Institut EURECOM
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis
(France)

Research Report[a] N^o 77 — RR-03-077

One-Time Authorization for off-line Interactions

Laurent Bussard and Refik Molva

May 15, 2003

[a] Eurecoms research is partially supported by its industrial partners: Bouygues Telecom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Telecom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, and Thales

One-Time Authorization for off-line Interactions

Laurent Bussard and Refik Molva

Institut Eurecom
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis
(France)
{bussard,molva}@eurecom.fr

Abstract. Ubiquitous application environments are characterized by lack of on-line access to communication facilities and lack of a priori trust among parties. In this paper we present an access control scheme suited to these environments that allows a user to get authorized access to a service based on the one-time credential concept. In this scheme, the user and the service provider do not need to be part of the same organization or to trust one another. The verification of the user's credential can be performed without any communication with a third party system, since the validity of each one-time credential can be locally checked by each service provider. The one-time property of credentials further prevents double use of an access right by the user subsequently attempting to access several service providers. The one-time property and the resulting double use prevention rely on a penalty mechanism whereby a cheating user looses some money he/she deposited as a guarantee of his/her loyalty prior to a serie of service accesses. The one-time property does not require a common trust structure encompassing clients and servers in that it only has recourse to a universal enforcement mechanism based on money.

1 Introduction

The ubiquitous 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, ubiquitous computing applications inherently bring 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.

New solutions addressing these issues are required both for the protection of users, including privacy measures, and for controlling the access to valuable resources like commercial services. In this paper we tackle the latter of these problems through the design of an access control scheme suited to ubiquitous computing. The suggested access control scheme allows a user to get authorized access to a service based on the one-time credential concept. As stated in requirement 2, using this access control scheme, the user and the service provider do not need to be part of the same organization or to trust one another. In line with requirement 1, the verification of the user's credential can be performed without any communication with a third party system, since the validity of each one-time credential can be locally checked by each service provider. The one-time property of credentials further prevents double use of an access right by the user subsequently attempting to access several service providers. The one-time property and the resulting double use prevention rely on a penalty mechanism whereby a cheating user looses some money he/she deposited as a guarantee of his/her loyalty prior to a series of service accesses. The one-time property also meets requirement 2 in that it only has recourse to a universal enforcement mechanism based on money.

We first give a precise description of the application scenario with respect to the application problem and analyze the limitations of existing access control solutions in the light of this scenario. Our solution based on the one-time capability concept is then introduced first with a high level sketch of the idea. A detailed description is given in terms of a protocol in three phases. The security of the protocol, its complexity and possible extensions for privacy are discussed in the last sections of the paper.

2 Ubiquitous Application Scenario

The application 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 each PCE and requests services from appliance servers. The ACA of each PCE is in charge of providing visitors rights to access servers. Due to the possibly large coverage of each PCE and the limited transmission capabilities of pervasive servers, no on-line connectivity between the servers and the ACA is required. However, servers periodically exchange some data with the ACA. For instance, once a day, vending 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 between the ACA and the servers of a PCE as *off-line*. Each user on the other hand 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. Within a PCE, servers trust the ACA with respect to the authorization scheme in that each access request bearing a valid authorization proof delivered by the ACA is granted access by the server to which it is destined. The policy according to which the ACA grants access rights is out of the scope of this paper. Moreover the ACA is only in charge of enforcing access control within a PCE and the fact that there is a single ACA in each PCE does not necessarily imply that all servers of the PCE belong to the same administrative domain. Furthermore, multiple servers might offer the same type of service like printing, vending food and beverages, or opening doors.

