) and an Access Control Authority (ACA). A dynamic user population called visitors ($V_1, V_2, \cdots, V_v$) randomly visits PCEs and requests services from appliance servers. The ACA of each PCE is in charge of providing visitors rights to access servers. In order to suit the possibly large coverage of each PCE and the limited transmission capabilities of pervasive servers, the access control solution does not require on-line connectivity between the servers and the ACA. However, servers periodically need a way to exchange some data with the ACA. For instance, once a day, coffee machines provide data to the ACA via the support personnel. Since servers cannot communicate with the ACA in a timely and interactive way, we qualify the interactions as *off-line*. Each user can establish interactive exchanges with the server he/she is in touch with. Another characteristic of this environment is the lack of a priori trust between servers and visitors. Servers do not trust visitors and possibly are not even able to identify them.

The access control in such pervasive computing scenario can be illustrated by an example as follows.

1 Institut Eurécom's research is partially supported by its members: Bouygues Télécom, Cegetel, France Télécom, Hasler Foundation, Hitachi, STMicroelectronics, Swisscom, Texas Instruments, and Thales

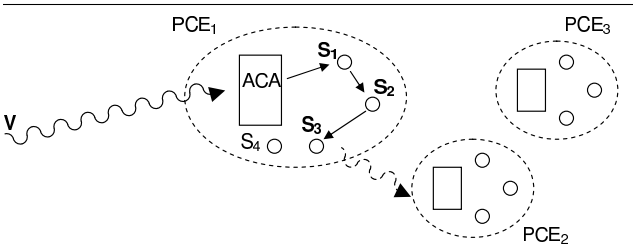

Figure 1. Pervasive computing scenario

2.1. Example

A visitor arrives at the gate of a shopping mall (PCE) that provides some free services (e.g. printers, coffee machines, and network access points). Once the visitor is through the registration he/she passes near a wireless sensor acting on behalf of the ACA. The sensor loads the visitor's personal device with a set of rights based on the visitor's attributes (e.g. location, profile). Using the rights he/she thus obtained, the visitor can access various services. The main security goal in this context is to prevent the visitor from unauthorized access to services even when the services are provided by off-line devices.

2.2. Existing Approaches

A large part of common approaches to control access rely on identity (e.g. access control lists, identity certificates) but identity is often meaningless in pervasive computing. The simple public key infrastructure (SPKI/SDSI) [7] allows for the verification of a party's credentials without explicit identification. However, this approach also requires some infrastructure to certify credentials.

When privacy is a major concern, [1] presents digital credentials with selective disclosure and Idemix [5] offers an interesting approach to create non-transferable anonymous multi-show credentials. Both approaches need some form of infrastructure for authenticating users receiving rights.

Electronic cash [4, 6] can be viewed as a one-time capability when the amount is replaced by some authorization. As an anonymous scheme suitable for off-line transactions, electronic cash nicely meets the requirements of pervasive computing. The main deterrent to double spending in this scheme is based on the disclosure of cheaters' identity by the banking organization. In pervasive computing environments, the disclosure of the identity is not an efficient deterrent against double use of credentials because there is no formal relationship between the service provider and the users.

3. Solution: One-time capabilities

Our solution consists of a capability scheme based on an enhancement of the electronic cash with a new penalty mechanism. The resulting one-time capability (OTC) scheme offers the following features:

- *One-time*: each capability or access right can be used at most once. It allows a fine control of users right without relying on time or revocation.
- *Untraceable*: the scheme ensures that users cannot be traced by the environment when behaving fairly.
- *Off-line*: access to services is done locally without requiring to establish a connection with some trusted third party.
- *Penalty*: in order to deter misbehaving visitors from attempting to use an OTC more than once, a deposit ensures that any cheater will lose money. The service provider is assured that he will receive money in case of misbehavior and visitors are assured that the deposit cannot be cashed when they behave fairly.

For the sake of simplicity, the OTC mechanism presented in this short paper does not properly address privacy. The basic solution can however easily be enhanced with untraceability based on Chaum's blind signature [3] technique at the expense of increased computational complexity. The solution consists of three phases: *capability creation* that includes the deposit mechanism, *access* to resources, and *detection of double use* to penalize cheaters.

