ANONYMOUS SERVICES USING SMART CARDS AND CRYPTOGRAPHY

Sébastien Canard
France Telecom R&D
42 rue des Coutures - BP6243
14066 CAEN Cedex
France
sebastien.canard@francetelecom.com

Jacques Traoré
France Telecom R&D
42 rue des Coutures - BP6243
14066 CAEN Cedex
France
jacques.traore@francetelecom.com

Abstract More and more services provided by Internet pose a problem of privacy and anonymity. One cryptographic tool that could be used for solving this problem is the group signature [1, 5, 8]. Each member of the group is able to anonymously produce a signature on behalf of the group and a designated authority can, in some cases, revoke this anonymity.
During the last decade, many anonymous services using this concept have been proposed: electronic auctions [11], electronic cash systems [12, 10, 7], anonymous credentials [2]. But for some other services where the anonymity is essential (such as electronic voting or call for tenders), group signature schemes cannot be applied as they are. For this reason, the authors of [6] proposed a variant that is partially linkable and not openable, called list signature scheme.
In this paper, we first improve the cryptographic tool of [6] by proposing some optional modifications of list signature schemes such as anonymity revocation. We then propose more efficient list signature schemes, by using a smart card to produce the signature. We finally propose some concrete implementations of our proposals. As a result, we obtain more efficient solutions that are useful in many more services.

Keywords: Cryptography, group signature schemes, voting, call for tenders, subscription tickets.

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

Introduction

Group signature schemes have been introduced in 1991 by Chaum and van Heyst [8]. They allow members to sign a document on behalf of the group in such a way that the signatures remain anonymous and untraceable for everyone but a designated authority, who can recover the identity of the signer whenever needed (this procedure is called "signature opening"). Moreover, this type of signature must be unlinkable for everybody but the authority (no one else can decide whether two different valid signatures were computed by the same group member or not). Currently, the most secure group signature scheme is the one of Ateniese et al. [1].

In most cases, it is desirable to protect group member's signing keys in a secure device, such as a smart card. However, these devices are restricted in terms of processing power and memory. To solve this problem, Canard and Girault [5] proposed a smart card based solution where all computations can be efficiently done inside the (tamper-resistant) smart card.

In some applications, group signature schemes can be used as they are. For example, an electronic auction system [11] can be based on any group signature scheme without any modification (other than completing it with some extra cryptographic mechanisms for auction purposes). However, in some other services, it is not possible to use group signatures as they are. It is often desirable to make possible for everybody (and without opening every signature) to link signatures produced by group members. For example, in [6], the authors, in order to develop an electronic voting system, need to introduce a variant of group signature schemes called list signature scheme. These signatures are similar to group signatures except that they are partially linkable and not openable. It permits them to introduce a new electronic voting system that directly uses list signature schemes and that satisfies the fundamental needs of security in electronic voting. In some other cases, we need that the number of anonymous signatures produced by the same member be limited, even in a multi-verifier setting and, again, without opening every signature.

In this paper, we improve the work of [6] in two ways. First, we generalise these list signature schemes by making them optionally openable. Second we propose various implementations of these list signatures (with optionally anonymity revocation) by assuming that a signature is produced by a tamper-resistant device (typically a smart card). Our concrete solutions permit us to use list signatures with optionally anonymity revocation in various applications: electronic voting system or opinion

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

poll, call for tenders and anonymous subscription tickets. Thus, we present some services that can be developed in a mobility context, using a device that is restricted in terms of processing power and memory.
The paper is organised as follows. In the next section, we recall existing solutions and we explain why they are not applicable in some context. Section 2 presents our technical implementations of list signatures with optionally anonymity revocation. Finally, Section 3 shows some examples of services that can be implemented in a smart card using our solutions.

1. Existing Solutions

This section presents some cryptographic tools for anonymous services in Internet. This paper does not consider blind signature schemes or mixnets, which are other tools that provide anonymity. Here, we restrict ourselves to group signature schemes and their variants.

1.1 Group Signature Schemes

There are many entities that are involved in a group signature scheme. A member of the group is denoted by $\mathcal{M}$. The Group Manager $\mathcal{GM}$ is the authority in charge of generating the keys of new members. The Opening Manager $\mathcal{OM}$ is the authority that revokes the anonymity of a group signature, whereas the Revocation Manager $\mathcal{RM}$ is the authority that has the capability of revoking the right of signing of a group member (also called member deletion). We do not consider this kind of revocation in this paper since generic solutions using smart cards (such as [5]) already exist for this problem. Consequently, we will not refer to $\mathcal{RM}$ anymore.

