An extended abstract of this paper appeared in
Progress in Cryptology - VIETCRYPT 2006 (25 – 28 september 2006, Hanoi, Vietnam)
P. Nguyen Ed. Springer-Verlag, LNCS 4341, pages 193–210.

Dynamic Fully Anonymous Short Group Signatures

Cécile Delerablée[1] and David Pointcheval[2]

[1] France Telecom Division R&D, Issy-les-Moulineaux, France
cecile.delerablee@orange-ftgroup.com
[2] CNRS-ENS, Paris, France
david.pointcheval@ens.fr

Abstract. Group signatures allow members to sign on behalf of a group. Recently, several schemes have been proposed, in order to provide more *efficient* and *shorter* group signatures. However, this should be performed achieving a strong security level. To this aim, a formal security model has been proposed by Bellare, Shi and Zang, including both dynamic groups and concurrent join. Unfortunately, very few schemes satisfy all the requirements, and namely the shortest ones needed to weaken the anonymity notion.
We present an extremely short dynamic group signature scheme, with concurrent join, provably secure in this model. It achieves stronger security notions than BBS, and namely the full anonymity, while still shorter. The proofs hold under the q-SDH and the XDH assumptions, in the random oracle model.

1 Introduction

Group signature schemes (thereafter denoted GSS) have been introduced by Chaum and van Heyst [12], in order to provide revocable anonymity to the signer, who is allowed to sign on behalf of a group. In such a scheme, an authority is able, in exceptional cases, to "open" any group signature, and thus recover the actual signer. Properties of group signature schemes make them very important cryptographic tools, with lots of applications (voting, bidding, *anonymous attestation*).

For many years, several GSS have been introduced, and namely the famous ACJT [1], which was the first provably secure coalition-resistant scheme, under the Strong RSA and DDH assumptions. More recently, Boneh, Boyen and Shacham (BBS) [6], and Camenisch and Lysyanskaya [11], proposed very efficient group signature schemes using bilinear maps. The former provides very short group signatures. Independently, Nguyen and Safavi-Naini (NS) [18] also proposed another group signature scheme using bilinear maps. Note that all these schemes were analyzed in the random oracle model [3].

Bellare, Micciancio and Warinschi (BMW) [2] gave formal definitions of the security properties of group signatures, and proposed the first scheme provably secure in the standard model (while totally unpractical). Independently, Kiayias and Yung [15] (and later [16]), also defined a security model. Bellare, Shi and Zhang (BSZ) [4] extended the BMW model to the case of dynamic groups. Unforgeability and anonymity are indeed crucial security notions, but they should be guaranteed even if the adversary is allowed to play various attack games: adaptively open signatures, join any user of his choice (dynamic group [4]), possibly concurrently (concurrent join [16]).

However, in several schemes, this model has been "weakened", to obtain better efficiency, or to fit with the actually achieved security notions, as done in BBS with CPA-full-anonymity, a weaker version of anonymity where the adversary is not allowed to open signatures when trying to break the anonymity notion. Very recently, Boyen and Waters [8] proposed the first *efficient* GSS that is provably secure without random oracles, but with an important loss of efficiency. Indeed, the length of group signatures grows according to the number of users, and the group public key too.

1.1 Motivations and Related Work

Recently, several schemes have been proposed, in order to reduce the computational cost and the size of group signatures. In particular, BBS [6] is the most efficient one, and provides the shortest

signatures so far. But they are still quite large if one compares to short classical signatures [7], and very short group signatures would be of great interest too.

Furthermore, the security level provided by BBS signatures does not fit in the security models proposed by Bellare et al. [2, 4]. Namely, *anonymity* is no longer formally guaranteed as soon as one signature is open. However, such an opening process is expected to happen, hence the importance of anonymity as defined in [2]: it must be guaranteed, even if the adversary can see/ask for several openings. Moreover, *non-frameability*, as defined in BSZ is not guaranteed, because the group manager is able to sign on behalf of any group member. However, the authors suggest a possible way to fix this security problem, what we exploited, as explained below. In NS [18], the (full) anonymity is guaranteed, but the computational cost and the size of the group signatures are larger, compared to BBS. Furthermore, while NS claims to be in the BSZ security model, an adaptive access to the join oracle is not properly dealt in the security proofs, and namely for the traceability.

Adaptive, together with concurrent join is specifically considered by Kiayias and Yung [16]. It is indeed a very attractive property since it allows for several users to register at the same time, which could not be avoided (without a drastic efficiency reduction) in many applications (Internet-based for example) However, their scheme provides quite long signatures, with quite high computational cost.

A weakness in the BSZ model is the lack of revocation procedure. They gave some reasons for that, however, revocation of group members is usually a major issue in practice, one has to deal with for an actual scheme.

1.2 Contribution

In this paper, we deal with all the above problems together (therefore in the full BSZ security model, and we even address revocation in the appendix). We thus present a new GSS, which provides the strongest security level (under by now classical computational assumptions) in the random oracle model, with quite practical features: concurrent join, very efficient signing and verification procedures, and eXtremely Short (XS) signatures. The short size is also due to an original application of the Forking Lemma [20], which is of independent interest.

Our signature scheme, named XSGS (eXtremely Short Group Signatures), provides anonymity (which is a better security level than BBS [6], and most of the other schemes, except NS [18] and KY [16]), with still very short signatures: 1444 bits, that is almost 70% shorter than a NS-Signature. Furthermore, it is more efficient than NS [18], which provides the same security level (and even better than BBS for verification.) Concurrent join and revocation are possible, which make our schemes very attractive for dynamic groups.

2 Group Signature Schemes

According to the BSZ security model [4], group signature schemes involve distinct authorities, with various rights, and should satisfy several security notions. Even if in many GSS, there is a single authority, holding both *Issuing* and *Opening* capacities, it is preferable to separate those two capabilities. One reason is the fact that for security proofs, we can consider one of the authority corrupted, or partially corrupted, and the other not corrupted (we detail this point later).

2.1 Entities

A group signature scheme involves several entities: the **group manager** $\mathcal{GM}$, which can add new members to the group, by issuing new certificates (we could extend the model to include the

revocation of certificates, but here we only consider the group manager as a certificate issuer. In the appendix B we briefly deal with the revocation process); the **opening manager** $\mathcal{OM}$, which can revoke the anonymity of any group signature; **users $\mathcal{U}$'s** which are group members; and **outsiders**, which do not belong to the group, but just have access to the group public key.