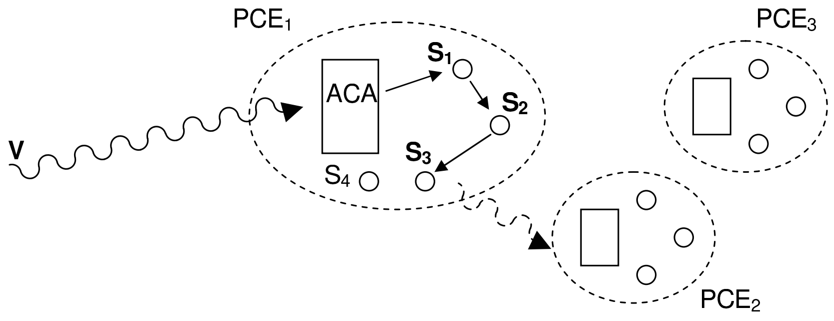

Fig. 1. Ubiquitous Application Scenario

The access control in the ubiquitous application scenario can be illustrated by an example as follows.

2.1 Example

A visitor arrives at the gate of a shopping mall (PCE). Once he/she 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 (e.g. cell-phone, PDA, or smart card) with a set of rights based on the visitor's attributes (e.g. role, identity, or location) and the services the visitor has subscribed to during the registration. Using the rights he/she thus obtained, the visitor can access various services like vending machines. The ACA is operated by the shopping mall but the services to which access is granted by means of the authorization scheme can be managed by independent service providers. 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. For instance, if the visitor is only authorized to get one coffee and tries to use his right with two different coffee machines that cannot communicate with one another or with the ACA, the access control mechanism should detect and prevent the duplicate access attempt.

2.2 Security Requirements

The ubiquitous application scenario described in Section 2 requires access control in order to properly validate each visitor request based on the corresponding server's security policy. Apart from the support of classical access control functions, the access control mechanism should address some specific requirements raised by the ubiquitous application scenario:

– *Security Requirement 1:* lack of a priori relationship between visitors and servers. Since there is no organization governing servers and visitors, access control decisions cannot refer to a security policy governing both servers and visitors; since there is no naming space encompassing both servers and visitors either, servers and visitors cannot reliably identify one another.
– *Security Requirement 2:* off-line servers. Due to the limited communication capabilities of the servers and the lack of a communication infrastructure, access control mechanisms should not rely on timely interaction between servers and the ACA and among the servers whereas one-way batch data transfers from the servers to the ACA can take place with a low periodicity. The lack of timely interaction between servers directly or through the ACA prevents servers from being able to detect multiple use of a one-time right with several servers.
– *Security Requirement 3:* privacy. Taking into account the possibly large coverage of PCEs, private information concerning the identity, attributes and behavior of each user is exposed to widespread dissemination. Hiding part or all of this information during visitor-server interaction is an essential requirement for the acceptance of ubiquitous applications by users.

2.3 Existing Approaches

A straightforward solution to access control consists of access control lists (ACL) that would be supported by the servers of the ubiquitous application scenario. Whereas ACLs satisfy Requirement 2 by allowing each server to be able to locally take the access control decision pertaining to its resources, because of requirement 1, the ubiquitous application scenario does not allow for the authentication of visitors by servers prior to the access control decision. Not only visitors and servers do not belong to the same security domain but there is not even a common naming convention for all visitors and servers on which to build an authentication mechanism. The simple public key infrastructure (SPKI/SDSI) [8] proposes authorization certificates to deal with unknown entities. The drawback of this approach is the complexity of revocation: the only revocation solution during off-line interactions is based on short-term certificates that would require the visitors to frequently communicate with the ACA to get new certificates and that would not allow detection of duplicate access with several off-line servers. Addressing the privacy requirement [1] presents digital credentials with selective disclosure. Idemix [4, 5] offers an interesting approach to create non-transferable anonymous multi-show credentials. Unlinkability is ensured: each user has different pseudonyms that cannot be correlated across various access attempts. Capabilities cannot be transferred because the transfer thereof would require sharing a pseudonym with another entity. Even though perfectly suitable for privacy concerns like unlinkability and non-transferability, Idemix would not meet off-line revocation requirements of the ubiquitous application scenario. Tamper-proof hardware (TPH) like the Trusted Computing Platform Alliance or a downgraded version thereof such as a smartcard could be viewed as a viable alternative for building a trusted reference monitor in each server but the access control mechanism implemented by the TPH would still suffer from the limitations of requirements 1 and 2. Chaum's electronic cash [3] offers a one-time credential that suits the off-line nature of the servers in the ubiquitous application scenario. In this scheme, unlinkability is assured by blind signatures [2]. When an electronic coin is used twice, it is possible to retrieve the identity of the cheater and the bank that issued the coin can act against this cheater. An access control scheme suitable for the ubiquitous application scenario could be envisioned based on an extension of electronic cash whereby the amount in the electronic cash is replaced by the encoding of some rights. A strong requirement is the existence of a shared banking organization that issues electronic coins to users and performs compensation for merchants. The main deterrent to double spending in this scheme is thus based on the disclosure of cheaters' identity by the shared banking organization. The ubiquitous application scenario does not allow for a shared organization within a PCE or across several PCEs.