3.1. Penalty without Hierarchical Relationships

In the OTC mechanism, a new type of electronic check serves as a deterrent against double use of a one-time capability. Unlike classical electronic checks [8], the new type of e-checks included in the OTC must allow for the verification of the issuer's signature without revealing this signature that enables one to cash them. A new signature scheme thus has to be used when V signs the e-check, when ACA verifies the signature, and when ACA cashes the e-check. This mechanism is a substitute for the signature of on-line as well as off-line e-checks. For the sake of simplicity this paper only presents a basic scheme where e-checks are on-line payment orders: V orders his bank to transfer some amount from his account to ACA's account.

During the creation of a one-time capability, V can prove to ACA that a secret $K \mid \mathcal{H}(K) \in HK$ where $HK = \{\mathcal{H}(K_1), \mathcal{H}(K_2), \cdots, \mathcal{H}(K_n)\}$ will be revealed in case of double use. A filled e-check is defined as $fc = \mathrm{SIGN}_{\mathrm{bank}}(V, ACA, \mathrm{amount}, \mathrm{nb})$ where nb is the check number and where ACA and V could be replaced by the account identification of ACA and V, respectively. This filled e-check is created during an

on-line interaction with the bank. V provides as deposit a signed e-check $sc = \mathrm{SIGN}_V(fc, HK)$. This deposit can only be cashed by ACA when one of the secret $K \mid \mathcal{H}(K) \in HK$ is known. Hence a valid e-check is the combination of a deposit and one of the corresponding secrets: $vc = \{sc, K\}$.

3.2. Phase 1: Capability Creation

During this phase the ACA provides a set of OTC to a visitor entering the PCE. The main purpose of the protocol in this phase is twofold: to prove the ACA that it will be able to cash the e-check if the visitor misbehaves (uses the OTC more than once) and to assure that the ACA cannot cash the e-check if the visitor properly behaves. These purposes are fulfilled by a new mechanism that allows the ACA to verify that the contents of the e-check are valid and that the capability includes a valid signature of the e-check that can be revealed only in case of misbehavior by the visitor.

Authentication is optional in this protocol. Any entity able to provide an e-check can receive some rights. However, some authorizations can be restricted to visitors with specific attributes (e.g. employee of a partner company). The following protocols do not address this point.

Let V define a set $N = \{1, 2, \cdots, n\}$ that will be used in two cut-and-choose protocols [4] during the creation of one-time capabilities and during the access to resources. The set size n is a security parameter (see Section 4).

1.1) V generates $K_i \in (0,1)^l \quad \forall i \in N$
V computes $HK_N = \{\mathcal{H}(K_1), \cdots, \mathcal{H}(K_n)\}$
$K_1, \cdots, K_n$ are kept secret by V and the set HK_N will be disclosed by V during further steps of the protocol.

1.2) V generates $m_N = \mathcal{H}(m_1 \| m_2 \| \cdots \| m_n)$
where $m_i = \mathcal{H}(a_i \| b_i)$
where $a_i = \mathcal{H}((c_i \oplus data_i) \| d_i)$
where $b_i = \mathcal{H}(c_i \| e_i)$
where $data_i = \{K_i, \mathrm{nb}\} \quad \forall i \in N$
This construction is necessary for preventing double use of capabilities. It assures that $data_i$ can only be revealed when c_i and $(c_i \oplus data_i)$ are known, i.e. when the capability has been used twice (see Step 3.2). c_i, d_i, and e_i are random numbers: $d_i, e_i \in (0,1)^l$ and $c_i \in (0,1)^{2 \cdot l}$. In electronic cash, a similar mechanism is used to reveal the identity of cheaters. Here, because of the lack of organization, the identity is useless and a valid e-check is directly available when V cheats: $data_i$ contains a secret $K_i \mid i \in N$ and a reference to a deposit (check number: nb). A valid e-check is obtained by combining this secret and this deposit.

1.3) $V \rightarrow ACA \quad HK_N, m_N, fc$
V sends ACA the data required to create the one-time capability. Before releasing the one-time capability, ACA verifies the e-check f_c.

