Group Signatures are Suitable for Constrained Devices

Sébastien Canard[1⋆], Iwen Coisel[2⋆⋆],
Giacomo De Meulenaer[2⋆⋆⋆], and Olivier Pereira[2]

[1] Orange Labs - 42 rue des Coutures - BP6234 - F-14066 Caen Cedex - France
[2] Université Catholique de Louvain - B-1348 Louvain-la-Neuve - Belgium
sebastien.canard@orange-ftgroup.com
{iwen.coisel, giacomo.demeulenaer}@uclouvain.be

Abstract. In a group signature scheme, group members are able to sign messages on behalf of the group. Moreover, resulting signatures are anonymous and unlinkable for every verifier except for a given authority. In this paper, we mainly focus on one of the most secure and efficient group signature scheme, namely XSGS proposed by Delerablée and Pointcheval at Vietcrypt 2006. We show that it can efficiently be implemented in a sensor node or an RFID tag, even if it requires 13 elliptic curve point multiplications, 2 modular exponentiations and one pairing evaluation to produce a group signature. This is done by securely outsourcing part of the computation to an untrusted powerful intermediary. The result is that XSGS can be executed in the MICAz (8-bit 7.37MHz ATmega128 microprocessor) and the TelosB (16-bit 4MHz MSP430 processor) sensor nodes in less than 200 ms.
Keywords. Constrained devices, server-aided computation, group signature, anonymity.

1 Introduction

Group signatures have been introduced by Chaum and van Heyst [9], and showed to be extremely useful in various applications such as anonymous credentials, e-cash, e-vote and identity management. These signatures allow any member of a group to sign a document and any verifier to confirm that the signature has been computed by a group member. Moreover, group signatures are anonymous and unlinkable for every verifier except, when needed, for a given authority. While being very appealing, implementing these signature schemes on low-power devices, like sensor nodes or RFID tags, appears to be a particularly challenging task, as the computation of a signature typically requires numerous modular exponentiations or pairing evaluations. For instance, it is necessary to compute 13 elliptic curve point multiplications, 2 modular exponentiations and 1 pairing

⋆ Supported by French ANR PACE project
⋆⋆ Supported by Walloon Region project SEE
⋆⋆⋆ Supported by Walloon Region project Nanotic

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

to produce a DLIN based eXtremely Short Group Signature (XSGS) [14], one of the most efficient and powerful schemes available today.

In this paper, we design a cooperative variant of the XSGS group signature scheme [14]. Our result is efficient enough to permit the group member to be associated to a constrained device which interacts with an intermediary (which can belong to the group member, e.g. a personal computer).

Related Work. The way to embed cryptography into low-power devices has been largely studied by the cryptographic community. One solution is to make pre-computations but this has the drawback of consuming a lot of memory space and thus simply shifts the problem. Another possibility is to modify the cryptographic mechanism to fit the device restrictions. This has already been done in the RFID case [19, 15] or when considering the integration of group signatures in a smart card [8, 36]. This may necessitates important modifications of the initial algorithm, and may imply some stronger (and questionable) assumptions such as, e.g. , tamper-resistance.

In this paper, we focus on a second approach, which consists in studying how a more powerful entity can help a small device to provide a group signature, as introduced in [27] and later used for DAA in *e.g.* [6]. Another approach has also been taken in the CAFE project [10, 11], which consists in designing schemes where a powerful prover interacts with a non-trusted smart card to perform some computations in such a way that the prover is unlinkable w.r.t. the smart card.

Our Contributions. The introduction of an intermediary device in a signature scheme must be carefully understood if one wants to avoid introducing severe security flaws in the system. Our contribution in this direction is threefold.

To begin with, we propose the first complete security model for cooperative group signatures. Our model allows clarifying the exact level of trust that is placed into the intermediary, this trust directly impacting the amount of computation that can be outsourced by the tag. The trust we place in the intermediary is quite limited: even compromised, it is not able to impersonate the signer. With this property, the security of standard group signature systems can be improved: the group members' secrets can be stored in well-protected embedded devices like contactless smart cards instead of being present in their personal computer, which may be unsecure (e.g. infected with a trojan).

Then, we propose a new cooperative group signature scheme, based on the XSGS protocol [14], and prove its security in our model. Our scheme is efficient enough to be implemented on small embedded devices.

We demonstrate this by documenting implementation results on two common wireless sensor nodes, the MICAz and the TelosB sensor nodes, the processor of the TelosB node being also used in contactless secure government electronic ID chips. The on-line phase of the protocol can be completed in less than 200 ms. The off-line phase requires four to six seconds to be completed, but this can be mitigated with a coupon mode (from [29] and the EU project CAFE ESPRIT 7023), which allows a node to pre-compute or pre-load up to 5000 coupons in

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

advance, while satisfying the memory constraints of our devices. We also show that, in our case, new coupons can be added at any time (at home, for instance), without any problem.

This article thus provides fundamental building blocks for the deployment of low-power privacy preserving applications, in contexts where the nodes involved in the applications cannot perform heavy computations, which is the case most of the time for the moment.

Outline. After a brief reminder on group signatures (Section 2), we extend the standard security properties to the cooperative setting (Section 3). Next, we present our cooperative XSGS protocol and prove its security with respect to our new security definitions (Section 4). We eventually demonstrate that our protocol can be executed on small devices, by presenting and discussing the performance of its implementation on small wireless sensor devices, of which the controllers are also sometimes included in RFID tags (Section 5).

2 Definition of Group Signature Schemes

In [9], Chaum and van Heyst introduce the notion of *group signature* schemes [1, 3, 17, 14, 5, 22] where members of the group can sign documents and any verifier can confirm that the signature comes from a group member. Moreover, group signatures are anonymous and unlinkable for every verifier except for a given authority.

2.1 Generic Description of Group Signatures

Formally speaking, a group signature scheme $\mathcal{GS}$ is described by the following polynomial-time procedures, where λ is a security parameter.