2.2 PKI environment

We assume that each user $\mathcal{U}_i$, before joining the group, obtains a personal secret key $\mathsf{usk}[i]$, associated to a personal certified public key $\mathsf{upk}[i]$ (in a PKI). The group manager will also have a certified pair of keys $(\mathsf{gmpk}, \mathsf{gmsk})$. This PKI environment is separated from the group environment, and thus the certification authority will be assumed fully trusted (the only one). Indeed, this PKI will provide the non-repudiation, but also the non-frameability property: even if the group authorities are corrupted, they cannot frame a group member. Such a PKI can be formalized by a *user-key generation* algorithm which generates a personal public and private key pair $(\mathsf{upk}[i], \mathsf{usk}[i])$ for a user $\mathcal{U}_i$.

2.3 Algorithms

In this section we recall the definitions regarding group signature algorithms, according to [4]:

- GKg– the key generation algorithm GKg generates, according to a security parameter, the group manager's secret key ik, the opening manager's secret key ok, and the group public key gpk.
- Join– running the *join* or issue algorithm $\mathsf{Join}(\mathcal{U}_i, \mathcal{GM})$, the group manager provides the member $\mathcal{U}_i$ with his secret key $\mathsf{gsk}[i]$. The group manager makes an entry $\mathsf{reg}[i]$ in the registration table reg, with the entire transcript of the process (unless explicitly stated).
- GSig– the *group signing* algorithm $\mathsf{GSig}(\mathsf{gsk}[i], m)$ generates a signature σ on a message m, in the name of the group, using the user's secret key $\mathsf{gsk}[i]$.
- GVf– the *group signature verification* algorithm $\mathsf{GVf}(\mathsf{gpk}, m, \sigma)$: takes as input the group public key, a group signature σ on message m. It decides whether the signature has been generated by a member of the group (this is a deterministic algorithm).
- Open– the *opening* algorithm $\mathsf{Open}(\mathsf{ok}, m, \sigma)$ revokes the anonymity of a signature, granted the opening manager's secret key. More precisely, with read-access to the registration table reg, the opening manager is able to recover the identity of the actual signer (this is a deterministic algorithm). The algorithm outputs an identity $\mathcal{U}_i$, and a proof τ of this claim (which will be used by the Judge algorithm).
- Judge– the *judge* algorithm takes as input the group public key gpk, the public key $\mathsf{upk}[i]$ of the user $\mathcal{U}_i$, a message m, a valid signature σ of m, and a proof τ. It is used to check that $\mathcal{U}_i$ produced the signature σ on the message m. This algorithm does not require any private information, and thus, the verification of the opening process is public.

2.4 Security Notions

The Oracles. In [4], the correctness and security definitions are formulated via experiments, which involve oracle access to the adversary. We briefly describe the oracles provided to adversaries in the security experiments.

- $\mathsf{AddU}(\cdot)$ – *add user* oracle, which on input an identity $\mathcal{U}_i$ of a new user runs the Join algorithm. This user $\mathcal{U}_i$ is added to the list HU of the Honest Users;
- $\mathsf{CrptU}(\cdot, \cdot)$ – *corrupt user* oracle, which on input an identity $\mathcal{U}_i$ of a new user and a string upk sets upk as the public key $\mathsf{upk}[i]$ of $\mathcal{U}_i$. This user is added to the list CU of the Corrupted Users;

- $\mathsf{SndToI}(\cdot,\cdot)$ – *send to issuer (group manager)* oracle, which allows a corrupted user to run the Join algorithm with the issuer;
- $\mathsf{SndToU}(\cdot,\cdot)$ – *send to user* oracle, which allows a corrupted group manager to run the Join algorithm with an honest user (which is important for the non-frameability);
- $\mathsf{USK}(\cdot)$ – *user secret key* oracle, which converts an honest user (in HU) into a corrupted user (in CU) by leaking the private keys $\mathsf{upk}[i]$ and $\mathsf{gsk}[i]$;
- $\mathsf{RReg}(\cdot)$ – *read registration table* oracle, which gives a read access to the registration table reg;
- $\mathsf{WReg}(\cdot,\cdot)$ – *write registration table* oracle, which gives a write access to the registration table reg;
- $\mathsf{GSig}(\cdot,\cdot)$ – *signing* oracle, which on input the identity i of an honest user and a message outputs the signature the user would produce;
- $\mathsf{Ch}_b(\cdot,\cdot,\cdot)$ – *challenge* oracle, which on input the identities $\mathcal{U}_{i_0}$ and $\mathcal{U}_{i_1}$ of two honest users and a message m, outputs the signature the user $\mathcal{U}_{i_b}$ would produce on m. The message–signature pair generated by this oracle is appended to the list Gset (initially set to empty);
- $\mathsf{Open}(\cdot,\cdot)$ – *opening* oracle, which on input a message–signature pair, not generated by the challenge oracle, and thus not in Gset, runs the Open algorithm to get the identity of the actual signer.

Security Notions. We review, in appendix A, the formal experiments [4] which model the security notions of a dynamic group signature scheme $\mathcal{GSS}$:

Correctness. Signatures generated by a honest member should be accepted, and the open algorithm should correctly identify the signer (and the judge should accept the proof returned by the opening algorithm).

Anonymity. Given signatures produced by a user (among two of his choice – left-or-right) the adversary should not be able to have a significant advantage in guessing which users (the left or the right) provided the signatures. The adversary has a full and adaptive access to the Open oracle, except on the signatures produced by the left-or-right signing oracle.

Traceability. It must be impossible to produce a valid signature such that either the honest opener is unable to identify the signer, or the opener believes it has identified the origin but is unable to produce a correct proof of its claim.

Non-frameability. Even the authorities (group manager and opener) are not able to wrongly accuse someone for having signed a message. For this security level, we assume a colluding subset of users and both authorities to be corrupted.

3 Preliminaries

Since our schemes use classical assumptions and notations, let us introduce them, and review the most famous pairing-based group signature scheme, proposed by Boneh, Boyen, and Shacham [6].

3.1 Computational Assumptions

All the protocols below will apply in three isomorphic cyclic groups of prime order p: $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$. We furthermore assume that there exists an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, which can be evaluated efficiently. We denote by ψ the isomorphism from $\mathbb{G}_2$ onto $\mathbb{G}_1$, that we assume to be one-way (easy to compute, but hard to invert).

The Decisional Diffie-Hellman Problem (DDH).

Definition 1. Let us consider any group $\mathbb{G}$ of prime order p, the decisional Diffie-Hellman problem is defined as follows: given a random generator $G \in \mathbb{G}$, two random elements aG, bG in $\mathbb{G}$, and a candidate $X \in \mathbb{G}$, one has to decide whether $X = abG$ or not.

We denote by $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\mathcal{A})$ the advantage of any adversary $\mathcal{A}$ in distinguishing the two distributions: (G, aG, bG, abG) and (G, aG, bG, cG). As usual, we also denote by $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$ the maximal advantage that any adversary can get within time t.

In our context, because of the efficient bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and the isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, the DDH problem is easy in $\mathbb{G}_2$: given a tuple $(G, aG, bG, cG) \in \mathbb{G}_2^4$, one simply checks whether $e(\psi(aG), bG) = e(\psi(G), cG)$.