1.4) $ACA \rightarrow V \quad R \subset N \quad$ where $|R| = \frac{|N|}{2}$
The ACA chooses randomly a subset R (half of the set N) for verification purposes and requests V to send details on how each $m_i \mid i \in R$ is constructed.

1.5) $V \rightarrow ACA \quad c_i, d_i, e_i, K_i \quad \forall i \in R$
$m_j \quad \forall j \in \bar{R} \ (R \cup \bar{R} = N)$
V discloses the details to construct each $m_i \mid i \in R$ for verification purposes.

1.6) ACA computes m_i from $c_i, d_i, e_i, K_i, \mathrm{nb} \quad \forall i \in R$
ACA verifies $\mathcal{H}(K_i) \stackrel{?}{\in} HK_N \quad \forall i \in R$
ACA verifies $m_N \stackrel{?}{=} \mathcal{H}(m_1 \| m_2 \| \cdots \| m_n)$
When all $m_i \mid i \in R$ are well-constructed, there is a high probability (see Section 4 for a discussion on the security evaluation) that other $m_i \mid i \in \bar{R}$ contain a secret $K_i | i \in \bar{R}$ usable to generate a valid e-check from the deposit.

1.7) ACA computes $m_{\bar{R}} = \mathcal{H}(m_{\bar{r}_1} \| \cdots \| m_{\bar{r}_{n/2}})$ and
$HK_{\bar{R}} = \{\mathcal{H}(K_{\bar{r}_1}), \cdots, \mathcal{H}(K_{\bar{r}_{n/2}})\}$
where $\bar{r}_1, \cdots, \bar{r}_{n/2} \in \bar{R}$
Both V and ACA only keep unrevealed secrets ($\bar{R} \subset N$) to build the one-time capability and the deposit. The ACA requests the deposit.

1.8) $V \rightarrow ACA \quad sc = \mathrm{SIGN}_V(fc, HK_{\bar{R}})$
V signs the set of unrevealed secrets resulting in the deposit that has to be combined with one of the secrets $K_i \mid i \in \bar{R}$ to become a valid e-check.

1.9) $ACA \rightarrow V \quad capability =$
$SIGN_{ACA}(m_{\bar{R}}, rights, \mathrm{validity})$
The ACA stores the deposit and provides the one-time capability to V. As with the simple public key infrastructure, rights are application specific and their representation is out of the scope of this paper.

3.3. Phase 2: Service Access with Capability

In the second phase of the OTC protocol, the visitor V uses the OTCs to access resources. The resource access takes place off-line, that is, the server S cannot rely on the ACA to verify the OTC. When the visitor proves that it knows the secret corresponding to the capability, part of the information to retrieve the signature is provided to the server. This information is not sufficient to get a valid signature of the e-check but prevents double use of the OTC.

2.1) $V \rightarrow S \quad request, capability$
V requests to access a resource and provides the one-time capability proving that the request is authorized.

2.2) $S \rightarrow V \quad T \subset \bar{R} \quad$ where $|T| = \frac{|\bar{R}|}{2}$

S starts a challenge-response based on a second cut-and-choose protocol: it randomly chooses a subset T (half of the set $\bar{R}$) and sends it to V.

2.3) $V \rightarrow S$ $\quad (c_i \oplus data_i), d_i, b_i \quad \forall i \in T$
$\quad c_j, e_j, a_j \quad \forall j \in \bar{T} \ (T \cup \bar{T} = \bar{R})$

V reveals half the information for the set $\bar{R}$ to prove that it can construct $m_{\bar{R}}$. However, S has no way to get any $data_i \mid i \in \bar{R}$ and thus cannot collude with ACA to cash the e-check. This step achieves two goals: it allows S to verify that the visitor knows the secret corresponding to the capability and causes V to reveal some information that is not sufficient to get a valid e-check but that would forbid double use.