- GenParam is a probabilistic algorithm which takes as input 1^λ and which outputs the public parameters of the system $\mathcal{PP} = (\mathsf{Gpk}, \mathsf{Rpk}, \mathsf{params})$ where Gpk is the group's public key, Rpk is the public key of the opening manager and params are public parameters (e.g. mathematical groups, generators,...), it also outputs the group manager's secret key gmsk and the opening manager's secret key rsk;
- UserKeyGen is a probabilistic algorithm which attributes to a user a pair of secret/public key $(\mathsf{usk}, \mathsf{Upk})$ respecting a PKI.
- Join is a probabilistic protocol between the group manager and a new group member U_i to provide the latter with his group secret key $\mathsf{gsk}[i]$. The group manager makes an entry $\mathsf{Tab}[i]$ in the registration table Tab, with the entire transcript of the process.
- Sign is a probabilistic algorithm which takes as input a secret signing key $\mathsf{gsk}[i]$ and a message m and returns the group signature σ on m;
- Verif is a deterministic algorithm which takes as input a message m, and a signature σ and returns either 1 if the signature is valid or 0 otherwise;

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

– Open is a deterministic algorithm which takes as input the opening manager secret key rsk, the registration table Tab, a message m and a signature σ and returns either an identity i or the symbol $\perp$ to indicate a failure, together with a proof τ of this claim;
– Judge is a deterministic algorithm which takes as input the registration table Tab, a message m, a signature σ, an identity i and a proof τ and returns 1 if the proof τ is a valid proof that user i has produced the signature σ and 0 otherwise.

2.2 Security Properties

We outline the formal security properties from the BSZ model, introduced by Bellare et al. [2], that are expected for (dynamic) group signature schemes.

– **Correctness**: a signature produced by a valid user U_i must be accepted by a verifier. Furthermore, the opening of this signature must return the identity of U_i and the judge must validate this opening.
– **Anonymity**: given several signatures of a user (randomly chosen among two users), it is infeasible to distinguish which of these two users have produced this set of signatures.
– **Traceability**: it is infeasible to produce a valid signature which cannot be opened or where the proof outputted by Open cannot be verified. This property must be verified even if several users and the group manager collude.
– **Non-Frameability**: it is infeasible, even for the opening and the group manager, to claim falsely that a signature has been produced by a user.

To prove that a scheme ensures these properties, Bellare et al. [2] define for each of these properties an experiment played by an adversary. Depending on the concerned property, the adversary has several possibilities to interact with the system. For example, an adversary can corrupt some users and thus obtains their group secret keys. In some cases, the group manager can be corrupted and thus the adversary can play his role during a Join procedure. All the possible interactions are realized through oracles which are listed below. Moreover, a list of honest users $\mathcal{HU}$ and one of corrupted users $\mathcal{CU}$ are needed.

– $\mathcal{O}^{\mathsf{CreateU}}$: this oracle generates a new user i using UserKeyGen.
– $\mathcal{O}^{\mathsf{AddU}}(i)$: this oracle adds a new user i in the group using UserKeyGen and the interactive protocol Join. The identity of this new user is added to the list $\mathcal{HU}$. The new public key Upk_i is outputted.
– $\mathcal{O}^{\mathsf{SJoin}}(i)$: during the request to this oracle, the adversary will play the role of the group manager during a Join protocol with a new honest user. First of all, the oracle generates a new user i with UserKeyGen and simulates him during the protocol with the adversary. This new user is added to $\mathcal{HU}$.
– $\mathcal{O}^{\mathsf{UJoin}}$: this oracle simulates the group manager during a Join protocol where the adversary plays the role of the user.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

– $\mathcal{O}^{\mathsf{CrptU}}(i)$: this oracle gives the total control of the user i to the adversary. In other words, the adversary obtains all the information related to this user (secret keys, random values, ...). The member i is moved from $\mathcal{HU}$ to $\mathcal{CU}$.
– $\mathcal{O}^{\mathsf{Reveal}}(i)$: this oracle outputs the secret keys $(\mathsf{usk}[i], \mathsf{gsk}[i])$ of the member i.
– $\mathcal{O}^{\mathsf{SignU}}(i, m)$: this oracle outputs the signature σ on m of the member i and adds the tuple (m, σ, i) in Set (initially empty).
– $\mathcal{O}^{\mathsf{Open}}(m, \sigma)$: this oracle outputs the identity of the user which produced σ.
– $\mathcal{O}^{\mathsf{Choose}_b}(m, i_0, i_1)$: if i_0 or i_1 have not been given as input to $\mathcal{O}^{\mathsf{CrptU}}$ (i.e. $i_0, i_1 \notin \mathcal{HU}$), this oracle outputs the signature σ on the message m of the member i_b (where b is a bit). σ cannot be given in input to the $\mathcal{O}^{\mathsf{Open}}$ oracle.

2.3 Some Group Signature Constructions

In this paper we focus on group signatures based on the use of pairings [3, 17, 14] since they are relatively efficient (compared to standard model based group signatures [22]) and does not need the manipulation of big integers (contrary to [1, 7]). The BBS scheme [3] only considers static group while others [17, 14] are secure in the dynamic case [2]. We here base our study on a variant of the XSGS protocol from [14], which one is described in Appendix A. In fact, our study is also relevant for the scheme in [17] but we have chosen the XSGS one as it includes a complete security study[3]. To prevent the use of the XDH assumption, which may be seen as a too strong assumption, we adapt XSGS (as suggested in [14]) by replacing the El Gamal encryption scheme [18] by the Linear encryption [3], at the cost of a slightly bigger group signature. In a nutshell, a user owns a group secret key gsk and a certificate (A, x) such that $(x+\mathsf{gmsk}).A = G_1 + \mathsf{gsk}.\mathsf{Rpk}_1$, where G_1 is a parameter, Rpk_1 the opening manager public key, and gmsk is the group manager secret key. To sign a message on behalf of the group, a member produces a double encryption of A and a signature of knowledge of m which must prove that the double encryption contains a part of a valid certificate (and is thus linked to the group master secret key).

The main drawback of XSGS is that it needs one pairing evaluation, many elliptic curve point multiplications and modular exponentiations (13 and 2 respectively) to produce a group signature. But this is the case for many other group signature schemes. In fact, this complexity places XSGS as one of the most efficient group signature scheme which is today available. Our purpose in this paper is now to propose a secure (see next section) and efficient (see Sections 4 and 5) cooperative version of XSGS.

3 Security of Cooperative Group Signatures

A cooperative group signature [27] allows a group member, with constrained resources, to be helped by some powerful entity, called an intermediary, in the

[3] In order to reach the full anonymity property described in [2], the proposal in [14] uses the Naor-Yung methodology [30], and thus twice the same encryption scheme with the same message together with a proof. The scheme in [17] does not totally uses this method and the resulting security is not discussed in the paper.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

production of the group signature. The group member is here the constrained device, while the role of the intermediary is played by a more powerful entity, e.g. a personal computer. Note that the cooperative system also requires verifiers, which have the same role as in standard group signatures. The problem on which we focus is that the intermediary may have some more knowledge that is traditionally not available for the adversary. We thus give all the assumptions about the intermediary and we next adapt the security properties of a group signature scheme in this context. This work has not been totally done in [27, 6].