The eXternal Diffie-Hellman Assumption (XDH). Note that we furthermore assumed this isomorphism to be one-way. This gives the chance for the following XDH assumption to be true. Such an assumption has been introduced by Camenisch, Hohenberger and Lysyanskaya in the full version of [9], and suggested in the full version of [6].

Definition 2. Given three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, as well as a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, while the DDH problem is easy in $\mathbb{G}_2$, the XDH assumption states that the DDH problem is hard in $\mathbb{G}_1$.

Note that the above assumption does not only imply the one-wayness of ψ, but also that there is not efficiently computable isomorphism from $\mathbb{G}_1$ onto $\mathbb{G}_2$. For supersingular curves, such an assumption is known to be false [14], however, it is conjectured to hold, using the Weil or Tate pairing on MNT curves (choosing curves with embedded degree > 1 and $\mathbb{G}_1$ to be the points defined over the ground field. In this case, one can use the Trace map to go from $\mathbb{G}_2$ to $\mathbb{G}_1$). This is reason why it has already been used in recent works [9], and the full version of [6].

The Strong Diffie-Hellman Assumption (SDH). A new assumption, similar to the Strong-RSA one, has been recently introduced by Boneh and Boyen [5]: the Strong Diffie-Hellman Assumption.

Definition 3. Let us be given two isomorphic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (together with the isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$.) The q-Strong Diffie-Hellman problem consists, on input a $(q+2)$-tuple $(G_1, G_2, \gamma G_2, \gamma^2 G_2, \ldots, \gamma^q G_2)$, for a random element $\gamma \in \mathbb{Z}_p$ and a random generator G_2 of $\mathbb{G}_2$, and $G_1 = \psi(G_2)$, in outputting a pair $\left(x, \frac{1}{\gamma+x} G_1\right)$, with $x \in \mathbb{Z}_p$.

We denote by $\mathsf{Succ}^{\mathsf{sdh}}_{(\mathbb{G}_1,\mathbb{G}_2)}(q, \mathcal{A})$ the success of any adversary $\mathcal{A}$ in outputting such a solution on a random input instance. We also denote by $\mathsf{Succ}^{\mathsf{sdh}}_{(\mathbb{G}_1,\mathbb{G}_2)}(q, t)$ the maximal success that any adversary can get within time t.

Definition 4. The q-SDH assumption states that this problem is intractable for a given q.

3.2 Common Parameters

One chooses a random generator G_2 in $\mathbb{G}_2$, and we denote by G_1 its transformation by ψ: therefore, $G_1 = \psi(G_2)$ is a generator of $\mathbb{G}_1$. We also need additional, and independent generators G, H and K in $\mathbb{G}_1$, whose relative discrete logarithms as well as discrete logarithms in basis G_1 are unknown (unless something else is made more precise). We will denote by $W = \gamma G_2$ the public key of the group (with all the above public informations: the groups and the generators). The value $\gamma \in \mathbb{Z}_p$ is kept secret by the group manager. It will be used to issue membership certificates.

3.3 BBS: Short Group Signatures

The idea of the BBS group signature [6] consists in providing a signature of knowledge of a solution to the SDH problem: (A, x) such that $(x + \gamma)A = G_1$. The latter is generated granted the help of the group manager who knows γ. However, in order to allow the anonymity revocation (the opening operation by the group manager), the proof must not be totally zero-knowledge but partially only: the group manager should be able to recover A.

Therefore, in order to sign a message m, the user first encrypts A with the encryption key of the group manager; he then provides a zero-knowledge proof that the plaintext actually contains an A for which he knows the corresponding x. The security analysis didnot follow the above BSZ model [4], because of some restrictions:

- the unforgeability of the certificates directly comes from the q-SDH assumption. However, the proposed format of the membership certificate does not make any value private to the group manager. Therefore, he can sign on behalf of any user: the non-frameability cannot be guaranteed.
- with a semantically secure encryption scheme, anonymity is guaranteed. However, since one works in groups subject to efficiently computable bilinear maps, the DDH problem may not be hard. Therefore, they prefer to use a new encryption scheme (linear encryption) instead of the classical ElGamal encryption (the ciphertext is larger: 3 group elements, instead of 2). Furthermore, it is semantically secure against chosen-*plaintext* attacks only: the semantic security (and even the one-wayness) can be broken if the adversary has access to the decryption oracle: the above definition of anonymity does not hold if the adversary has access to the Open-oracle. This is the reason why they defined a weaker notion of anonymity, the so-called CPA-full-anonymity, or weak anonymity.

On the other hand, the main goal was a *short* signature, which indeed consists of three elements of $\mathbb{G}_1$ (the encryption) and six elements of $\mathbb{Z}_p$ (the proof of knowledge). Hence, the size is just 1533 bits.

3.4 Improvements

In order to improve the security (anonymity and non-frameability), it seems natural that we have to enhance the scheme, and thus to degrade the size:

- make the encryption scheme IND-CCA2 [21], by adding a proof to ensure security against chosen-ciphertext attacks;
- involve an extra parameter in the membership certificate, known to the user only.

Actually, it is possible to make these security improvements without loosing anything from the efficiency point of view (and even improving it too), making the XDH assumption.

4 XS Group Signatures

Note that for simplicity, we will use "certificate" to designate $(A_i, \mathsf{gsk}[i])$ in general, and just A_i at some specific time. In our scheme, we exploit and study two suggestions from [6], also used in [18], together with new tricks:

- First, we make the assumption that the DDH holds in the group $\mathbb{G}_1$, which is true under the XDH assumption. This will then allow a compact IND-CCA2 ElGamal-based encryption scheme;

– Then, we enhance the membership certificate with an additional secret y, known to the user only: (A, x, y), with $A \in \mathbb{G}_1$, $x, y \in \mathbb{Z}_p$, such that $(x + \gamma)A = G_1 + yH$. Applying $e(\cdot, G_2)$ on both sides, one gets that a triple (A, x, y) is a valid certificate if and only it satisfies the relation:

$$e(A, G_2)^x \cdot e(A, W) \cdot e(H, G_2)^{-y} = e(G_1, G_2).$$

– Finally, we revisit the forking lemma [20] in order to even shorten the signatures.