3 Solution: One-time capabilities for off-line interactions

In order to come up with an access control solution that meets the requirements of the ubiquitous application scenario, we introduce the concept of one-time capabilities (OTC). The OTC issued by the ACA represents the right to perform a single access to a resource. A OTC can be verified by a server off-line, that is, without any interaction with another server or with the ACA. The validation of the access right encoded in the OTC does not require the authentication of the visitor that issued the request including the OTC; the visitor only needs to prove that it is the party to whom the OTC was granted. The ultimate issue in this context is the assurance of the one-time property with off-line servers. Our solution to this problem is based on the postponed punishment principle, inspired by electronic cash, that if a visitor uses an OTC more than once then the violation of the one-time property will necessarily be detected later and cause the punishment of the cheating visitor with a penalty mechanism. Unlike electronic cash whereby the penalty consists of the disclosure of the cheater's identity and compensation of double spending by a banking organization that is trusted both by the payers and the payee, the OTC penalty mechanism does not require a unique banking organization or access control authority for all visitors and servers. The OTC penalty mechanism is based on a universal payment order or an electronic check (e-check). The payment order or the e-check do not need to be issued by a unique banking organization, any order issued by a financial organization recognized by the ACA is suitable for the purpose of the OTC mechanism. Since visitors mutually distrust both the ACA and the servers, the payment order (called the *e-check* for the sake of simplicity) embedded in an OTC has to be protected against possible misbehavior as follows:

- The ACA or the server should not be able to cash the e-check if the OTC is properly used (only once) by the visitor.
- The ACA should be able to verify that a valid e-check is embedded in the OTC.

Even though privacy is stated as an important requirement for ubiquitous applications, the OTC mechanism presented in this paper does not properly address privacy. The OTC mechanism offers a simple version of anonymity in that the access control mechanism does not require servers to know the identity of the visitors. Advanced privacy objectives like unlinkability are not pursued by the basic OTC approach (the ACA can link capabilities to visitors). The basic solution can however easily be enhanced with unlinkability based on Chaum's blind signature technique at the expense of increased computational complexity.

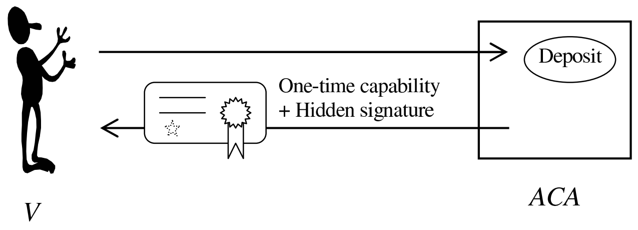

Fig. 2. Capability withdrawal

The solution consists of three phases: First, during the *capability withdrawal* phase (Figure 2) the ACA provides a set of OTC to a visitor entering the PCE. Apart from the classical access control operations through which access rights will be granted in terms of capabilities, 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 within the PCE) 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 e-check includes a valid signature that can be revealed only in case of misbehavior by the visitor. In the next phase the visitor uses the OTCs to *access* resources kept by various servers (Figure 3). The resource access takes place off-line, that is, the server 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 but prevents double use of the OTC.