3.1 Concept and Assumptions

We first assume that the intermediary does not know any secret information and thus, at the beginning of a protocol, the signer may transfer some data to the intermediary in order to decrease its computation complexity. Consequently, an adversary may obtain more information to break the security of the scheme, e.g. by eavesdropping. It is thus obvious that we must model all her new abilities.

In the cooperative setting, the "standard" adversary (meaning the adversary of the original group signature scheme) can be improved in three different manners. Firstly, the adversary can obtain from the intermediary all data that have been sent by the device. Secondly, she can eavesdrop all communications during a signature protocol (at least the shared data but potentially more information). Finally, she can impersonate the intermediary, and thus obtain all the exchanged information. Moreover, she can learn all the choices made by the intermediary during the protocol (e.g. random values). It is clear that the last adversary is more powerful than the two others. Consequently, we only formally model this one in the cooperative setting and thus introduce the new following oracle.

– $\mathcal{O}^{\mathsf{PartialSign}(i,m)}$: this oracle simulates for the adversary the behaviour of the user i realizing the cooperative signature of a message m. Several exchanges between the oracle and the adversary can be done as it simulates a real cooperative protocol execution between a constrained device and the adversary playing the role of the intermediary.

Based on this new adversary's ability, we must adapt the security properties of group signature schemes to the cooperative setting, based on the formal model of Bellare et al. [2].

3.2 Adaptation of the Correctness

The first security property concerns the correctness and focus on the signature, the opening and the judge verification. In our cooperative context, we decide that the intermediary realizes the connection with the "outside world". Thus, if it decides to send a false signature, the signer cannot do anything to rectify this. Consequently, it seems impossible to ensure such correctness when the adversary can actively participate during the experiment. Nevertheless, the cooperative protocol should at least ensure the "standard" correctness property.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

Definition 1. *The* correctness predicate *of a group signature scheme, denoted* $\mathcal{E}_{corr}^{GSS}$*, is verified for a user* i *and a message* m *if and only if the following conditions are verified:*

$$\textsc{Verif}(m, \textsc{Sign}(\mathsf{gsk}[i], m)) = 1 \wedge \textsc{Open}(\mathsf{rsk}, \mathsf{Tab}, m, \textsc{Sign}(\mathsf{gsk}[i], m)) = (i, \tau)$$
$$\wedge\ \textsc{Judge}(\mathsf{Tab}, m, \textsc{Sign}(\mathsf{gsk}[i], m), i, \tau) = 1$$

We denote $\mathcal{E}_{corr}^{GSS}(i, m) = 1$ *if this predicate is true, and 0 otherwise.*

A cooperative scheme ensures the correctness *property if there exists a negligible function* $\epsilon(\lambda)$ *such that:*

$$\forall (i, m): \quad Pr[\mathcal{E}_{corr}^{GSS}(i, m) = 0] < \epsilon(\lambda)$$

Remark 1. In case the intermediary is "behind" the signer and has no link with the verifier, it is possible to define a strong cooperative completeness where the intermediary can be corrupted. Since this is not the practical case we are studying, we will not consider it.

We say that a protocol is the *cooperative version* of another one (called the standard one) if their outputs are constructed identically. Then, a cooperative version of a protocol ensures the correctness property if the standard is also correct in the BSZ model [2].

3.3 Adaptation of the Anonymity

From the anonymity point of view, it is possible to assume that the signer and the intermediary live in a personal environment. In fact, as the intermediary can most of the time recognize the signer by some other means, allowing it to know the user identity does not introduce a threat. In this case, the cooperative scheme should only verifies the "standard" anonymity property. More precisely, if the initial group signature scheme provides anonymity (in the BSZ sense), then a cooperative version necessarily verifies this "weak" anonymity property. By doing this assumption, it is generally possible to transfer more data to the intermediary and thus to reduce the signer's complexity by a better factor.

Definition 2 (Anonymity Property). *A cooperative scheme ensures the* anonymity *property if there exists a negligible function* $\epsilon(\lambda)$ *such that:*

$$\Big| Pr\big[\mathcal{A}(\mathsf{gmsk}) \to 1 \mid b = 1\big] - Pr\big[\mathcal{A}(\mathsf{gmsk}) \to 1 \mid b = 0\big] \Big| < \epsilon(\lambda)$$

for any polynomial adversary $\mathcal{A}$*, who have access to* $\mathcal{O}^{\mathsf{CreateU}}$*,* $\mathcal{O}^{\mathsf{AddU}}$*,* $\mathcal{O}^{\mathsf{SJoin}}$*,* $\mathcal{O}^{\mathsf{UJoin}}$*,* $\mathcal{O}^{\mathsf{CrptU}}$*,* $\mathcal{O}^{\mathsf{Reveal}}$*,* $\mathcal{O}^{\mathsf{SignU}}$*,* $\mathcal{O}^{\mathsf{Open}}$ *and* $\mathcal{O}^{\mathsf{Choose}_b}$*.*

Remark 2. In some cases, being unlinkable *w.r.t.* the intermediary is a really important issue and needs to be studied. For the completeness of the model, it is consequently possible to provide a stronger definition for the cooperative anonymity property. The adversary is thus additionally given access to the $\mathcal{O}^{\mathsf{PartialSign}}$ oracle in the above experiment.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

3.4 Adaptation of the Traceability

Concerning the two remaining security properties, it is necessary to give access to the adversary the $\mathcal{O}^{\mathsf{PartialSign}(i,m)}$ oracle in the cooperative versions of the related experiments.

Definition 3 (Traceability Property). *The* traceability predicate *of a group signature scheme, denoted $\mathcal{E}^{GSS}_{trac}$, is verified for (m, σ) if and only if the following conditions are verified:*

$$\textsc{Verif}(m,\sigma) = 1 \wedge \Big[\textsc{Open}(m,\sigma,\mathsf{rsk}) = \perp \vee \Big(\textsc{Open}(m,\sigma,\mathsf{rsk}) = (\mathsf{Upk},\tau) \wedge \textsc{Judge}(\sigma,m,\tau,\mathsf{Upk}) = \perp\Big)\Big]$$

We denote $\mathcal{E}^{GSS}_{trac}(m,\sigma) = 1$ if this predicate is true, and 0 otherwise.