4.1 Concurrent Join Protocol and Revocation

In order to guarantee the non-frameability, one needs a specific Join procedure which provides a group member with a certificate such that the group manager does not know the private key. During the Join protocol, a future group member interacts with the group manager, in order to obtain a valid group certificate (A, x, y), with a private y. This Join protocol is presented on figure 1, where

$\mathcal{U}$ $(\mathsf{upk}, \mathsf{usk})$		$\mathcal{GM}$ (γ, gmsk)
$y \xleftarrow{R} \mathbb{Z}_p$, $C \leftarrow yH$		
$U \leftarrow \begin{pmatrix} c = \mathsf{Ext\text{-}Commit}(y), \\ \mathsf{NIZKPEqDL}(c, C, H) \end{pmatrix}$	$\xrightarrow{C,U}$	Verifies $C \in \mathbb{G}_1$, checks U
		$x \xleftarrow{R} \mathbb{Z}_p$, $A \leftarrow (\frac{1}{\gamma+x})(G_1 + C)$
		$B \leftarrow e(G_1 + C, G_2)/e(A, W)$
		$D \leftarrow e(A, G_2)$
	$\xleftarrow{A,V}$	$V \leftarrow \mathsf{NIZKPoKDL}(B, D)$
$B \leftarrow e(G_1 + C, G_2)/e(A, W)$		
$D \leftarrow e(A, G_2)$		
Verifies $A \in \mathbb{G}_1$, checks V		
$S \leftarrow Sign_{\mathsf{usk}}(A)$	$\xrightarrow{S}$	Checks S w.r.t. upk and A
Checks that $(x + \gamma)A \stackrel{?}{=} G_1 + yH$	$\xleftarrow{x}$	adds (upk, A, x, S)
i.e. $e(A, G_2)^x \cdot e(A, W) \cdot e(H, G_2)^{-y} = e(G_1, G_2)$		

Fig. 1. Join Protocol

– Ext-Commit is an extractable commitment, that is a commitment which is perfectly binding, and computationally hidding, and a trapdoor allows to open it. Actually, the trapdoor will not be known to anybody, except to our simulator in the security proofs of the traceability and non-frameability. A good example, well-suited to our situation, is the Paillier's encryption scheme [19]: as any encryption scheme, injectivity implies the unconditional binding property, while the computational hiding relies on the semantic security, the high-residuosity assumption. The decryption key allows the extraction;
– $\mathsf{NIZKPEqDL}(c, C, H)$ denotes a zero-knowledge proof of equality of the discrete logarithm of C in basis H with the committed value in c, non-interactive in the random oracle model. We won't detail such a proof, but it can be efficiently done with the Paillier's encryption scheme, since it is an equality of discrete logarithms (in different groups). Note that such a proof of membership together with an extractable commitment becomes a proof of knowledge: the user necessarily built C knowing y.
– $\mathsf{NIZKPoKDL}(B, D)$ denotes a zero-knowledge proof of knowledge of the discrete logarithm of B in basis D, non-interactive in the random oracle model.

Let us explain the steps in this protocol: This protocol is concurrently secure since all the proofs are non-interactive (NIZKPEqDL, NIZKPoKDL, and the signature), and everything is defined in the first move (the 2 first flows), while the second move (the 2 last flows) involves a signature before revealing the certificate to the user. It ensures the non-frameability property. Indeed, the signature $Sign_{\mathsf{usk}}(A)$ ensures that $\mathcal{U}$ owns the certificate A, in a non-repudiable way. But such a signature is provided by $\mathcal{U}$ only after having checked V: $\mathcal{GM}$ actually knows x, and thus used the C chosen by $\mathcal{U}$. Therefore, he cannot know the associated y.

The *revocation*, which allows the group manager to remove a member from the group, works almost exactly as in [6] (inspired by [10]). We describe it in more details in appendix B.

4.2 XSGS: an eXtremely Short Group Signature Scheme

Since we make the XDH assumption, it is reasonable to apply a classical ElGamal encryption, to hide the certificate, in a revocable way. In order to reach the (full) anonymity property, we enhance the encryption scheme with the IND-CCA2 security, using the Naor-Yung methodology [17], but in the random-oracle model [13]. In order not to increase too much the size of the signature, the above H (involved in the certificate) will be used as one of the opening manager's public keys. Actually, the secret key of the opening manager is the pair (ξ_1, ξ_2) such that $H = \xi_1 K$ and $G = \xi_2 K$.

Parameters. We thus have:

- group public key: $\mathsf{gpk} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi; G_1, K, H = \xi_1 K, G = \xi_2 K; G_2, W = \gamma G_2)$;
- group manager's secret key: $\mathsf{ik} = \gamma$, which helps to generate the certificate triples (A, x, y), with $A \in \mathbb{G}_1$, $x, y \in \mathbb{Z}_p$, such that $(x + \gamma)A = G_1 + yH$;
- opening manager's secret key: $\mathsf{ok} = (\xi_1, \xi_2)$, which will help to decrypt ElGamal ciphertexts;
- an extractable commitment scheme. In the case of the Paillier's encryption scheme [19], one has to choose an RSA modulus n, and an element g of maximal order in $\mathbb{Z}^\star_{n^2}$, without knowing/keeping the factorization.

Double ElGamal Encryption. The signer who owns a certificate (A, x, y), randomly chooses $\alpha, \beta \in \mathbb{Z}_p$ and computes: $T_1 = \alpha K \quad T_2 = A + \alpha H \quad T_3 = \beta K \quad T_4 = A + \beta G$.

First, in order to make the encryption scheme resistant to chosen-ciphertext attacks, one has to prove that (T_1, T_2) and (T_3, T_4), which are two ciphertexts with independent keys and independent random coins, encrypt the same plaintext: there exist α and β such that

$$T_1 = \alpha K \qquad T_3 = \beta K \qquad T_2 - T_4 = \alpha H - \beta G.$$

Secondly, as before, (T_1, T_2) is the encryption of a valid certificate (A, x, y) if and only if there exists an α such that (with $z = x\alpha + y$)

$$T_1 = \alpha K \quad \text{and} \quad e(T_2, G_2)^x \cdot e(H, W)^{-\alpha} \cdot e(H, G_2)^{-z} = e(G_1, G_2)/e(T_2, W).$$

Signature. The signer has thus to prove the knowledge of (α, β, x, z) which satisfies the 4 above relations. Such a proof of knowledge clearly shows that, both there exist convenient α and β values, and the prover knows a certificate. It can be performed with classical techniques, and the non-interactive version uses the Fiat-Shamir paradigm, in the random-oracle model: in order to sign m, $\mathcal{U}$ randomly chooses 4 elements r_α, r_β, r_x and r_z in $\mathbb{Z}_p$ and computes

- $R_1 = r_\alpha K \qquad R_2 = e(T_2, G_2)^{r_x} \cdot e(H, W)^{-r_\alpha} \cdot e(H, G_2)^{-r_z}$
  $R_3 = r_\beta K \qquad R_4 = r_\alpha H - r_\beta G.$

- $c = \mathcal{H}(m, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$, where $\mathcal{H}$ outputs k-bit long elements;
- $s_\alpha = r_\alpha + c\alpha \bmod p \qquad s_\beta = r_\beta + c\beta \bmod p$
  $s_x = r_x + cx \bmod p \qquad s_z = r_z + cz \bmod p.$

A signature therefore consists of the tuple $(T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z)$, and the verifier finally checks whether the following relations are satisfied or not:

$$s_\alpha K = R_1 + cT_1 \qquad s_\beta K = R_3 + cT_3 \qquad s_\alpha H - s_\beta G = R_4 + c(T_2 - T_4)$$
$$e(T_2, G_2)^{s_x} \cdot e(H, W)^{-s_\alpha} \cdot e(H, G_2)^{-s_z} = R_2 \cdot (e(G_1, G_2)/e(T_2, W))^c$$

Open. To open a signature, $\mathcal{OM}$ uses the decryption key ok to recover A (and provides a publicly-verifiable proof τ that he did it well —which is a simple proof of equality of discrete logarithms in $\mathbb{G}_1$)—, and then the actual signer, using his read-access to the registration table reg (to prove, in τ, that the designated user $\mathcal{U}_i$ has not be framed, $\mathcal{OM}$ uses $S = Sign_{\mathsf{usk}[i]}(A)$).