Finally, *detection of double use* is necessary to identify and punish visitors that use an OTC more than once (Figure 4). This phase is postponed as long as the servers are not on-line. With off-line servers, visitor access logs will be provided in batch by servers to the ACA (for instance through a daily data

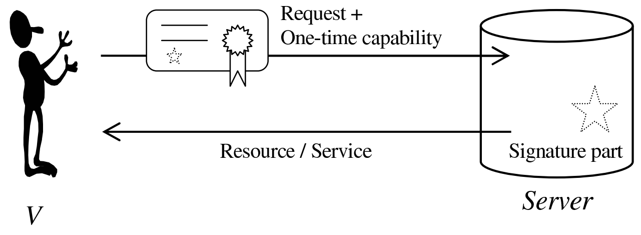

Fig. 3. Access

collection by service personnel. 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.

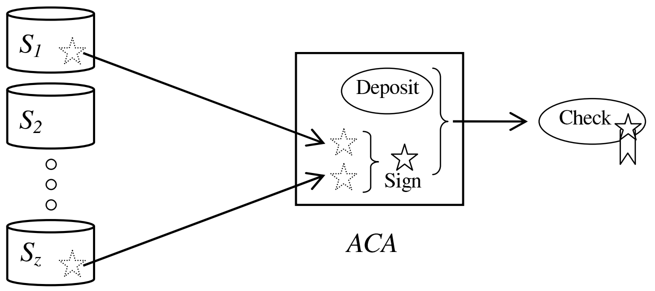

Fig. 4. Detection of double use

4 One-Time Capability Protocols

This section presents the OTC protocols for capability withdrawal, service access, and detection of double use using the notations of Table 1.

4.1 Penalty without Hierarchical Relationships

In the OTC mechanism an electronic check serves as a deterrent against double use of a one-time capability. Common electronic checks [10] are not sufficient in this context because it is not possible to verify that they have been signed without revealing the signature that allows 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 can replace signature of on-line as well as off-line e-checks. For the sake of simplicity this paper

PK_A	public key of entity A.
SK_A	private key of entity A.
$E_K(m)$	plaintext m encrypted with key K.
$\mathrm{h}(m)$	digest of m, cryptographic hash function h applied to data m. $\mathrm{h}: \ (0,1)^* \rightarrow (0,1)^l$ where l is the size of the digest.
$m_1 \parallel m_2$	the concatenation of m_1 and m_2.
$\mathrm{SIGN_A}(m)$	m signed with the private key of the entity A. $\mathrm{SIGN_A}(m) = \{m, E_{SK_A}(\mathrm{h}(m))\}$.

Table 1. Protocol notations

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 withdrawal of a one-time capability, V can prove to ACA that a secret $K \mid \mathrm{h}(K) \in HK$ where $HK = \{\mathrm{h}(K_1), \mathrm{h}(K_2), \cdots, \mathrm{h}(K_n)\}$ will be revealed in case of double use (Section 4.2). A filled e-check is defined as $fc = \mathrm{SIGN_{bank}}(V, ACA, \mathrm{amount}, \mathrm{number})$ where ACA and V could be replaced by information on respective accounts. 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 $K \mid \mathrm{h}(K) \in HK$ is known. A valid e-check is the combination of a deposit and one of the corresponding secrets: $vc = \{sc, K\}$. It can be endorsed and cashed by ACA. Table 2 summarizes those steps.