Currently, the most secure group signature scheme is the one of Ateniese, Camenisch, Joye and Tsudik [1] (ACJT for short). In this proposal, if someone wants to become a group member, he computes a secret key x and interacts with $\mathcal{GM}$ in order to obtain a certificate (A, e) (such that $A^e = a_0 a^x \pmod n$ where n, a and a_0 are public and where the factorization of n is only known by $\mathcal{GM}$). An ACJT group signature consists in performing

1 an El Gamal encryption of A, that is computing $T_1 = Az^w$ and $T_2 = g^w$ where z is the encryption public key and w is a random number. $\mathcal{OM}$ is able to decrypt it in case of anonymity revocation by using the discrete logarithm of z in base g as the decryption key.

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

2 a zero-knowledge proof of knowledge of (A, e, x) (a zero-knowledge proof of knowledge on a message M of a value α that verifies the predicate f is denoted by $PK(\alpha : f(\alpha))(M)$).

The group signature scheme of Canard and Girault [5] is based on the use of a tamper-resistant device (such as a smart card) to produce a group signature. Consequently, each member of a group owns a smart card. This proposal makes use of a signature scheme (such as RSA, DSA, Schnorr or GPS) and an encryption scheme that can be symmetric (such as AES) or asymmetric (such as RSA or El Gamal). It is however necessary that this encryption scheme be probabilistic.
During a setup phase, the group manager $\mathcal{GM}$ computes a signature private key sk_G such that he can distribute it without knowing it. This can be done, for example, by using a secret sharing protocol: sk_G is shared by various entities and the associated public key, denoted by pk_G, is computed by the cooperation of all these entities. The signature of the message M with sk_G is denoted by $\mathbf{Sign}_G(M)$.
$\mathcal{OM}$ generates a decryption private key $sk_{\mathcal{OM}}$ and the associated encryption public key $pk_{\mathcal{OM}}$. He keeps the first one secret. If the chosen encryption algorithm is symmetric, then $pk_{\mathcal{OM}} = sk_{\mathcal{OM}}$ and the encryption key must also be kept secret. The encryption of a message M is denoted by $\mathbf{Encrypt}_{\mathcal{OM}}(M)$.
When someone wants to become a new group member, he interacts with the group manager $\mathcal{GM}$ that sends him an identifier $Id_{\mathcal{M}}$ and the signature private key sk_G (consequently, this key is shared by all group members). It is important that $\mathcal{GM}$ knows the link between the identifier $Id_{\mathcal{M}}$ and the identity of the group member $\mathcal{M}$. $\mathcal{M}$ also obtains the encryption key $pk_{\mathcal{OM}}$ from the opening manager $\mathcal{OM}$.
After that, a group member can sign on behalf of the group by using his smart card. Then the algorithm in figure 1 is executed inside the smart card. The verification of a group signature only consists in verifying the

GSign $(M, Id_{\mathcal{M}}, pk_{\mathcal{OM}}, sk_G)$:
C := $\mathbf{Encrypt}_{\mathcal{OM}}(Id_{\mathcal{M}})$;
$\tilde{M}$:= $\mathbf{Concatenate}(M, C)$;
S := $\mathbf{Sign}_G(\tilde{M})$;
return (C, S).

Figure 1. Group Signature of [6].

signature S by using the public key pk_G. If this signature is correct,

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

then the group signature is correct. Finally, if the anonymity needs to be revoked, $\mathcal{OM}$ uses his private key $sk_{\mathcal{M}}$ to decrypt C to obtain $Id_{\mathcal{M}}$. The Group Manager knows the link between this value and the identity of the group member.

1.2 Openable versus Non-openable Signatures

The previous section has shown that anonymity revocation is made possible by the encryption of an identifier. An ACJT group signature includes an El Gamal encryption of a value A that is known by $\mathcal{GM}$. In the group signature scheme of [5], a signature is made of an encryption of the identifier $Id_{\mathcal{M}}$ that is shared by the group member and $\mathcal{GM}$. Consequently, if one wants to make a group signature scheme non openable, it is (almost) sufficient to remove this encryption.
In ACJT, it is then unnecessary to compute $T_1 = Az^w$ and $T_2 = g^w$ anymore. But, to prove his membership, the group member has to make a commitment on A. This can be done by computing $T_1 = Az^w$ and $T_2 = g^w h^{w_1}$ where w and w_1 are random and where the discrete logarithm of z in base g is unknown (which is different from the ACJT group signature scheme where the discrete logarithm of z in base g is known by $\mathcal{OM}$ to open signatures). This is the approach chosen in [6](see above for more details).
In the group signature of [5], it is sufficient to remove the value $C := \mathbf{E}\text{ncrypt}_{\mathcal{OM}}(Id_{\mathcal{M}})$ during the group signature process.