4.3 Properties

Such a signature contains 4 elements from $\mathbb{G}_1$ (over 171 bits) and 4 scalars (modulo p) of 170 bits. The challenge can just be on 80 bits: the signature can thus be encoded on 1444 bits (less than 181 bytes). From the effiency point of view:

- for the signature, one can compute

  $$R_2 = e(A, G_2)^{r_x} \cdot e(H, W)^{-r_\alpha} \cdot e(H, G_2)^{\alpha r_x - r_z}.$$

  Since all the pairing values can be precomputed, the signature globally requires 7 multi-exponentiations in $\mathbb{G}_1$ and 1 multi-exponentiation in $\mathbb{G}_T$.
- to verify a signature one has to compute

  $$R_2 = e(T_2, s_x G_2 + cW) \cdot e(H, W)^{-s_\alpha} \cdot e(H, G_2)^{-s_z} \cdot e(G_1, G_2)^{-c}.$$

  Most of the pairing values can be precomputed: the verification requires 3 multi-exponentiations in $\mathbb{G}_1$, 1 multi-exponentiation in $\mathbb{G}_2$, 1 pairing computation and 1 multi-exponentiation in $\mathbb{G}_T$.

4.4 Without the XDH Assumption.

One should note that the XDH assumption helps to get the very short signature, but is not crucial for our construction: if the XDH assumption does not hold, one can use a double variant of the Linear Encryption (the Linear Encryption has been introduced in [6], and is secure assuming the Decision Linear Diffie-Hellman Assumption). Thus we can obtain group signatures of 2126 bits (6 elements from $\mathbb{G}_1$, 6 scalars (modulo p) of 170 bits, and the challenge).

4.5 Security Analysis of XSGS

In order to prove the **correctness** of the group signature scheme, we first need to show that the interactive proof of knowledge is complete, then, the correctness of the Open algorithm immediately leads to the result. Actually, in order to prove the **traceability**, we furthermore need to show that the interactive proof of knowledge is an honest-verifier zero-knowledge and sound proof of knowledge. Thereafter, a simple application of the forking lemma 7 leads to the expected result. This means that we first need the following lemma, which proof can be found in appendix C.

Lemma 5 (Honest-Verifier Zero-Knowledge Proof of Knowledge). *The interactive proof is a honest-verifier zero-knowledge proof of knowledge.*

From the correctness of the proof of knowledge and the use of a correct encryption scheme, one gets the correctness of the group signature scheme:

Theorem 6 (Correctness). *The group signature scheme* XSGS *is correct.*

If one gets a closer look at the proof of the forking lemma [20], with a random oracle $\mathcal{H}$ which outputs k-bit elements, one can claim the following lemma:

Lemma 7 (Forking Lemma). *Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data and which can ask q_H queries to the random oracle, with $q_H > 0$. We assume that, within the time bound T, $\mathcal{A}$ produces a valid signature $(m, \sigma_1, h, \sigma_2)$, with probability $\varepsilon \geq 1/2^k + \eta$ for some $\eta > 240q_H/2^k$. Then, within time $T' \leq 9q_H T/\varepsilon$, and with probability $\varepsilon' \geq \frac{1}{6}$, a replay of this machine outputs two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$.*

Proof. First, with probability greater than η, $\mathcal{A}$ outputs a signature $(m, \sigma_1, h, \sigma_2)$ that is valid, such that h has been obtained as an $\mathcal{H}$ answer on (m, σ_1). Therefore, if we run the attacker $2/\eta$ times with different random tapes, we get a success with probability greater than $1 - e^{-2} \geq \frac{6}{7}$, such that the query $\mathcal{H}(m, \sigma_1)$ has been asked, and answered by h: the crucial query.

By applying the Splitting-Lemma [20], we know that with probability of 1/4, for each replay, we have a new success with probability greater than $\eta/4q_H$: we thus replay the attack $8q_H/\eta$ times with a new random oracle (but the same answers until the crucial query). With probability greater than $\frac{6}{7}$, we get another success. The challenge is different from the previous one with probability $8q_H/\eta 2^k$.

Finally, after less than $2(1 + 4q_H)/\eta$ replays of the attack, with probability greater $1/5 - 8q_H/\eta 2^k$ which is greater than 1/6, as soon as $\eta \geq 240q_H/2^k$, we get two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m', \sigma_1', h', \sigma_2')$ with $h' \neq h$. □

The following lemma shows that the signature is unforgeable without the knowledge of a certificate, even if the hash function outputs 80-bit values:

Lemma 8 (Unforgeability). *It is computationally impossible to produce a valid signature, without the knowledge of a membership certificate, even under chosen-message attacks, in the random oracle model: if there exists an adversary $\mathcal{A}$ able to build a valid signature within time t, with probability $\varepsilon \leq 1/2^k + \eta + q_S(q_H + q_S)/p^4$, for some $\eta > 240q_H/2^k$, after q_H queries to the random oracle $\mathcal{H}$ and q_S queries to the signing oracle, then one can build a membership certificate in expecting time $\mathcal{O}(q_H t/\eta)$.*

Proof. We remind the signature consists of $((m, T_1, T_2, T_3, T_4), c, (s_\alpha, s_\beta, s_x, s_z))$, which is not exactly the framework used in the above Forking Lemma. Anyway, with a few extra computation (but no new hash-query), one can make such a signature of the more classical form $((m, T_1, T_2, T_3, T_4), (R_1, R_2, R_3, R_4), c, (s_\alpha, s_\beta, s_x, s_z))$, where $c = \mathcal{H}(m, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$.