fc	filled e-check (filled by V and signed by the bank). $fc = \mathrm{SIGN_{Bank}}(\mathrm{payer}, \mathrm{beneficiary}, \mathrm{amount}, \mathrm{number})$ For instance, $fc_1 = \mathrm{SIGN_{Bank}}(PK_V, PK_{ACA}, 10\$, 001)$
sc	signed e-check (also called deposit). $ec = \mathrm{SIGN_{payer}}(fc, HK) \quad \mathrm{where} \quad HK = \{\mathrm{h}(K_1), \mathrm{h}(K_2), \cdots, \mathrm{h}(K_n)\}$ For instance, $ec_1 = \mathrm{SIGN_V}(fc_1, HK_1)$
vc	valid e-check (that can be directly cashed). $vc = \{ec, K\} \quad \mathrm{where} \quad \mathrm{h}(K) \in HK$.

Table 2. E-checks notations

4.2 Phase 1: Capability Withdrawal

One-time capabilities are created by the access control authority. Visitors receive rights and can be authenticated. ACA can verify that a penalty (i.e. an e-check) is embedded within the capability. However, the ACA cannot obtain a valid electronic check during this process (see Figure 2).

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 having some attributes (e.g. employee of a partner company, identity). In this case authentication is required. For instance, attributes certificates and challenge-response protocols can be used. The following protocols do not address this point.

The granted rights and the deposit have to be negotiated. The ACA proposes some authorizations to the visitor and asks for a corresponding e-check. At the end of this phase, V and ACA have agreed on the authorization (rights) that will be provided to V and the penalty (e-check) that will be released if the capability is used twice. A filled e-check is generated: $fc_V = \mathrm{SIGN}_{\mathrm{Bank}}(V, ACA, \mathrm{amount})$

Let V define a set $N = \{1, 2, \cdots, n\}$ that will be used in two cut and choose protocols [3] during the creation and the use of one-time capabilities. The set size n is a security parameter that is determined by the probability of undetected double use (see Section 5.1).

1.1) V generates $K_i \in (0,1)^l \quad \forall i \in N$
V computes $HK_N = \{\mathrm{h}(K_1), \mathrm{h}(K_2), \cdots, \mathrm{h}(K_n)\}$

$K_1, \cdots, K_n$ 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 $m_N = \mathrm{h}(m_1 \| m_2 \| \cdots \| m_n)$

where	$m_i = \mathrm{h}(a_i \| b_i)$	$\forall i \in N$
where	$a_i = \mathrm{h}((c_i \oplus data_i) \| d_i)$	$\forall i \in N$
and	$b_i = \mathrm{h}(c_i \| e_i)$	$\forall i \in N$
where	$data_i = \{K_i, \mathrm{number}\}$	$\forall i \in N$

This construction is necessary for avoiding double use of capabilities. It ensures 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). A valid e-check is obtained when combining this secret and this deposit.

1.3) $V \rightarrow ACA \quad HK_N, m_N, fc_V$

V sends data necessary to create the one-time capability. Before releasing the one-time capability, ACA verifies the e-check.

1.4) $ACA \rightarrow V \quad R \subset N \qquad$ 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) $V \rightarrow ACA \quad c_i, d_i, e_i, K_i \qquad \forall i \in R$
$m_i \qquad \forall i \in \bar{R} \qquad$ where $R \cup \bar{R} = N$

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, number $\forall i \in R$
ACA verifies $\mathrm{h}(K_i) \overset{?}{\in} HK_N$ $\forall i \in R$
ACA verifies $m_N \overset{?}{=} \mathrm{h}(m_1 \| m_2 \| \cdots \| m_n)$
Verify the results. When all $m_i \mid i \in R$ are well-constructed, there is a high probability (see Section 5.1) 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 $m_{\bar{R}} = \mathrm{h}(m_{\bar{r}_1} \| \cdots \| m_{\bar{r}_{n/2}}) \mid \bar{r}_1, \cdots, \bar{r}_{n/2} \in \bar{R}$
ACA computes $HK_{\bar{R}} = \{\mathrm{h}(K_{\bar{r}_1}), \cdots, \mathrm{h}(K_{\bar{r}_{n/2}})\} \mid \bar{r}_1, \cdots, \bar{r}_{n/2} \in \bar{R}$
Both V and ACA suppress verified secrets ($R \subset N$) and keep unrevealed secrets ($\bar{R} \subset N$) to build the one-time capability and the deposit. The ACA requests the deposit.