A cooperative scheme ensures the traceability *property if there exists a negligible function $\epsilon(\lambda)$ such that for any polynomial adversary $\mathcal{A}$, who have access to $\mathcal{O}^{\mathsf{CreateU}}$, $\mathcal{O}^{\mathsf{AddU}}$, $\mathcal{O}^{\mathsf{SJoin}}$, $\mathcal{O}^{\mathsf{UJoin}}$, $\mathcal{O}^{\mathsf{CrptU}}$, $\mathcal{O}^{\mathsf{Reveal}}$, $\mathcal{O}^{\mathsf{SignU}}$, $\mathcal{O}^{\mathsf{Open}}$, $\mathcal{O}^{\mathsf{PartialSign}}$:*

$$Pr\Big[\mathcal{A}(\mathsf{gmsk}) \rightarrow (m,\sigma) : \mathcal{E}^{GSS}_{trac}(m,\sigma) = 1\Big] < \epsilon(\lambda).$$

Note that the traceability predicate is verified even when the user, which possess Upk, is corrupted, as in the standard security definition [2].

3.5 Adaptation of the Non-frameability

We next study the non-frameability property, for which we introduce a list Set which contains all valid signatures outputted during the experiment (i.e. realized by the $\mathcal{O}^{\mathsf{SignU}}$ oracle).

Definition 4 (Non-Frameability Property). *The* non-frameability predicate *of a group signature scheme, denoted $\mathcal{E}^{GSS}_{NonFra}$, is verified for (m, σ) if and only if the following conditions are verified, where $(\mathsf{Upk}_i, \tau) = \textsc{Open}(m, \sigma, \mathsf{rsk}, \mathsf{Tab})$:*

$$\textsc{Verif}(m,\sigma) = 1 \wedge (m,\sigma,i) \notin \mathsf{Set} \wedge i \in \mathcal{HU} \wedge \textsc{Judge}(m,\sigma,\tau,\mathsf{Upk}_i,\mathsf{Tab}) = 1.$$

We denote $\mathcal{E}^{GSS}_{NonFra}(m,\sigma) = 1$ if this predicate is true, and 0 otherwise.

A cooperative scheme ensures the non-frameability *property if there exists a negligible function $\epsilon(\lambda)$ such that for any polynomial adversary $\mathcal{A}$, who have access to $\mathcal{O}^{\mathsf{AddU}}$, $\mathcal{O}^{\mathsf{CrptU}}$, $\mathcal{O}^{\mathsf{Reveal}}$, $\mathcal{O}^{\mathsf{SignU}}$, $\mathcal{O}^{\mathsf{Open}}$, $\mathcal{O}^{\mathsf{PartialSign}}$:*

$$Pr\Big[\mathcal{A}(\mathsf{gmsk},\mathsf{rsk}) \rightarrow (m,\sigma) : \mathcal{E}^{GSS}_{NonFra}(m,\sigma) = 1\Big] < \epsilon(\lambda).$$

4 The Cooperative Version of XSGS

Our aim is now to adapt the XSGS protocol [14] (described in Appendix A) in a secure cooperative manner such that it can be embedded in a RFID tag. For this reason, we consider that the tag is not anonymous w.r.t. the reader. We thus describe a cooperative version of the XSGS scheme and prove its security.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

4.1 Protocol Description

To obtain the best possible gain in terms of efficiency, we adapt the XSGS protocol such that the user will not be anonymous for the intermediary. However, we will prove that all the other security properties remain. Thus, at the beginning of a signature protocol, the user transmits its certificate (A, x), which is part of its group secret key (see Appendix A for details) to the intermediary which will performs all the computations related to this certificate. The user keeps secret his group secret key and computes all values based on it. The obtained cooperative version of the protocol is described in Figure 1. The efficiency gain from the user's point of view is huge as there only remains one point multiplication to compute instead of one pairing, 13 point multiplications and 2 modular exponentiations in the DLIN based XSGS protocol. In this section we use notations introduced in Section 2 and Appendix A. In a nutshell, a group member owns a group secret key gsk and a certificate (A, x) obtained during the JOIN protocol. The revocation manager has two couples of key $(\mathsf{rsk}_1, \mathsf{Rpk}_1), (\mathsf{rsk}_2, \mathsf{Rpk}_2)$. The group public key is denoted GMpk. Finally, $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi)$ is a bilinear environment (see Appendix A.1) and $\mathcal{H}$ denotes a cryptographically secure hash function.

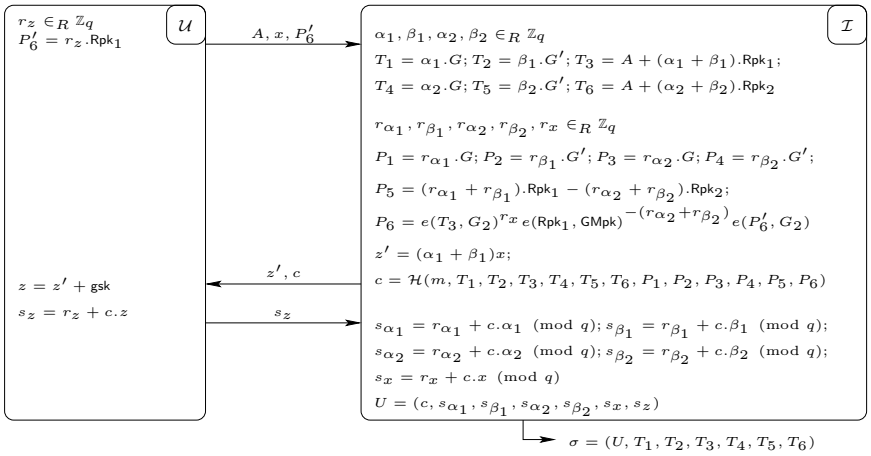

Fig. 1. Sign procedure of the cooperative XSGS Protocol.

4.2 Security Analysis

Intuitively, the transmission of the certificate does not introduce any security flaw since both **Traceability** and **Non-Frameability** assume that even the group manager cannot break these properties. As this entity knows all users' certificates, it should be also hard for an active adversary to break them. Nevertheless we will formally prove these results. Note that our proof are in the

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

random oracle model as in the original paper [14]. First of all, we prove that our cooperative protocol perfectly realizes a XS group signature, and thus that it verifies the correctness property.