1.3 List Signature Schemes

List signatures have been introduced by Canard and Traoré [6]: they correspond to partially linkable and non openable group signatures.
In a list signature scheme, the time is divided into distinct sequences, each of them being a fixed time period (one hour, one day, one month, the time of an election day, ...). A list signature scheme involves the same protagonists as a group signature scheme, except the Opening Manager. Moreover, in some cases, $\mathcal{GM}$ needs to create a public value that is representative of a sequence: this is executed at the begin of each sequence. Furthermore, everybody is able, taking as input two valid signatures produced during a particular sequence, to test whether or not they have been produced by the same list member or not. However, two signatures produced during two different sequences are unlinkable.

In [6], $\mathcal{GM}$ firstly randomly computes a $2l_p$ bits safe RSA modulus $n = pq = (2p'+1)(2q'+1)$ and chooses random elements $a, a_0, g, h \in_R QR(n)$. When someone becomes a new list member, he obtains a secret key

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

$sk_{\mathcal{M}} = x$ and a list certificate (known by $\mathcal{GM}$) $c_{\mathcal{M}} = (A, e)$ such that $A^e = a_0 a^x \pmod{n}$.
Before the beginning of a new sequence, $\mathcal{GM}$ generates a representative of the sequence, that is a random integer $f \in QR(n)$ (where $QR(n)$ denotes the group of quadratic residues modulo n). A simple solution is to compute $f = (\mathcal{H}(date))^2 \pmod{n}$ where *date* is the date of the beginning of the sequence and $\mathcal{H}$ is a collision-resistant hash function.
Each list member can then sign on behalf of the group by doing the following:

- choosing random values $w, w_1 \in \{0,1\}^{2l_p}$.
- computing $T_1 = Az^w \pmod{n}$, $T_2 = g^w h^{w_1} \pmod{n}$, $T_3 = g^e h^w \pmod{n}$ and $T_4 = f^x \pmod{n}$.
- making a proof of knowledge $U = PK(\alpha, \beta, \gamma, \delta, \zeta, \eta : a_0 = T_1^\alpha / (a^\beta z^\gamma) \wedge 1 = T_2^\alpha / (g^\gamma h^\eta) \wedge T_2 = g^\delta h^\zeta \wedge T_3 = g^\alpha h^\delta \wedge T_4 = f^\beta)(M)$ where M is the message to be signed.

The couple (T_1, T_2) is a commitment of the value A (as explained in Section 1.2) and the signature on M is finally (T_1, T_2, T_3, T_4, U).
The verifier only has to check the validity of the proof of knowledge U to be sure that the signer is actually a member of the list. Finally, since a signer is constrained to compute T_4 by using the representative f of the sequence and his secret key x, he will always compute the same T_4 for a particular sequence. It is consequently possible to know if two or more signatures produced during a particular sequence come from the same member or not.

1.4 Limitations of these Proposals

Group signature is a convenient tool for anonymous auctions, as explained in [11]. List signature schemes can be used to enhance the security of electronic voting system since they have been designed for this purpose in [6]. In some other interesting services (such as call for tenders for example), it can be desirable to have a cryptographic scheme in which:

- each group member can anonymously sign a message on behalf of the group,
- a designated authority can revoke this anonymity and
- it is important to know when two (or more) signatures have been produced by the same user.

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

In this case, existing solutions (group and list signatures) are not suitable and we have to design list signatures with revocable anonymity.
In some other cases, the number of anonymous signatures produced by the same member needs to be limited. We must be sure that each member is unable to produce one more signature per sequence. An anonymous subscription ticket requires this property since the user of the ticket cannot use it more than the number of uses he has subscribed for. Again, group signatures and list signatures are not suitable for this kind of service.
Moreover, the existing list signature schemes are too expensive in terms of processing power and memory since they involve a lot of modular exponentiations. Consequently, it is not possible to use them in a mobility context and a smart card (possibly embedded in a mobile phone or a PDA) cannot be used.