1.8) $V \rightarrow ACA$ $sc_V = \mathrm{SIGN_V}(fc_V, HK_{\bar{R}})$
V signs the set of unrevealed secrets. This is the deposit that has to be combined with one of the secrets $K_i \mid i \in \bar{R}$ to be a valid e-check. However, the protocol ensures that $K_i \mid i \in \bar{R}$ can only be revealed when V cheats.

1.9) $ACA \rightarrow V$ $capability = SIGN_{ACA}(m_{\bar{R}}, rights,$ validity$)$
The ACA stores the deposit and provides the one-time capability to V. This capability is anonymous, i.e. does not refer to the identity or public key of V. However, the ACA can recognize the capability and thus unlinkability is not ensured (see Section 5.2). As in the simple public key infrastructure, authorizations are application specific and thus defining right format is out of the scope of this paper.

4.3 Phase 2: Service Access with Capability

Visitor V shows his capability to one of the servers $S_1, S_2, \cdots, S_z$ in order to get access to resources or services (see Figure 3). Servers cannot rely on the ACA during this phase.

2.1) $V \rightarrow S$ $resource, capability$
V interacts with a server S that trusts ACA. V requests a resource and provides the one-time capability to prove it is an authorized operation. S verifies that the resource is part of the rights, checks that the capability is signed by the ACA and that the capability is still valid.

2.2) $S \rightarrow V$ $T \subset \bar{R}$ where $|T| = \frac{|\bar{R}|}{2}$
S starts a challenge-response based on cut and choose: The server chooses randomly a subset T (half of the set $\bar{R}$) and sends it to V. This step has two goals: it verifies that the visitor knows the secret corresponding to the capability and forces him to reveal some information that are not sufficient to get a valid e-check but that forbid double use.

2.3) $V \rightarrow S$ $\quad (c_i \oplus data_i), d_i, b_i \quad \forall i \in T$
$\quad c_i, e_i, a_i \quad \forall i \in \bar{T} \quad$ where $T \cup \bar{T} = \bar{R}$

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.

2.4) S computes m_i from $(c_i \oplus data_i), d_i, b_i \quad \forall i \in T$
S computes m_i from $c_i, e_i, a_i \quad \forall i \in \bar{T}$
S verifies $m_{\bar{R}} \overset{?}{=} h(m_{\bar{r}_1} \| \cdots \| m_{\bar{r}_{n/2}}) \mid \bar{r}_1, \cdots, \bar{r}_{n/2} \in \bar{R}$

S verifies that the visitor knows the secret corresponding to the capability. V can be authorized to access the resource.

4.4 Phase 3: Detection of Double Use

Servers $S_1, S_2, \cdots, S_z$ are periodically on-line and thus able to send data to the ACA. If a one-time capability has been used twice, there is a high probability that ACA can retrieve the embedded penalty (i.e. a valid e-check) and use it.

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

Periodically, when the server S_1 is on-line, it sends relevant data to the ACA (dotted star of Figure 4). 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) $S_z \rightarrow ACA \quad c_i \oplus data_i \quad \forall i \in T_{S_z}$
$\quad c_i \quad \forall i \in \bar{T}_{S_z} \quad$ where $T_{S_z} \cup \bar{T}_{S_z} = \bar{R}$

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 c_i and $(c_i \oplus data_i)$ are known. Thus ACA can retrieve $data_i$ and the secret K_i. This secret combined with the deposit is a valid e-check: $vc_V = (sc_V, K_i)$ where $\mathrm{h}(K_i) \in HK_{\bar{R}}$. The ACA can send the e-check to the bank in order to cash it.