Furthermore, one can efficiently simulate the signing algorithm, in the name of any user (with a statistically negligible probability of failure when setting a random oracle value: less than $(q_H + q_S)/p^4$ for each signature simulation.) If the adversary succeeds with probability $\varepsilon \leq 1/2^k + \eta + q_S(q_H + q_S)/p^4$, then including the signature simulation, we build a *no-message* adversary that makes a forgery with probability greater than $1/2^k + \eta$. According to the above forking lemma, one can extract two related signatures, with the same hash-query but different challenges

$$(m, T_1, T_2, T_3, T_4), \quad (R_1, R_2, R_3, R_4), \quad c, (s_\alpha, s_\beta, s_x, s_z) \quad c', (s'_\alpha, s'_\beta, s'_x, s'_z)$$

in expected time $\mathcal{O}(q_H t/\eta)$. Thereafter, simply applying the same technique as the one used to prove the soundness, one gets a valid certificate (A, x, y). □

Theorem 9 (Traceability). *The group signature scheme* XSGS *is traceable*[1]

Proof. Suppose there is an adversary $\mathcal{A}$ that wins with probability ε the traceability game against our scheme. We describe an algorithm $\mathcal{B}$ that can break the SDH problem, with the help of the adversary $\mathcal{A}$. Let $\{(A_i, x_i, y_i)\}_{i=1}^q$ be the set of certificates generated during the whole attack, by honest users, who can be (all) corrupted later by the adversary (using USK oracle).

If the adversary can generate a signature which opens to a new $A^\star$ (not associated to an existing user), using the "unforgeability" technique (see lemma 8), one can find a new certificate $(A^\star, x^\star, y^\star)$ in reasonable expected time. From the success of the adversary in the attack game, we know that $A^\star$ does not belong to $\{A_i\}_{i=1}^q$. Let $\mathcal{B}$ be given a q-SDH instance $(G, G', \Theta G', \ldots, \Theta^q G')$. It

- randomly chooses $\alpha \xleftarrow{R} \mathbb{Z}_p$ and $x_i \xleftarrow{R} \mathbb{Z}_p$, for $i = 1, \ldots, q$, such that the x_i's are pairwise distinct
- randomly chooses $y_i \xleftarrow{R} \mathbb{Z}_p$, for $i, \ldots, q$, and $k \xleftarrow{R} \{1, ..., q\}$
  (lets us formally define $\gamma \leftarrow \Theta - x_k$, which is unknown)
- computes from the challenge q-SDH instance (since all the formula involve polynomials in Θ of degree at most q times G' or G, and $G = \psi(G')$),

$$G_2 \leftarrow \alpha\left[\prod_{i=1}^q (\Theta + x_i - x_k)\right] G' - y_k \left[\prod_{\substack{i=1 \\ i\neq k}}^q (\Theta + x_i - x_k)\right] G'; \quad G_1 \leftarrow \psi(G_2)$$
$$H \leftarrow \left[\prod_{\substack{i=1 \\ i\neq k}}^q (\Theta + x_i - x_k)\right] G; \qquad\qquad W \leftarrow \gamma G_2$$

- randomly generates ok and compute the corresponding encryption keys
- generates the extractable commitment, but *knowing* the trapdoor,
- simulates the users $\{(A_i, x_i, y_i)\}_{i=1}^q$, computing $A_i = \frac{1}{x_i+\gamma}(G_1 + y_i H)$ according to i:
  - if $i = k$, $A_k = \dfrac{1}{x_k+\gamma}(\alpha\Theta H) \leftarrow \alpha \left[\displaystyle\prod_{i=1, i\neq k}^{q} (\Theta + x_i - x_k)\right] G$
  - if $i \neq k$, since

$$y_i = (y_i - y_k) + y_k \text{ and } (y_i - y_k)H = (y_i - y_k)\left[\prod_{j=1, j\neq k}^{q} (\Theta + x_i - x_k)\right] G,$$

$$A_i \leftarrow (y_i - y_k)\left[\prod_{j=1, j\neq i, k}^{q} (\Theta + x_j - x_k)\right] G + \alpha\left[\prod_{j=1, j\neq i}^{q} (\Theta + x_j - x_k)\right] G.$$

Finally, the certificate satisfies

$$A^\star = \frac{1}{x^\star+\gamma}(G_1 + y^\star H) = \frac{1}{x^\star + \Theta - x_k}(\alpha\Theta + y^\star - y_k)\left[\prod_{i=1, i\neq k}^{q} (\Theta + x_i - x_k)\right] G.$$

Since $A^\star \notin \{A_i\}_{i=1}^q$, and namely $A^\star \neq A_k$, $y^\star - y_k \neq \alpha(x^\star - x_k)$. When we extract $(x^\star, y^\star)$ (in reasonable expected time), two cases may happen:

[1] The proposed Join protocol does not guarantee traceability if the adversary is allowed to add new corrupted members. To allow that, the simulator should be able to choose the y_i of corrupted users, which is possible if the group manager sends back an additional $(y', C' = y'H)$ to be used by the user in addition to his own (y, C).

1. $x^\star \notin \{x_1, \ldots, x_q\}$ with probability greater than $1/2$. By an Euclidean division, one can express $A^\star = (C/(\Theta + x^\star - x_k) + P(\Theta))\, G$, with

$$C = (\alpha(x_k - x^\star) + y^\star - y_k) \left[\prod_{i=1, i\neq k}^{q} (x_i - x^\star) \right] \neq 0$$

   and P a polynomial of degree q. And thus, $\mathcal{B}$ can compute C and $P(\Theta)G$ from the initial instance, and therefore $(\frac{1}{x+\Theta}G, x)$, a solution to the q-SDH problem (with $x = x^\star - x_k$).
2. $x^\star \in \{x_1, \ldots, x_q\}$ with probability greater than $1/2$. Since no information leaks about k, $x^\star = x_k$ with probability $1/q$, and then

$$\frac{1}{y_k - y^\star}(y_j A^\star - y^\star A_j) = \frac{1}{\Theta} G_1 = \alpha \left[\prod_{\substack{i=1 \\ i\neq k}}^{q} (\Theta + x_i - x_k) \right] G - \frac{y_k}{\Theta} \left[\prod_{\substack{i=1 \\ i\neq k}}^{q} (\Theta + x_i - x_k) \right] G$$

   As above, it can be expressed as $(C/\Theta + P(\Theta)G$, where P is a polynomial of degree $q-1$. $\mathcal{B}$ has obtained $(\frac{1}{\Theta}G, 0)$, solution to the q-SDH problem (with $x = 0$).

□

The **anonymity** property (not only CPA-full-anonymity [6]) is achieved granted the Double El-Gamal encryption scheme, which is IND-CCA [13].