Theorem 1. *The Coop-XSGS protocol ensures the Correctness property.*

Proof. Recall that for this property we assume that the intermediary has a honest behaviour and executes perfectly his part of the protocol. We remark that only the signature of knowledge U slightly deviates from the standard one and we need to verify its correctness. More precisely, the deviation is on the P_6 value. Based on the pairing property (see Section A.1) it is obvious that:

$$\begin{aligned} P_6 &= e(T_3, G_2)^{r_x} e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} e(P'_6, G_2) \\ &= e(T_3, G_2)^{r_x} e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} e(\mathsf{Rpk}_1, G_2)^{r_z} \end{aligned}$$

Thus, the whole group signature is computed identically as in the standard protocol. Consequently, the cooperative protocol is correct. □

Theorem 2. *The Coop-XSGS protocol ensures the Anonymity property.*

Proof. As explained in Section 3.3 and since the scheme ensures correctness, the proposed cooperative scheme ensures the anonymity. □

Theorem 3. *The Coop-XSGS protocol ensures the Traceability property.*

Proof. This proof is obvious since the adversary has no more information than the adversary in the standard model. Indeed, this property is verified even when the adversary represents a collusion of members (thus knowing their group secret keys and certificates). Thus, the cooperative version of the XSGS protocol trivially verifies the traceability property. □

Theorem 4. *The Coop-XSGS protocol ensures the Non-Frameabilty property.*

Proof. In the original proof (see proof of Theorem 11 in [14]), the authors use the "unforgeability techniques" to retrieve the certificate and the group secret key used in a signature outputted by an adversary. Thus, they build an algorithm which interacts with this adversary in order to break the discrete logarithm either in $\mathbb{G}_1$ or in $\mathbb{G}_2$ (depending of the retrieved group secret key). This proof only works if the adversary does not know the group secret key of the targeted user. As the JOIN procedure does not leak any information about it, their proof is correct. For the cooperative protocol, this proof can also be applied if we prove that an active adversary cannot learn any information about this secret key.

For this purpose, we first highlight the fact that the protocol between the constrained device and the intermediary can be interpreted as a Schnorr protocol (see [33] for further details) which has been proven to be a zero-knowledge proof of knowledge. As a consequence, the values P'_6 and s_z do not reveal any information about gsk. It is next obvious that the intermediary has no information about the value r_z under the discrete logarithm assumption. Consequently,

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

given a fixed value of s_z, whatever the value gsk is equal to, there exists one value r_z such that $s_z = r_z + ((\alpha_1 + \beta_1)x + \mathsf{gsk})c$. As a result, a perfect simulation of $\langle P'_6, c, s_z \rangle$ can be realized as for the zero-knowledge property of Schnorr's protocol (see [33]). Then, if r_z is uniformly chosen in $\mathbb{Z}_p$, the group secret key of the user is perfectly hidden in s_z. □

Theoretically speaking, this protocol appears to be really efficient. However, in order to demonstrate this efficiency in practice, we describe in the next section an implementation of this new cooperative protocol.

5 Implementation of Coop-XSGS in a RFID Tag

To assess the suitability of the cooperative XSGS protocol for small portable devices, we have implemented it using a wireless sensor node for the prover and a laptop for the powerful helping entity. Wireless sensor nodes are small autonomous devices equipped with a small microcontroller and a transceiver. In this work, we studied the performances of the protocol on two representative sensor node platforms, the MICAz [12] equipped with an 8-bit 7.37MHz ATmega128L microprocessor and the TelosB [13], based on the 16-bit 4MHz MSP430 processor. These devices are conceptually quite close to contactless smart cards. For instance, the TI RF360 chip for contactless secure government electronic ID embeds the same MSP430 processor as the TelosB node [34]. Therefore, although we consider an active device, the results of our implementation can easily be extended to platforms such as contactless smart cards.

The protocol implemented follows the cooperative sign procedure described in Figure 1. The most costly operation for the prover is the point multiplication $P'_6 = r_z.\mathsf{Rpk}_1$. It can be computed prior to the interactions with the intermediary, either during idle time (in case of an active device) or precomputed and preloaded on the tag. The latter case corresponds to the *coupon* mode, as in [20], where a *coupon* is a pair of $(r_z,\ r_z.\mathsf{Rpk}_1)$ loaded on the device. In the following, the operation leading to P'_6 is denoted as the off-line phase, although it might still be computed on-line in the case of a passive device avoiding coupons.

Concerning the pairing parameters, we chose an asymmetric pairing, as it allows to use small-length inputs on the tag, reducing therefore the storage and bandwidth costs. For the elliptic curves on which the pairing is applied, we selected the so-called type D curves (following the classification of [25]), i.e., the ordinary curves with embedding degree 3, 4 or 6 known as MNT curves [28]. This type of curve ensures a small input length (around 170 bits for an embedding degree 6) together with an efficient pairing computation [25].

The prover computations were coded in TinyOS [35] on both the MICAz and TelosB sensors. For the point multiplication, we extended the TinyECC library [24] to support the MNT curves. The parameters of the used curve were taken from the PBC library [26] written by B. Lynn. They are labeled as the d159 parameters, where 159 is the size of the base field of the curve. Their security level is equivalent to the hardness of the discrete log problem on $6 \cdot 159 =$

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

Table 1. Running times of the various phases of the cooperative XSGS protocol. The on-line phase of the sign procedure takes less than 200 ms.

Phase	Detail	Time (ms)
Off-line Sign		
	Prover (MICAz)	3800
	Prover (TelosB)	6400
	Intermediary	195
On-line Sign		
	Prover (MICAz)	7
	Prover (TelosB)	9
	Intermediary	65
	Communication	120
	Total	$<$ **200**
Verify	–	55
Open	–	35

954 bits. Parameters for a higher security could be selected if required. On the intermediary side, the computations were implemented in C using the GMP [21] library and the PBC library to achieve the pairings. They took place on a laptop equipped with a 64-bit 1.4 GHz Intel Core 2 Duo. The pairing computations were rather fast in this setting: 11 ms were sufficient to perform a Tate pairing.

The running times for the various parts of the sign procedure are given in Table 1. The on-line phase of the protocol is performed in less than 200 ms with both sensor nodes. The communication delay and the intermediary computations are the main components of the on-line phase latency. The time required by the prover computations, i.e. , the calculation of s_z, is marginal (<10 ms).

In the off-line phase, the computation of the point multiplication giving P'_6 lasts about 4 and 6 seconds on the MICAz and TelosB respectively. While reasonably efficient, the TinyECC library, on which our implementation is based, is however significantly slower than recently proposed ECC implementations on sensor nodes. We therefore expect the point multiplication to be significantly faster using the same techniques as for instance the implementation proposed in [23], which performs a fixed-base point multiplication in less than a second on a 192-bit elliptic curve group. As a result, even if the prover device has to compute the point multiplication P'_6 during the on-line phase, the whole procedure can be done within a few seconds (much less if a dedicated hardware ECC engine is available on the tag).