5 Discussion

5.1 Security Evaluation

Sections 3 and 4 defines 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 an electronic check is revealed. Two security parameters have to be evaluated:

- n is the number of steps in the cut and choose protocol. It defines the probability that a double use is not detected. Acceptable probability of successful attack depends on the application.
- l is the size of the one-way function outputs. It leads to the probability of revealing a penalty without misbehavior.

Probability of Undetectable Double-Use: Attacks against this scheme require that the visitor can obtain a valid capability embedding a faked e-check. The capability withdrawal 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:

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

It is possible to choose a probability of successful attack as small as required. For instance, if $n = 100$, $p_{attack} \cong 2^{-96}$

Impact of Multiple Use: Since each server keeps trackof 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 on the other hand will be detected by the ACA based on the protocol and the penalty mechanism. But 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:

$$p_{disclose} = \underbrace{\frac{n}{2}}_{|\bar{R}|} \cdot \underbrace{2^{-l}}_{hash} \qquad \text{where } l \text{ is the size of the hash output.}$$

For instance, using $n = 100$ and hash function SHA-1 ($l = 160$ bits), $p_{disclose} \cong 2^{-154}$

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.

A rogue server could perpetrate a pervasive computing denial of service attack by getting a one-time capability without delivering the requested service. Even though this type of attack does not provide any benefit to the attacker, it would prevent further legitimate access to the service by the visitor.

5.2 Computation Complexity and Cost of Privacy

In a pervasive computing environment, it is reasonable to assume that visitors and servers have limited computational power. Since it can be implemented by a powerful workstation, the ACA is on the other hand not impacted by such limitations. In step 1.8 of the protocol the visitor has to sign the deposit using computationally expensive asymmetric cryptography. To suit the limited computational power of the visitor's device, the asymmetric signature in this phase can be substituted by a one-time signature [6] whereby instead of $\mathrm{SIGN_V}(fc_V, HK_{\bar{R}})$, the visitor generates a private one-time key $OTSK$ and the corresponding public key $OTPK$, he then signs the one-time public key with his private key $\mathrm{SIGN_V}(OTPK)$ and uses the one-time private key to sign the e-check $\mathrm{SIGN_{OTSK}}(fc_V, HK_{\bar{R}})$. The one-time signature is efficient but it does not replace asymmetric cryptography that still is required. For instance, when using an efficient scheme such as the one proposed by [6], $l+log_2(l)$ hash functions and one asymmetric signature are required. Even though the total complexity increases with this scheme, the overall result is more efficient than with the straightforward use of digital signatures because signatures can be pre-computed. Table 3 shows the impact of this scheme on computation requirements. With both alternatives, the ACA has to verify the signature of the e-check and sign the capability.

Protocol	**Entity**	**Asymmetric cryptography**	**hash functions**
Withdrawal	V	$\underbrace{\text{1 signature}}_{\text{e-check }^{1)}}$	$\underbrace{l + log_2(l)}_{\text{OTS }^{1)}} + \underbrace{4 \cdot n}_{m_N, HK_N}$
	ACA	$\underbrace{\text{1 verif. sig.}}_{\text{e-check}} + \underbrace{\text{1 signature}}_{\text{capability}}$	$\underbrace{(l + log_2(l))/2}_{\text{verify OTS}} + \underbrace{2 \cdot n}_{m_N, HK_N}$
Access	S	$\underbrace{\text{1 verif. sig.}}_{\text{capability}}$	$\underbrace{n}_{m_{\bar{R}}, HK_{\bar{R}}}$ (cut and choose)

$^{1)}$: can be pre-computed

Table 3. Cryptographic operations of withdrawal and access phases.