Theorem 10 (Anonymity). *Under the XDH assumption, the group signature scheme* XSGS *is anonymous: if there exists an adversary $\mathcal{A}$ able to break the anonymity game, with advantage ε, and within time t (in the random oracle model), after q_H queries to the random oracle $\mathcal{H}$ and q_S queries to the challenge oracle, then one can break the DDH problem in $\mathbb{G}_1$ with advantage $\varepsilon/4 - (q_H + q_S)/p^4$, within time $t' \leq t + 4q_S T_{\mathsf{pairing}} + (2 + 8q_S)T_{\mathsf{exp}}$, where T_{pairing} is the time of a pairing computation, and T_{exp} is the time of a (multi)-exponentiation.*

Proof. We are given a quadruple $(K, T, U = uK, V = vT)$ in $\mathbb{G}_1$ such that either $u = v$ (DDH quadruple) or v is random (random quadruple). From such a tuple, using the classical random self-reducibility, one can derive many independent tuples: $(K, T, U_i = u_i U + v_i K, V_i = u_i V + v_i K)$. We will choose the group manager's secret key γ, and compute $W = \gamma G_2$. Then, we flip a coin d, and define, either $H = \xi_1 K$ and $G = T$ (if $d = 0$), or $H = T$ and $G = \xi_2 K$ (if $d = 1$), to set the group public key as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi; G_1, K, H, G; G_2, W = \gamma G_2)$. Actually, we only know half of the opening manager's key. We will show it is enough to simulate the decryption process (while still allowing to extract something from the adversary). All the other queries can be perfectly answered since using γ, we can build certificates (Join-queries), and thus answer GSig-queries too. Hash-queries can be simulated as usual with a new random value in $\{0,1\}^k$ for any new query. For the challenge queries, the simulator $\mathcal{B}$ chooses two independent random bits b and d', and will try to simulate the signature from A_b: the i-th request is answered, given a message m, and two certificates (A_0, x_0, y_0) and (A_1, x_1, y_1)

- the encryption: according to d, by choosing an additional random bit d':
  - if $d = 0$, $T_1 \leftarrow \alpha K$, $T_2 \leftarrow A_{d'} + \alpha H$, $T_3 \leftarrow U_i$ and $T_4 \leftarrow A_b + V_i$, for a random α;
  - if $d = 1$, $T_1 \leftarrow U_i$, $T_2 \leftarrow A_b + V_i$, $T_3 \leftarrow \beta K$ and $T_4 \leftarrow A_{d'} + \beta G$, for a random β.
- the proof of validity can be simulated (see the simulator for the zero-knowledge property in appendix C). The latter may fail only with a negligible probability when setting the random oracle value, but less than $(q_H + q_S)/p^4$.

In case of failure, $\mathcal{B}$ exits, otherwise, $(T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z)$ is the signature of m given back to $\mathcal{A}$. Eventually, the latter returns its guess b' for b. Our algorithm $\mathcal{B}$ answers $\beta = (b' = b)$ as its guess about the tuple (K, T, U, V).

If this is a DDH tuple, and $d' = b$, the 2 ElGamal encryptions always encrypt A_b, this is a valid signature (the advantage of $\mathcal{A}$ in guessing b is ε.) However, if $d' \neq b$, both certificates are encrypted, $\mathcal{A}$ has thus no advantage in guessing b (we will indeed show below that decryption queries do not reveal any information about d.) If this is a random tuple, the signature is independent of b, thus the adversary's advantage is 0. As a consequence, our algorithm $\mathcal{B}$ has an advantage $\varepsilon/4$ in distinguishing DDH quadruples (in a group subject to bilinear maps, hence breaking XDH.)

Now, since we know half of the opening manager's key, as soon as a signature is valid (with identical plaintexts), the decryption of half of the ciphertext is enough. The soundness of the proof of validity (see in appendix C) showed that incorrect proofs are very unlikely. □

The final property is the **non-frameability**, which is important for honest users: it guarantees that neither the group manager or the opening manager can cheat, and frame him.

Theorem 11 (Non-Frameability). *The group signature scheme* XSGS *is non-frameable.*

Proof. First, it is clear, from the Open protocol is publicly verifiable, that a wrong open procedure is statistically negligible (even for a powerful adversary).

Second, suppose there is an adversary $\mathcal{A}$ that breaks the non-frameability of our scheme. We describe an algorithm $\mathcal{B}$ that can break the DL problem. Let $\{(A_i, x_i, y_i)\}_{i=1}^{q+q'}$ be the set of certificates generated during the attack. γ and all the (A_i, x_i) are given to the adversary (so the adversary has access to all $y_i H$), but only $\{y_i\}_{i=1}^{q}$, for the insider colluders. If the adversary can generate a signature which opens to a $A^\star \in \{A_i\}_{i=q+1}^{q+q'}$ (outside the collusion), using the "unforgeability" technique (replay attack and soundness), one can find the whole certificate $(A^\star, x^\star, y^\star)$, and two cases may happen:

Case 1: $A^\star \in \{A_i\}_{i=q+1}^{q+q'}$ and $(A^\star, x^\star) \notin \{(A_i, x_i)\}_{i=q+1}^{q+q'}$ more than half of the time. We show that given a discrete logarithm instance (G, G') in $\mathbb{G}_2$, $\mathcal{B}$ can compute $\Theta = \log_G G'$. In this case, $\mathcal{B}$ computes the group public key: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi; G_1 = \psi(G), K = \xi H, H = \psi(G'); G_2 = G, W = \gamma G)$ with $(\gamma, \xi) \overset{R}{\leftarrow} \mathbb{Z}_p^{\star 2}$, randomly chosen by $\mathcal{B}$. It can furthermore simulate any kind of join procedure using γ. We then have $H = \Theta G_1$. Since $A^\star = A_j \in \{A_i\}_{i=q+1}^{q+q'}$ (with $x^\star \neq x_j$, and thus $y^\star \neq y_j$), we have

$$\begin{aligned} A^\star &= \frac{1}{x^\star + \gamma}(G_1 + y^\star H) = v\frac{1}{x^\star + \gamma}(G_1 + y^\star \Theta G_1) \\ &= \frac{1}{x_j + \gamma}(G_1 + y_j H) = \frac{1}{x_j + \gamma}(G_1 + y_j \Theta G_1). \end{aligned}$$

Therefore, $(x_j + \gamma)(1 + y^\star\Theta) = (x^\star + \gamma)(1 + y_j\Theta)$ and $y^\star \neq y_j$, which easily leads to Θ.