2. List Signature Schemes and Smart Cards

In this section, our aim is to solve the problems identified in the previous one. We consequently design some list signatures with optionally anonymity revocation that can be produced by a tamper-resistant device which is restricted in terms of processing power and memory. In the following, we first propose two solutions, called "Testing and Deciding" and "Semi-Probabilistic Encryption Scheme". We then improve the second one to obtain the "Pseudo Randomizing" method.

2.1 Testing and Deciding

Our first solution implies that we let decide the smart card about its capacity to sign on behalf of the group. In this solution, it is not possible to link two signatures made by the same user during the same sequence. In fact, each member has the possibility to anonymously sign a fixed number of times during a particular sequence.
Our list signature scheme relies on a group signature scheme (that can be the ACJT one or the one described in [5]). Each smart card can sign on behalf of the group by using this group signature scheme. The group signature of a message M is denoted by $\mathbf{GSign}(M)$. Furthermore, we assume that each smart card has a tamper-resistant memory space denoted by Mem and manages a counter cpt.
At the beginning of a new sequence, the group manager $\mathcal{GM}$ randomly creates a representative of this sequence, denoted by $RepSeq$. This element is not simply public but also authenticated and timestamped by $\mathcal{GM}$.

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

The group manager also manages an integer that represents the number of signatures that a list member's smart card can produce during one sequence. This integer, denoted by $NbSig$, can be the same for every list member but can also be different for each smart card: this depends on the service. It must also be authenticated by $\mathcal{GM}$. We denote by $Sig_{\mathcal{GM}}$ the signature on $RepSeq$ and $NbSig$[1].
When a list member wants to sign a message M, he uses his smart card that receives the message M, the representative of the sequence $RepSeq$, the signature $Sig_{\mathcal{GM}}$ and the number $NbSig$ of signatures that it is allowed to produce during this sequence. Then, the smart card executes the algorithm presented in figure 2. For each new signature during the

```
LSign_{T&D} (M, RepSeq, Sig_{GM}, NbSig):
  if ( Verify(Sig_{GM}) ==  OK)
  {
    if (RepSeq is not recorded in Mem);
    {
      Record RepSeq in Mem;
      cpt := 0;
    }
    cpt := cpt + 1;
    if (cpt > NbSig)
    {
      return  Error;
    }
    return  GSign(M);
  }
  else return  Error;
```

Figure 2. "Testing and Deciding" List Signature: $\mathbf{LSign}_{T\&D}$.

sequence $RepSeq$, the smart card increments its proper counter cpt and, consequently, can test whether or not its owner has exceeded the number of authorized signatures for this particular sequence.
A verifier of a correct list signature only knows that the card that produced this signature belongs to a valid list member without knowing which one. He knows that if this signature was produced it means that the list member has the right to do it and that he has not exceeded the number of authorized signatures.
Our list signatures are openable, due to the property of the underlying

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

group signature scheme. To create a non openable list signature, it is sufficient to transform the group signature into a non openable one, as explained in Section 1.2. It is also possible to revoke the anonymity of a signature when the user tries to produce more list signatures than he is entitled to do. Instead of the signature, the smart card can then send an identifier (possibly encrypted with $\mathcal{OM}$'s public key) as $Id_{\mathcal{M}}$ that can be linked to the identity of the cheat.
This list signature scheme can be used in call for tenders, in e-voting or opinion poll and in anonymous subscription tickets.

2.2 Semi-Probabilistic Encryption Scheme

In our second solution, we propose a list signature scheme that necessary supports revocable anonymity. As for [6], it is also possible to link two signatures produced during a particular sequence. Moreover, the list signature will be produced by a smart card since we do not trust the list member.
As for the previous list signature scheme, the following list signature scheme relies on a group signature scheme (that can be the ACJT one or the proposal in [5]). Each smart card can sign on behalf of the list by using this group signature scheme. The latter one must use a probabilistic encryption scheme for anonymity revocation (this is the case for both group signature schemes presented in Section 1.1). The group signature of the message M is denoted by $\mathbf{GSign}(M, C)$ where C denotes the ciphertext derived from the group signature scheme (in ACJT, $C = (T_1, T_2)$ and in [5], $C = \mathbf{Encrypt}_{\mathcal{OM}}(Id_{\mathcal{M}})$).
Roughly, a probabilistic encryption scheme takes as input the message to be encrypted and a freshly generated random number. This random number is not used for the decrypting phase. The core of our second solution is that, for a particular sequence, this random number will be fixed. Consequently, the encryption scheme is deterministic during a sequence and probabilistic for two different sequences: this is what we call a pseudo probabilistic encryption scheme.
At the beginning of a new sequence, the group manager $\mathcal{GM}$ randomly creates a representative of this sequence. This representative is denoted by *RepSeq*. This element is public, authenticated and timestamped by $\mathcal{GM}$ but it may also be required that this number be publicly verifiable: everybody must be convinced that $\mathcal{GM}$ generated it in a truly random manner. For this purpose, we suggest to use the proposal of [6] and to define *RepSeq* as follows: $RepSeq = \mathcal{H}(date)$ where *date* is, for example, the date of beginning of the sequence and where $\mathcal{H}$ is a hash function.
Each list member's smart card holds, from the used group signature