The one-time capabilities presented in this paper assure the anonymity of visitors: no server or eavesdropper can get any information on the visitor's identity or public key. However, the ACA authenticates the user during the first phase and thus, in phase 3, ACA can know who owns a capability returned by a server. Thus, the ACA can trace visitors: unlinkability is not ensured. It is possible to offer unlinkability by defining capabilities based on Chaum's blind signature technique [2] as used in electronic cash [3]. In this case, the privacy of the user is fully protected as long as he does not cheat. Unfortunately, the combination of cut-and-choose protocols and blind signature requires more computational power: during the withdrawal phase, n asymmetric operations are required. We are currently working on possible substitutes to this scheme.

5.3 Validity End

In off-line scenarios, it is not possible to rely on a certificate revocation list to verify the validity of long-term capabilities. The alternative to revocation, which still does not suit the off-line nature of the ubiquitous application scenario, consists of short-term certificates that in turn require frequent interaction between the holder of the certificate and the issuer for the renewal of certificates. As opposed to these alternatives, the validity of one-time capabilities presented in this paper does not rely on time. However, it can be interesting to introduce a lifetime for the deposit in order to limit the storage of deposits in time. When the appliances $S_1 \cdots S_z$ cannot afford a real-time clock, the ACA can act as a trusted time server to synchronize the servers when getting in touch with them to collect the access logs. The ACA can thus discard deposits that are no more required by any server.

6 Conclusion

This paper proposes a solution for access control in pervasive computing environments. The major characteristics of pervasive computing environments are lack of infrastructure requiring part of the interactions to be performed off-line with respect to the communication infrastructure and lack of trust between actors requiring new enforcement techniques to prevent misbehavior. Timely validation of authorizations in pervasive environments is not always feasible due to the off-line nature of communications. As a solution we suggest in this paper 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 withdraws 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.

From the computational complexity point of view this solution is comparable to the simple public key infrastructure and it thus can efficiently be deployed in mobile environments offering restricted computational power.

Privacy of users is a major concern in such open environments. The scheme presented in this paper ensures that servers and eavesdroppers cannot know the identity of authorized users. However, the security infrastructure, which is in charge of delivering authorizations and controlling double use, can trace visitors. Unlinkability could be assured by enhancing the one-time capability scheme with the blind signature technique proposed by Chaum [2] but the computational requirement of this approach would be too high to suit the limitations of pervasive devices. Further work will focus on more efficient solutions for privacy.

References

1. Stefan Brands. *A technical Overview of Digital Credentials.* Research Report, February 2002
2. D. Chaum, R.L. Rivest, *Blind Signatures for Untraceable Payments*, Advances in Cryptology, Proceedings of Crypto 82, pp. 199-203.
3. D. Chaum, A. Fiat, M. Naor, *Untraceable Electronic Cash*, Proceedings of Crypto'88, LNCS 403, Springer Verlag, pp. 319-327.
4. J. Camenisch and A. Lysyanskaya. *Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation.* In Advances in Cryptology - Eurocrypt 2001. v 2045 of LNCS, pages 93–118, 2001.
5. J. Camenisch and E. Van Herreweghen. *Design and Implementation of the idemix Anonymous Credential System* In ACM CCS 2002.
6. R. Merkle. *A digital signature based on a conventional encryption function.* In Advances in Cryptology (CRYPTO'87), volume 293 of Lecture Notes in Computer Science, pages 369–378. Springer-Verlag, 1987.
7. S. Micali, R. Rivest, *Micropayments revisited.* In Progress in Cryptology, volume 2271 of LNCS, February 2002.
8. Network Working Group, Request for Comments 2693: *SPKI Certificate Theory*, September 1999.
9. A. Pfitzmann, M. Köhntopp, *Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology*, Workshop on Design Issues in Anonymity and Unobservability (2000).
10. M.H.Sherif, *Protocols for Secure Electronic Commerce.* Advanced and Emerging Communications Technologies Series, CRC Press, March 2000.