The fast on-line phase of the cooperative protocol makes group signatures of practical interest for small devices like contactless smart cards. By contrast, signing with the original XSGS protocol would require a much longer interaction between the passive device and the reader. To give a rough idea, a pairing evaluation takes about 5.5 second on the MICAz (with the TinyPBC library [31]) and an ECC point multiplication requires 0.71 second with the implementation from [23] (there are 13 of these in XSGS): the whole protocol would take on the order of 15 seconds with the MICAz.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

Table 2. Storage requirements (in bytes) of our implementation on the sensor nodes. In coupon mode, 21 B must be added per stored coupon.

Mode	Memory	MICAz	TelosB
Coupon	RAM	1067 (26%)	1071 (10%)
	ROM	23764 (18%)	14470 (29%)
No Coupon	RAM	1699 (41%)	1739 (17%)
	ROM	36576 (28%)	19164 (39%)

The original XSGS protocol would also consume a considerable amount of memory. On the other hand, the memory usage of the cooperative protocol is relatively modest (Table 2), even when the ECC point multiplication is performed on the node. The tiny operating system already consumes a significant fraction of the used memory (about 700 B RAM and a little more than 10 kB ROM on both nodes). A coupon $(r_z, r_z.\mathsf{Rpk}_1)$ requires normally 60 B, i.e. three 20-B field elements, but it can be reduced to a little more than 20 B using point compression and a PRNG sequence for storing all the r_z, as done in [20]. As the coupons can be placed in RAM or ROM, their storage in both the MICAz and TelosB is not a problem. Considering the available memory resources, the MICAz and the TelosB could be filled with more than 5000 and 2500 coupons respectively, which is more than practical for many applications.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *CRYPTO'00*, volume 1880 of *LNCS*, pages 255–270. Springer, 2000.
2. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.
3. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.
4. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
5. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.
6. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security 2004*, pages 132–145, 2004.
7. Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2004.
8. S. Canard and M. Girault. Implementing group signature schemes with smart cards. In *CARDIS '02*, pages 1–10. USENIX, 2002.
9. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

10. David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO 92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, 1992.
11. Ronald Cramer and Torben P. Pedersen. Improved privacy in wallets with observers (extended abstract). In *EUROCRYPT 93*, pages 329–343, 1993.
12. CrossBow. MICAz low-power wireless sensor module. http://www.xbow.com/-Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf, April 2010.
13. CrossBow. TelosB low-power wireless sensor module. http://www.xbow.com/-Products/Product_pdf_files/Wireless_pdf/TELOSB_Datasheet.pdf, April 2010.
14. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In *VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.
15. M. Feldhofer, S.Dominikus, and J. Wolkerstorfer. Strong authentication for rfid systems using the aes algorithm. In *CHES 2004*, volume 3156 of *LNCS*, pages 357–370. Springer, 2004.
16. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
17. Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 455–467. Springer, 2005.
18. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
19. M. Girault and D. Lefranc. Public key authentication with one (online) single addition. In *CHES 2004*, volume 3156 of *LNCS*, pages 413–427. Springer, 2004.
20. Marc Girault, Loic Juniot, and Matt Robshaw. The Feasibility of On-the-Tag Public Key Cryptography. In *Workshop on RFID Security – RFIDSec'07*, Malaga, Spain, July 2007.
21. GMP. The GNU Multiple Precision Arithmetic Library. Available online at http://gmplib.org/, April 2010.
22. J. Groth. Fully anonymous group signatures without random oracles. In *ASI-ACRYPT 2007*, pages 164–180, 2007.
23. Christian Lederer, Roland Mader, Manuel Koschuch, Johann Großschdl, Alexander Szekely, and Stefan Tillich. Energy-Efficient Implementation of ECDH Key Exchange for Wireless Sensor Networks. In *Information Security Theory and Practices — WISTP 2009*, pages 112–127. Springer Verlag, LNCS 5746, September 2009.
24. A. Liu and P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN*, pages 245–256, April 2008.
25. B. Lynn. *On the implementation of pairing-based cryptosystems.* PhD thesis, Stanford University, 2007.
26. B. Lynn. PBC, the Pairing-Based Cryptography Library. Available online at http://crypto.stanford.edu/pbc/, April 2010.
27. G. Maitland and C. Boyd. Co-operatively formed group signatures. In *CT-RSA 2002*, volume 2271 of *LNCS*, pages 218–235. Springer, 2002.
28. Atsuko Miyaji, Masaki Nakabayashi, and Shunzou TAKANO. New explicit conditions of elliptic curve traces for fr-reduction, 2001.
29. David Naccache, David M'Raïhi, Serge Vaudenay, and Dan Raphaeli. Can d.s.a. be improved? complexity trade-offs with the digital signature standard. In *EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 77–85. Springer, 1994.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

30. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.
31. L.B. Oliveira, M. Scott, J. Lopez, and R. Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on*, pages 173–180, June 2008.
32. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, Lecture Notes in Computer Science, pages 223–238. Springer, 1999.
33. C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.
34. TexasInstruments. Texas Instruments Government Electronic Identification. http://www.ti.com/rfid/docs/manuals/brochures/govid_trifold.pdf, April 2010.
35. TinyOS. An open-source operating system designed for wireless embedded sensor networks . http://www.tinyos.net/, April 2010.
36. S. Xu and M. Yung. Accountable ring signatures: A smart card approach. In *CARDIS'04*, pages 271–286. Kluwer, 2004.

A XSGS Group Signature

In this section, we give some useful tools and next focus on the XSGS group signature scheme, using additive notations.

A.1 Some Notations and Tools

Bilinear Groups. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order q and let ψ be an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. G_1 (resp. G_2) is a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). Finally, let e be a computable bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ such that $e(G_1, G_2) \neq 1$ and for all $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}$, $e(a.P_1, b.P_2) = e(P_1, P_2)^{ab}$. $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2, e, \psi)$ is called a bilinear environment.

Zero-Knowledge Proofs of Knowledge. Roughly speaking, a Zero Knowledge Proof of Knowledge (ZKPK) is an interactive protocol during which an entity $\mathcal{P}$ proves to a verifier $\mathcal{V}$ that he knows a set of secret values $\alpha_1, \ldots, \alpha_q$ verifying a given relation R without revealing anything else. These protocols are also used to prove that some public values are well-formed from secret values known by the prover. It is possible to transform these protocol into non-interactive proof of knowledge, generally called signature of knowledge, using the Fiat-Shamir heuristic [16].

In the sequel, we denote by $\mathrm{SoK}(\alpha_1, \ldots, \alpha_q : \mathsf{R}(\alpha_1, \ldots, \alpha_q))$ a signature of knowledge of the secrets $\alpha_1, \ldots, \alpha_q$ verifying the relation R. We also define π as the interactive protocol between a prover $\mathcal{P}$, on input $\alpha_1, \ldots, \alpha_q$ and R and a verifier $\mathcal{V}$ on input R and which allows $\mathcal{P}$ to prove that she knows the secrets in a zero-knowledge manner. The output of $\mathcal{V}$ is either 1 if the prover is accepted and 0 otherwise.