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

scheme, an identifier denoted by $Id_{\mathcal{M}}$. In ACJT, this identifier corresponds to the certificate (A, e) whereas in [5], it corresponds to the value also denoted $Id_{\mathcal{M}}$.
We assume that the smart card can produce a group signature and that the group signature scheme relies on a probabilistic encryption scheme $\mathbf{Encrypt}_G$ (that takes as input a random number and a message). It also has access to a pseudo-random number generator **PRNG** that depends on two variables. Each smart card also holds a secret key K that is only known by it. The pseudo random number generator can be, for example, the ANSI X9.17 pseudo random bit generator (it is also possible to take the FIPS 186 generator family, that is SHA-1 or DES, or any other suitable pseudo random number generator). It takes as input the number of output bits, a seed and a 3DES key. In our scheme, when a list member wants to sign a message M, the seed is the representative of the sequence and the 3DES key is the secret key of the smart card. The size of the output depends on the used probabilistic encryption scheme and is not specified in this paper.
Using this pseudo random number generator, the smart card can then use the output as the random input of the encryption scheme (for example, in the ACJT group signature scheme the input of $\mathbf{Encrypt}_G$ corresponds to w). It finally computes a group signature of the message M by using the output of the encryption scheme. The production of a list signature can be summarized as explained in figure 3. The verifi-

```
LSign_SPES (M, RepSeq, Sig_GM, K):
  if ( Verify(Sig_GM) == OK)
  {
    r := PRNG(K, RepSeq);
    C := Encrypt_G(r, Id_M);
    return GSign(M, C);
  }
  else return Error;
```

Figure 3. "Semi-Probabilistic Encryption Scheme" List Signature: $\mathbf{LSign}_{SPES}$.

cation of such a list signature only consists in verifying the validity of the group signature. The opening of a valid list signature corresponds to the opening procedure of the used group signature scheme.
For a given sequence (and consequently a given representative $RepSeq$) and a given smart card (and consequently a given secret key K), the output r of the pseudo random generator will always be the same, as

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

well as the ciphertext C. Using C, it will be possible to link various signatures made by the same list member during a given sequence. But, for two distinct sequences, the two corresponding representatives $RepSeq1$ and $RepSeq2$, and consequently the two corresponding outputs r_1 and r_2 of the pseudo random generator, will be different. Since r_1 and r_2 will be different, the two corresponding ciphertexts C_1 and C_2 will be different and unlinkable.
This second list signature scheme is suitable for call for tenders and opinion polls applications and supports anonymity revocation. We can modify this proposal to make it optionally with anonymity revocation and this is the purpose of the next section, that develops a solution also based on the use of a pseudo random number generator.

2.3 Pseudo Randomizing

Our third solution is close to the previous one. In this list signature scheme, it is also possible to link two signatures produced during a particular sequence. Again, we do not trust the list member but we trust his smart card.
Our solution is related to the one of [5]. Consequently, it needs an ordinary public-key signature scheme (such as RSA, DSA, GPS, GQ, Schnorr, etc.) with a public key sk_G which is shared by all list member's smart cards while the associated public key pk_G is public. The signature of a message M is denoted by $\mathbf{Sign}_G(M)$. If we require for the list signature scheme to be openable, then our solution requires a (symmetric or asymmetric) encryption scheme. The corresponding encryption key $pk_{\mathcal{OM}}$ is sent to all smart cards (secretly if the scheme is symmetric, and publicly if it is asymmetric) whereas the decryption key $sk_{\mathcal{OM}}$ is kept secret by $\mathcal{OM}$. The encryption of the message M is denoted by $\mathbf{Encrypt}_{\mathcal{OM}}(M)$. In this case, each list member is known by the group manager by an identifier denoted by $Id_{\mathcal{M}}$. This data is stored in the smart card.
The core of our solution is, as for the second proposal, the use of a pseudo random number generator **PRNG** that takes as input a seed denoted by s and a key K that is secret and different for all smart cards.
At the beginning of a new sequence, the group manager randomly creates a representative of this sequence denoted by $RepSeq$ and which is used as the seed of **PRNG**. This can be done as for the second proposal (using the date and a hash function).This element is public, authenticated and dated by $\mathcal{GM}$.
When a list member wants to sign a message on behalf of the list, he uses his smart card. The algorithm executed by the smart card depends