2.4) S computes $\quad m_i$ from $(c_i \oplus data_i), d_i, b_i \quad \forall i \in T$
S computes $\quad m_j$ from $c_j, e_j, a_j \quad \forall j \in \bar{T}$
S verifies $\quad m_{\bar{R}} \stackrel{?}{=} \mathcal{H}(m_{\bar{r}_1} \| \cdots \| m_{\bar{r}_{n/2}})$
where $\bar{r}_1, \cdots, \bar{r}_{n/2} \in \bar{R}$

This step allows S to verify that the visitor knows the secret corresponding to the capability before authorizing resource access.

3.4. Phase 3: Detection of Double Use

This phase is necessary to identify and punish visitors that have used a OTC more than once. Penalty is postponed as long as the servers have not sent data to the ACA. Visitor access logs will be provided in batch by servers to the ACA. When the use of the same OTC appears in more than one server's log, the ACA will be able to retrieve the signature of the e-check embedded in the OTC and to cash the e-check.

3.1) $S_1 \rightarrow ACA \quad c_i \oplus data_i \quad \forall i \in T_{S_1}$
$\quad c_i \quad \forall i \in \bar{T}_{S_1} \ (T_{S_1} \cup \bar{T}_{S_1} = \bar{R})$

Periodically, when the server S_1 is on-line, it sends relevant data to the ACA. The set T_{S_1} has been randomly chosen by S_1 and is different for each server and for each capability. As long as V does not cheat, those data are useless.

3.2) $S_z \rightarrow ACA \quad c_i \oplus data_i \quad \forall i \in T_{S_z}$
$\quad c_i \quad \forall i \in \bar{T}_{S_z} \ (T_{S_z} \cup \bar{T}_{S_z} = \bar{R})$

If the same one-time capability has been used with servers S_1 and S_z, there is a high probability that $\exists i$ such that $i \in T_{S_1}$ and $i \notin T_{S_z}$. Thus c_i and $(c_i \oplus data_i)$ are known and ACA can retrieve $data_i$ and the secret K_i. This secret combined with the deposit is a valid e-check: $vc = (sc, K_i)$ where $\mathcal{H}(K_i) \in HK_{\bar{R}}$. This e-check can be cashed.

4. Security Evaluation

Protocols described in Section 3 define a solution to avoid double use of one-time capabilities in off-line context. When a capability is used more than once, there is a high probability that the embedded electronic check will be revealed. Two security parameters have to be evaluated:

- n is the number of steps in the first cut-and-choose protocol.
- l is the size of the one-way function outputs.

Probability of undetectable double-use: in the main attack against this scheme, a visitor can try obtaining a valid capability that does not embed a valid signature on the deposit. The capability creation protocol ensures that the secrets of half the set are verified. Thus, when an attacker tries to generate a capability that will never reveal a valid e-check, he has to provide $n/2$ invalid secrets. The probability that the ACA does not verify one of those invalid data is:

$$\mathbf{p_{2use}} = \underbrace{\frac{\frac{n}{2}}{n}}_{\text{valid } m_1} \cdot \underbrace{\frac{\frac{n}{2}-1}{n-1}}_{\text{valid } m_2} \cdot \cdots \cdot \underbrace{\frac{1}{\frac{n}{2}+1}}_{\text{valid } m_{n/2}} = \frac{(\frac{n}{2}!)^2}{n!}$$

It is possible to set the probability of the successful attack at an arbitrary small value by choosing the right value for n. For instance, if $n = 100$, $p_{attack} \cong 2^{-96}$.

Impact of multiple use: since each server keeps track of capabilities it already received, double use attempts performed with the same server will be detected by the server itself. Double use of an OTC with different servers is postponed but will be detected by the ACA and enable penalties. However, when the same OTC is used more than twice with different servers, only one e-check can be cashed as part of the penalty mechanism. The degree of the penalty (the amount of the e-check) should thus be set sufficiently high in order to eliminate possible advantages of multiple uses beyond the double use.