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

A.2 Decision Linear Problem and Encryption

The Decision Linear Problem has been introduced in [3] and is defined as follows.

Definition 5. *Given $G, G', H, \alpha.G, \beta.G', \gamma.H \in \mathbb{G}$ as input, the Decision Linear Problem consists to decide if $\gamma = \alpha + \beta$ or not.*

A great advantage of this problem is that it is still a hard problem even in bilinear groups where the DDH problem is easy. Based on this problem, the authors of [3] introduced a new encryption scheme called Linear Encryption:

- GenParam(1^λ): let $\mathbb{G}$ be a group of prime order q. Select three generators G, G' and Rpk such that there exists $\mathsf{rsk}_1, \mathsf{rsk}_2 \in \mathbb{Z}_q$ which verify $\mathsf{Rpk} = \mathsf{rsk}_1.G = \mathsf{rsk}_2.G'$. The public-key of the system is the tuple (G, G', Rpk) while the secret key is $(\mathsf{rsk}_1, \mathsf{rsk}_2)$.
- Enc(m): to encrypt the message $m \in \mathbb{G}$, this algorithm selects two random values $\alpha, \beta \in \mathbb{Z}_q$ and computes $T_1 = \alpha.G, T_2 = \beta.G', T_3 = m + (\alpha + \beta)\mathsf{Rpk}$. The encrypted message is (T_1, T_2, T_3).
- Dec(T_1, T_2, T_3): to decrypt a message, this algorithm computes $m = T_3 - \mathsf{rsk}_1.T_1 - \mathsf{rsk}_2.T_2$.

To define the parameters of this scheme verifying is $\mathsf{Rpk} = \mathsf{rsk}_1.G = \mathsf{rsk}_2.G'$, a solution is described by the next steps:

- choose a random generator $G \in \mathbb{G}$;
- choose a random value $\mathsf{rsk} \in_R \mathbb{Z}_q$ and compute $G' = \mathsf{rsk}.G$;
- choose a first secret key $\mathsf{rsk}_1 \in_R \mathbb{Z}_q$ and compute $\mathsf{Rpk} = \mathsf{rsk}_1.G$;
- compute $\mathsf{rsk}_2 = \mathsf{rsk}_1/\mathsf{rsk} \pmod q$.

A.3 The XSGS Group Signature Scheme

We now focus on the XSGS protocol, introduced by Delerablée and Pointcheval in [14]. For security reasons (see Section 8.1 of the extended version of [3] for more details), we use the XSGS scheme without the XDH assumption. We thus use the double linear encryption scheme, introduced by Boneh et al. in [3] and described in Appendix A.2, instead of a double ElGamal encryption, as suggested in [14]. The group signature scheme is described by the following procedures, where λ is a security parameter.

- GenParam(1^λ): this procedure generates the public parameters of the system and also the keys of the different entities as follows:
  - a bilinear environment $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi)$;
  - the parameters for the double linear encryption, i.e. a generator $G \in_R \mathbb{G}_1$ and another generator $G' = \mathsf{rsk}.G$ where $\mathsf{rsk} \in_R \mathbb{Z}_q$.
  - the secret keys of the opening judge $(\mathsf{rsk}_1, \mathsf{rsk}_3) \in_R \mathbb{Z}_q^2$, $\mathsf{rsk}_2 = \mathsf{rsk}_1/\mathsf{rsk}$, $\mathsf{rsk}_4 = \mathsf{rsk}_3/\mathsf{rsk}$ and the associated public keys $\mathsf{Rpk}_1 = \mathsf{rsk}_1.G = \mathsf{rsk}_2.G'$ and $\mathsf{Rpk}_2 = \mathsf{rsk}_3.G = \mathsf{rsk}_4.G'$;

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

- the secret key $\mathsf{gmsk} \in_R \mathbb{Z}_q$ of the group manager $\mathcal{GM}$ and the associated public key $\mathsf{GMpk} = \mathsf{gmsk}.G_2$;
- the parameters of Paillier's encryption scheme [32] for ExtCommit;

The public parameters of the system are $\mathcal{PP} = \{\lambda, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi, G_1, G_2, G, G', \mathsf{GMpk}, \mathsf{Rpk}_1, \mathsf{Rpk}_2\}$.

– UserKeyGen(1^λ): before that a user, denoted U_i, can join a group, he has to be registered in a PKI. This procedure permits to ensure the unlinkability and the non-repudiation of the system. At the end of this procedure, the user obtain a couple of key $(\mathsf{usk}_i, \mathsf{Upk}_i)$. The value Upk_i is added in a table UPK which is supposed public.

– Join$[\mathcal{U}_i(\mathsf{usk}_i, \mathsf{Upk}_i) \leftrightarrow \mathcal{GM}(\mathsf{UPK}, \mathsf{gmsk})]$: this interactive protocol between a user U_i and the group manager results by the adhesion of the new user to the group. Consequently, the user obtain a group certificate $\mathsf{cert}_i = (A_i, x_i)$, and his group secret key gsk_i. The group manager add an entry $(\mathsf{Upk}_i, A_i, x_i, S)$ in Tab, where S is a signature of A_i made by the user U_i with his secret key usk_i. This interactive protocol is presented in Figure 2 where ExtCommit is an extractable commitment done with the Paillier's encryption scheme [32].

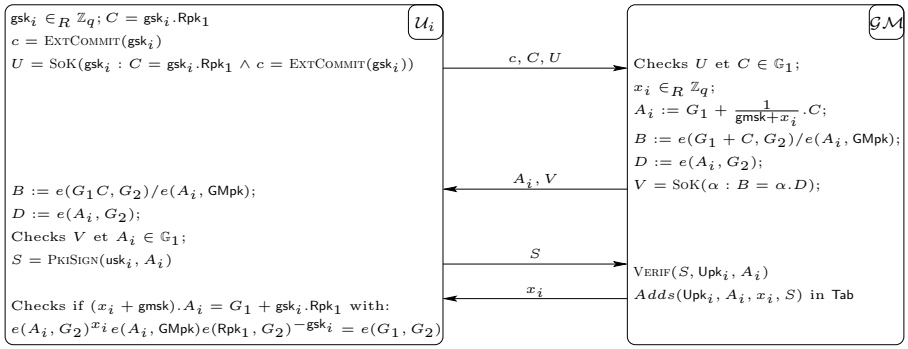

Fig. 2. XSGS Join protocol

– Sign$(m, \mathsf{gsk}_i, \mathsf{cert}_i)$: the signature of m is composed of two steps
- a double linear encryption, namely, the user randomly chooses $(\alpha_1, \beta_1, \alpha_2, \beta_2) \in_R \mathbb{Z}_q$ and computes the fours values

$$T_1 = \alpha_1.G; \quad T_2 = \beta_1.G'; \quad T_3 = A + (\alpha_1 + \beta_1)\mathsf{Rpk}_1;$$
$$T_4 = \alpha_2.G; \quad T_5 = \beta_2.G'; \quad T_6 = A + (\alpha_2 + \beta_2)\mathsf{Rpk}_2;$$