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

on the "openability" of the scheme. If the list signature is not openable, the smart card performs what is described in figure 4. In case the list

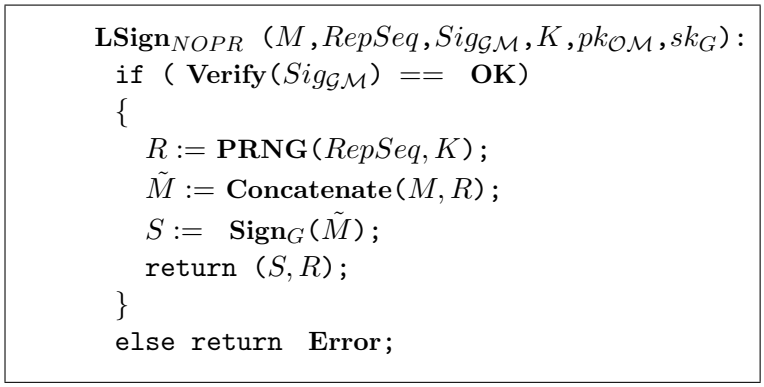

Figure 4. Non-Openable "Pseudo Randomizing" List Signature: **LSign**$_{NOPR}$.

signature scheme supports anonymity revocation[2], the smart card needs to execute the algorithm presented in figure 5. The verification of a list

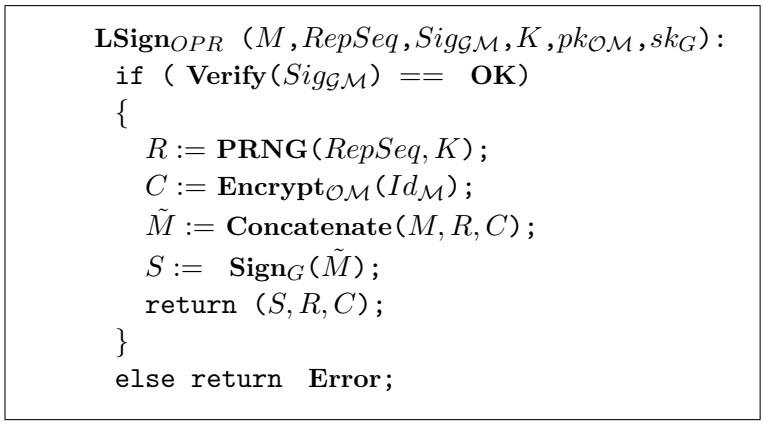

Figure 5. Openable "Pseudo Randomizing" List Signature: **LSign**$_{OPR}$.

signature produced by **LSign**$_{OPR}$ or **LSign**$_{NOPR}$ can simply be done by verifying the signature S, using the public key pk_G. If this signature is correct, then the list signature is considered as valid.
If the signature is openable, $\mathcal{OM}$ can revoke the anonymity by decrypting C to obtain the identifier $Id_{\mathcal{M}}$. The group manager, who knows the link between this data and the actual identity of the list member, can then identify the member who produces this signature.
Finally, everybody is able to link list signatures produced by the same

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

list member during a specified sequence, using the value R. Indeed, for a given sequence (and consequently a given representative $RepSeq$) and a given smart card (and consequently a given secret key K), the output R of the pseudo random generator will be always the same. However, for two distincts sequences, the two corresponding representative $RepSeq1$ and $RepSeq2$ will be different and, consequently, the corresponding outputs of the pseudo random generator will be different and unlinkable.
Our third solution is suitable for call for tenders and opinion polls.

3. Examples of Applications

We have seen in Section 1.4 that the state of the art is not suitable for some services such as opinion polls, call for tenders and anonymous subscription tickets. We explain in the sequel why the tools introduced in the previous sections are more convenient for the above services.