Probability of penalty disclosure: it is important to protect the visitor against a malicious service provider trying to get a valid signature for an existing deposit in order to retrieve a valid e-check. The attacker has to find a valid K_i corresponding to an embedded $h(K_i)$ where $i \in \bar{R}$. The birthday attack is not relevant in this case and the probability of a successful brute force attack against the hash function is:

$$p_{disclose} = \underbrace{\frac{n}{2}}_{|\bar{R}|} \cdot \underbrace{2^{-l}}_{hash}$$

Protection of the visitor: it is important to assure that a visitor's capabilities and corresponding secrets cannot be disclosed by intruders since based on this information an intruder could get a valid e-check. It is also necessary to prevent any operation that could cause unintended double use by the visitor. A tamper-resistant module such as a smart card could be used to protect a visitor's secrets.

5. Conclusion and Future Work

This paper proposes a solution for access control in pervasive computing environments. We suggest one-time capabilities whose validation does not require any on-line communication with a security infrastructure. Each capability is one-time in that it can be used only once. The one-time property of the capability is assured by a strong deterrent: if a user misbehaves by showing more than once a one-time capability, he/she will undeniably incur a penalty. Due to the lack of organizational pressure mechanisms in the pervasive environment, the solution has recourse to money as a universal penalty mechanism. When a user gets a one-time capability, he/she has to prove that an electronic check will be available for payment in case of misbehavior, i.e. double use of the capability. Conversely, the user has the guarantee that the electronic check cannot be cashed if he/she correctly behaves. These properties are assured based on a new scheme that allows electronic checks embedded in capabilities to be verified without revealing their signature.

Privacy of users is a major concern in such open environments. The scheme presented in this paper is based on two cut-and-choose protocols (phases 1.4 and 2.2) and thus capabilities can straightforwardly become untraceable by using Chaum's e-cash protocol [4]. Design of a solution based on more efficient approaches that do not rely on cut-and-choose (e.g. [6]) is still an open issue.

We are investigating how this proposal can be adapted to thwart denial of service (DoS) attacks that are specific to pervasive computing. For instance, an airport lounge could offer some printing facility to any visitor but cannot afford DoS attacks during which some visitor prints anonymously hundreds of pages on all surrounding printers. Our scheme fits very well the protection of such free services that are potential targets of DoS attacks. Indeed, such attacks generally occur when the access to the service costs nothing and when attackers are not traceable. The solution presented in this paper enables anonymous access to free services but assures direct penalty in case of misbehavior. An airport lounge can provide capabilities that allow visitors to print up to ten pages. Visitors, which know that mounting a DoS attack will reveal their deposit (e.g. fifty dollars), would behave fairly. It is also necessary to forbid a rogue server from mounting a DoS attack against visitors. In this case, some authentication of the server can be necessary: visitors will only show OTC to servers certified by the entity that delivered this capability.

In this paper, we assume that a visitor cannot get hundreds of one-time capabilities to mount a denial of service attack without revealing an e-check. The fact that a visitor only get few credentials is enforced by authentication during retrieval of capabilities (first phase) and/or by a face to face interaction between the visitor and some person that welcomes him and delivers capabilities. We are working on automatic mechanisms to forbid visitors from obtaining numerous capabilities.

References

[1] Stefan Brands. *A technical Overview of Digital Credentials*. Research Report, February 2002.

[2] L. Bussard and R. Molva. *One-time Authorization for Offline Interactions*, Technical Report, Eurecom Institute, RR-03-077, 2003.

[3] D. Chaum, R.L. Rivest, *Blind Signatures for Untraceable Payments*, Advances in Cryptology, Proceedings of Crypto 82, pp. 199–203, 1982.

[4] D. Chaum, A. Fiat, M. Naor, *Untraceable Electronic Cash*, Proceedings of Crypto'88, LNCS 403, Springer Verlag, pp. 319–327, 1988.

[5] J. Camenisch and E. Van Herreweghen. *Design and Implementation of the idemix Anonymous Credential System* In ACM CCS 2002.

[6] N. Ferguson. *Single term off-line coins*, In Advances in Cryptology – EUROCRYPT 93, volume 765 of LNCS, pages 318–328. Springer-Verlag, 1994.

[7] Network Working Group, Request for Comments 2693: *SPKI Certificate Theory*, September 1999.

[8] M.H.Sherif, *Protocols for Secure Electronic Commerce*. Advanced and Emerging Communications Technologies Series, CRC Press, March 2000.