- a signature of knowledge U, where $z = (\alpha_1 + \beta_1).x + \mathsf{gsk}_i$:

$$U = \text{SoK}\Big(\alpha_1, \beta_1, \alpha_2, \beta_2, x, z : T_1 = \alpha_1.G \wedge T_2 = \beta_1.G' \wedge T_4 = \alpha_2.G$$
$$\wedge T_5 = \beta_2.G' \wedge T_3 - T_6 = (\alpha_1 + \beta_1)\mathsf{Rpk}_1 - (\alpha_2 + \beta_2)\mathsf{Rpk}_2 \wedge$$
$$e(T_3, G_2)^x e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(\alpha_1+\beta_1)} e(\mathsf{Rpk}_1, G_2)^{-z} = \frac{e(G_1, G_2)}{e(T_3, \mathsf{GMpk})}\Big)(m).$$

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.

The user outputs the signature $\sigma = (T_1, T_2, T_3, T_4, T_5, T_6, U)$.

- $\textsc{Verif}(m, \sigma)$ this procedure simply verifies the correctness of the signature of knowledge U, as detailed in Section A.4.
- $\textsc{Open}(m, \sigma, (\mathsf{rsk}_1, \mathsf{rsk}_2, \mathsf{rsk}_3, \mathsf{rsk}_4), \mathsf{Tab})$ if σ is valid, the opening judge computes $A = T_3 - (\mathsf{rsk}_1.T_1 + \mathsf{rsk}_2.T_2)$ and realizes the signature of knowledge $\tau = \textsc{SoK}\big(\mathsf{rsk}_1, \mathsf{rsk}_2 : A = T_3 - (\mathsf{rsk}_1.T_1 + \mathsf{rsk}_2.T_2) \wedge \mathsf{Rpk}_1 = \mathsf{rsk}_1.G \wedge \mathsf{Rpk}_1 = \mathsf{rsk}_2.G'\big)$. By using Tab, the judge can retrieve the key Upk_i associated to the user certificate A. Then he outputs Upk, $S(= \textsc{PkiSign}_{\mathsf{usk}}(A))$, A and τ.
- $\textsc{Judge}(m, \sigma, A, \tau, \mathsf{Upk}, \mathsf{Tab})$ this procedure verifies the correctness of the signature of knowledge τ. The signature S, stored in Tab, permits to check that the certificate A is the one which have been given to the user during the $\textsc{Join}$ procedure. If both signatures (τ and S) are valid, the procedure outputs 1 else it outputs 0.

This protocol ensures all the security requirements of a group signature scheme under the q-SDH [4] and the decision linear assumption (see Section A.2). We refer the interested reader to [14] and [3] for the security aspects of this scheme.

A.4 Focus on the Signature of Knowledge

During the signature of a message, a user must produce the signature of knowledge U. We detailed here how this should be done.

- Choose $r_{\alpha_1}, r_{\beta_1}, r_{\alpha_2}, r_{\beta_2} \in_R \mathbb{Z}_p$; $r_x, r_z \in_R \mathbb{Z}_q$
- Compute

$$\begin{aligned}
&P_1 = r_{\alpha_1}.G; P_2 = r_{\beta_1}.G'; P_3 = r_{\alpha_2}.G; P_4 = r_{\beta_2}.G';\\
&P_5 = (r_{\alpha_1} + r_{\beta_1})\mathsf{Rpk}_1 - (r_{\alpha_2} + r_{\beta_2})\mathsf{Rpk}_2;\\
&P_6 = e(T_3, G_2)^{r_x} e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(r_{\alpha_1}+r_{\beta_1})} e(\mathsf{Rpk}_1, G_2)^{-r_z}
\end{aligned}$$

- Compute $c = \mathcal{H}(m, T_1, T_2, T_3, T_4, T_5, T_6, P_1, P_2, P_3, P_4, P_5, P_6)$
- Compute

$$\begin{aligned}
&s_{\alpha_1} = r_{\alpha_1} + c.\alpha_1 \pmod q; s_{\beta_1} = r_{\beta_1} + c.\beta_1 \pmod q;\\
&s_{\alpha_2} = r_{\alpha_2} + c.\alpha_2 \pmod q; s_{\beta_2} = r_{\beta_2} + c.\beta_2 \pmod q;\\
&s_x = r_x + c.x \pmod q; s_z = r_z + c.z \pmod q.
\end{aligned}$$

The signature is the tuple $U = (c,\ s_{\alpha_1},\ s_{\beta_1},\ s_{\alpha_2},\ s_{\beta_2},\ s_x,\ s_z)$.

The verification of this signature of knowledge is done as follow. The verifier first computes:

- $P_1 = s_{\alpha_1}.G - c.T_1$, $P_2 = s_{\beta_1}.G' - c.T_2$, $P_3 = s_{\alpha_2}.G - c.T_4$, $P_4 = s_{\beta_2}.G' - c.T_5$ and $P_5 = (s_{\alpha_1} + s_{\beta_1})\mathsf{Rpk}_1 - (s_{\alpha_2} + s_{\beta_2})\mathsf{Rpk}_2 - c.(T_3 - T_6)$
- $P_6 = e(T_3, G_2)^{s_x} e(\mathsf{Rpk}_1, \mathsf{GMpk})^{-(s_{\alpha_1}+s_{\beta_1})} e(\mathsf{Rpk}_1, G_2)^{-s_z} \left(\frac{e(G_1,G_2)}{e(T_3,\mathsf{GMpk})}\right)^{-c}$

Finally the verifier validates the signature of knowledge if:

$$c = \mathcal{H}(m, T_1, T_2, T_3, T_4, T_5, T_6, P_1, P_2, P_3, P_4, P_5, P_6).$$

Appeared in K. H. Rhee and D. Nyang (Eds.): ICISC 2010, volume 6829 of LNCS, pp. 133-150, 2010.