3.1 Opinion Polls

In [6], the authors propose an off-line electronic voting system based on list signature schemes. To vote, each voter produces a list signature of his vote and sends an encryption of the vote and the signature to a ballot box. This system is also directly applicable for an opinion poll service. But, in a mobility context, such as an opinion poll by the means of a mobile phone, the list signature of [6] in not useful since it is too expensive in terms of processing power and memory.
In our proposal, each user can have the possibility to produce a list signature with the help of his mobile phone. For each opinion poll, as for the voting system of [6], the embedded smart card produces a list signature of the voting option of his owner, then encrypts the choice and the signature, and finally sends the encrypted value to the ballot box. It is important to prevent someone from voting many times for a given election and our list signature schemes suit very well.
In this context, it is possible to use $\mathbf{LSign}_{SPES}$, $\mathbf{LSign}_{NOPR}$ or $\mathbf{LSign}_{OPR}$. The list signature can be either openable or not, depending on the organizer of the opinion poll (it can be useful to revoke the anonymity in case of fraud). It can also be possible to use $\mathbf{LSign}_{T\&D}$ with $NbSig = 1$.
For a choice c from the user, the embedded smart card executes the algorithm described in figure 6, where $\mathbf{Encrypt}_{OP}$ is the encryption algorithm related to the opinion poll. The organizer of the opinion poll creates a new sequence and a corresponding representative $RepSeq$ of it. He also signs it. This algorithm, and more particularly the list signature, also needs some more data (such as K, $NbSig$, etc.), denoted by $data$, that are not described in this section. During the counting phase, it is

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

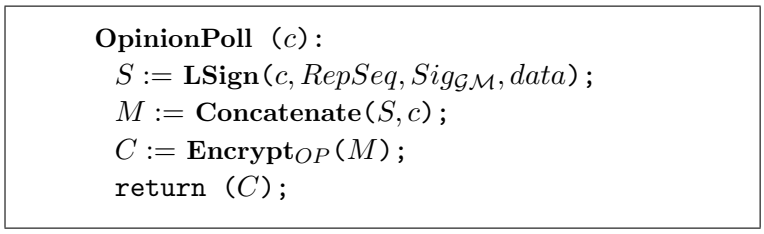

Figure 6. Opinion Poll Service.

sufficient to decrypt each ballot and then to verify the validity of each list signature (to verify that the ballot is valid). Finally, one can publish the result.

3.2 Call for Tenders

In an anonymous call for tenders service, each tenderer can propose an anonymous offer. Moreover, the winner must be identified at the end of the call. In some cases, it can be desirable to prevent a participant from proposing several tenders[3]. Sometimes, we can let the tenderer propose several offers and take the better one, or the last one. In these two cases, it is important to be able to link the offers which come from the same tenderer. For these reasons, our list signature schemes are useful for this purpose.

Our proposal is close to the opinion poll system described above. It first consists in supplying each participant with a smart card. When someone wants to make an offer for a call for tenders, he produces a list signature of his proposal and then encrypts the signature along with his proposal. The call for tenders authority can decrypt each proposal and then test its validity by verifying the list signature scheme. The jury can identify the winner by opening the list signature.

The signing algorithm can be $\mathbf{LSign}_{SPES}$ or $\mathbf{LSign}_{OPR}$. It is also possible to use $\mathbf{LSign}_{T\&D}$ with $NbSig = 1$ if the authority does not want a participant to propose several offers to a given call for tenders.

For a proposal p from a participant, the embedded smart card executes the algorithm described in figure 7, where $\mathbf{Encrypt}_{CT}$ is an encryption algorithm related to the call for tenders (necessary to prevent a fraudulent user from learning something about the proposal of a rival). The authority creates a new sequence and a corresponding representative $RepSeq$. This algorithm, and more particularly the list signature, needs some *data* that depend on the used scheme (see previous section).

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

```
CallTenders (p):
  S := LSign(p, RepSeq, Sig_{G_M}, data);
  M := Concatenate(S, p);
  C := Encrypt_{CT}(M);
  return (C);
```

Figure 7. Call for Tenders Service.

3.3 Anonymous Subscription Tickets

An anonymous subscription ticket permits someone to use a service he had paid for without being traced by the supplier. This is useful for example for cinema tickets, subway or bus tickets, etc.
In our solution, everybody can (non anonymously) obtain a subscription ticket by simply buying it. This one is represented by a numeric data that can be stored in a mobile phone, a PDA, a dongle or a specific smart card. This numeric data corresponds to the number $NbSig$ of "entries" that the customer has bought and is signed, with the identity of the customer, by the issuer. The embedded smart card is able to produce a "Testing and Deciding" list signature with a number of signatures $NbSig$.
When the user wants to go to cinema or to take the bus, he only has to exhibit his mobile phone (for example), which produces a list signature. The embedded smart card knows when all tickets are used and, in this case, refuses to sign.
In this context, the sequence can either correspond to the end of validity of the ticket or the lifetime of the system. In the first case, the representative of the sequence $RepSeq$ corresponds to an identifier of the end date. In the latter case, we can imagine that an embedded smart card can be used, for example, in various cinemas. Thus, $RepSeq$ represents one cinema.
When a smart card is presented to a verification machine, this last one sends it a commitment c and the representative $RepSeq$ (in the first case, this representative is sent after the reception of the date of validity from the smart card). The smart card then produces a list signature $S :=$**LSign**$_{T\&D}(c, RepSeq, Sig_{\mathcal{GM}}, NbSig)$ that can easily be verified by the verification machine.
In some cases, in can be useful to revoke the anonymity of a signature (for example, if it is necessary to know who was on the cinema at a particular date).

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004

Notes

1. For sake of simplicity, we assume that there is only one signature for both values *RepSeq* and *NbSig*, even if, in practice, it can have two differents signatures, since these two values can be obtained at different times.

2. In terms of efficiency, this solution is not very interesting in comparison with the "Semi-Probabilistic Encryption Scheme" solution since there are two computations instead of one. We nevertheless present both solutions to be as general as possible.

3. A tenderer can try to propose various times the same offer to leave nothing to chance.

References

[1] G. Ateniese, J. Camenisch, M. Joye, G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In L. Bellare, editor, Advances in Cryptology-Crypto'2000, volume 1880 of LNCS, pages 255-270. Springer-Verlag, 2000.

[2] J. Camenisch, A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In B. Pfitzmann, editor, Advances in Cryptology-Eurocrypt'2001, volume 2045 of LNCS, pages 93-118. Springer-Verlag, 2001.

[3] J. Camenisch, M. Michels. A Group Signature Scheme based on an RSA-variant. Technical Report RS-98-27, BRICS, Dept. of Comp. Sci., University of Arhus, preliminary version in Advances in Cryptology-EUROCRYPT'98, volume 1514 of LNCS.

[4] J. Camenisch, M. Stadler. Efficient Group Signature Schemes for Large Groups. In B. Kaliski, editor, Advances in Cryptology-CRYPTO'97, volume 1296 of LNCS, pages 410-424. Springer-Verlag, 1997.

[5] S. Canard, M. Girault. Implementing Group Signature Schemes with Smart Cards. Proc. of CARDIS 2002.

[6] S. Canard, J. Traoré. List Signature Schemes and Application to Electronic Voting. Proc. of WCC'03.

[7] S. Canard, J. Traoré. On Fair E-cash Systems based on Group Signature Schemes. Proc. of ACISP 2003, volume 2727 of LNCS, pages 237-248. Springer-Verlag, 2003.

[8] D. Chaum, E. van Heyst. Group Signatures. In D. W. Davies, editor, Advances in Cryptology-Eurocrypt'91, volume 547 of LNCS, pages 257-265. Springer-Verlag, 1991.

[9] G. Maitland, C. Boyd. Co-operatively Formed Group Signatures. In B. Preneel, editor, CT-RSA 2002, volume 2271 of LNCS, pages 218-235. Springer-Verlag, 2002.

[10] G. Maitland, C. Boyd. Fair Electronic Cash Based on a Group Signature Scheme. ICICS 2001, volume 2229 of LNCS, pages 461-465. Springer-Verlag, 2001.

[11] K.Q. Nguyen, J. Traoré. An Online Public Auction Protocol Protecting Bidder Privacy. Information Security and Privacy, 5th Australasian Conference-ACISP 2000, pages 427-442. Springer-Verlag, 2000.

[12] J. Traoré. Group Signatures and Their Relevance to Privacy-Protecting Off-Line Electronic Cash Systems. ACISP'99, volume 1587 of LNCS, pages 228-243. Springer-Verlag, 1999.

Appeared in J.-J. Quisquater, P. Paradinas, Y. Deswarte, A. A. El Kalam (Eds.): CARDIS 2004, 2004