Case 2: $(A^\star, x^\star) \in \{(A_i, x_i)\}_{i=q+1}^{q+q'}$ more than half of the time. We show that given a discrete logarithm instance (G, G') in $\mathbb{G}_1$, $\mathcal{B}$ can compute $\Theta = \log_G G'$. In this case, $\mathcal{B}$ computes the group public key: $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi; G_2 \overset{R}{\leftarrow} \mathbb{G}_1, K = \xi H, H = G; G_1 = \psi(G_2), W = \gamma G)$ with $(\gamma, \xi) \overset{R}{\leftarrow} \mathbb{Z}_p^{\star 2}$, randomly chosen by $\mathcal{B}$. It can simulate any join procedure as above, with γ, but also chooses a random $j \in \{q+1, ..., q+q'\}$, for which honest user it makes $A_j = \frac{1}{x_j+\gamma}(G_1 + G')$. Since

$(A^\star, x^\star) \in \{(A_i, x_i)\}_{i=q+1}^{q+q'}$, we have $(A^\star, x^\star) = (A_j, x_j)$ whith probability $1/q'$, and thus:

$$A^\star = \frac{1}{x^\star + \gamma}(G_1 + y^\star G) = A_j = \frac{1}{x_j + \gamma}(G_1 + G')$$
$$= \frac{1}{x_j + \gamma}(G_1 + \Theta G) = \frac{1}{x^\star + \gamma}(G_1 + \Theta G)$$

which easily leads to Θ ($\Theta = y^\star$.) □

Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments. This work has been done thanks to the French RNRT Crypto++ contract.

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *Crypto '00*, LNCS 1880, pages 255–270. Springer-Verlag, Berlin, 2000.
2. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *Eurocrypt '03*, LNCS 2656, pages 614–629. Springer-Verlag, Berlin, 2003.
3. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.
4. M. Bellare, H. Shi, and C. Zang. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT – RSA '05*, LNCS 3376, pages 136–153. Springer-Verlag, Berlin, 2005.
5. D. Boneh and X. Boyen. Short Signatures without Random Oracles. In *Eurocrypt '04*, LNCS 3027, pages 56–73. Springer-Verlag, Berlin, 2004.
6. D. Boneh, X. Boyen, and H.Shacham. Short Group Signatures. In *Crypto '04*, LNCS 3152, pages 41–55. Springer-Verlag, Berlin, 2004.
7. D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Asiacrypt '01*, LNCS 2248, pages 514–532. Springer-Verlag, Berlin, 2001.
8. D. Boneh and B. Waters. Compact group signatures without random oracles. In *Eurocrypt '06*, LNCS 4004, pages 427–444. Springer-Verlag, Berlin, 2006.
9. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-cash. In *Eurocrypt '05*, LNCS 3494, pages 302–321. Springer-Verlag, Berlin, 2005.
10. J. Camenisch and A. Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Crypto '02*, LNCS 2442, pages 61–67. Springer-Verlag, Berlin, 2002.
11. J. Camenisch and A. Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Crypto '04*, LNCS 3152, pages 52–72. Springer-Verlag, Berlin, 2004.
12. D. Chaum and E. van Heyst. Group Signatures. In *Eurocrypt '91*, LNCS 547, pages 257–265. Springer-Verlag, Berlin, 1992.
13. P. A. Fouque and D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In *Asiacrypt '01*, LNCS 2248. Springer-Verlag, Berlin, 2001.
14. S. Galbraith and V. Rotger. Easy Decision-Diffie-Hellman Groups. *LMS Journal of Computation and Mathematics*, 7:201–218, 2004.
15. A. Kiayias and M. Yung. Extracting Group Signatures from Traitor Tracing Schemes. In *Eurocrypt '03*, LNCS 2656, pages 630–648. Springer-Verlag, Berlin, 2003.
16. A. Kiayias and M. Yung. Group Signatures with Efficient Concurrent Join. In *Eurocrypt '05*, LNCS 3494, pages 198–214. Springer-Verlag, Berlin, 2005.
17. M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *Proc. of the 21st STOC*, pages 33–43. ACM Press, New York, 1989.
18. L. Nguyen and R. Safavi-Naini. Efficient and Provably Secure Trapdoor-free Group Signature Schemes from Bilinear Pairings. In *Asiacrypt '04*, LNCS 3329, pages 372–386. Springer-Verlag, Berlin, 2004.
19. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In *Eurocrypt '99*, LNCS 1592, pages 223–238. Springer-Verlag, Berlin, 1999.
20. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
21. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, Berlin, 1992.

A Security Notions: Experiments

A.1 Correctness

To any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$ we associate the experiment $\mathsf{Exp}^{corr}_{(\mathcal{GSS},\mathcal{A})}(k)$ (see figure 2), and

Experiment $\mathsf{Exp}^{corr}_{(\mathcal{GSS},\mathcal{A})}(k)$

$(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok}) \xleftarrow{R} \mathsf{GKg}(1^k)$;$\mathsf{CU} \leftarrow \emptyset$;$\mathrm{HU} \leftarrow \emptyset$;$(i, m) \xleftarrow{R} \mathcal{A}(\mathsf{gpk} : \mathsf{AddU}(.), \mathsf{RReg}(.))$
If $i \notin \mathsf{HU}$ then return 0; If $\mathsf{gsk}[i] = \epsilon$ then return 0
$\sigma \leftarrow \mathsf{GSig}(\mathsf{gpk}, \mathsf{gsk}[\mathrm{i}], m)$; If $\mathsf{GVf}(\mathsf{gpk}, m, \sigma) = 0$ then return 1
$(j, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \mathsf{reg}, m, \sigma)$; If $i \neq j$ then return 1
If $\mathsf{Judge}(\mathsf{gpk}, i, \mathsf{upk}[i], m, \sigma) = 0$ then return 1 else return 0

Fig. 2. Correctness Experiment

define

$$\mathsf{Adv}^{corr}_{(\mathcal{GSS},\mathcal{A})}(k) = \Pr[\mathsf{Exp}^{corr}_{(\mathcal{GSS},\mathcal{A})}(k) = 1].$$

A dynamic group signature scheme $\mathcal{GSS}$ is *correct* if $\mathsf{Adv}^{corr}_{(\mathcal{GSS},\mathcal{A})}(k) = 0$ for any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$ ($\mathcal{A}$ is not computationally restricted).

A.2 Anonymity

To any adversary $\mathcal{A}$, a bit $b \in \{0, 1\}$ (which defines the Ch_b-oracle) and any $k \in \mathbb{N}$ we associate the experiment $\mathsf{Exp}^{anon-b}_{(\mathcal{GSS},\mathcal{A})}(k)$ (see figure 3), and define

Experiment $\mathsf{Exp}^{anon-b}_{(\mathcal{GSS},\mathcal{A})}(k)$ // $b \in \{0, 1\}$

$(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok}) \xleftarrow{R} \mathsf{GKg}(1^k)$;$\mathrm{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$; $\mathsf{Gset} \leftarrow \emptyset$
$d \xleftarrow{R} \mathcal{A}(\mathsf{gpk}, \mathsf{ik} : \mathsf{Ch}_b(\cdot,\cdot,\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SndToU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{usk}(\cdot), \mathsf{CrptU}(\cdot,\cdot))$
return d

Fig. 3. Anonymity Experiment

$$\mathsf{Adv}^{anon}_{(\mathcal{GSS},\mathcal{A})}(k) = \Pr[\mathsf{Exp}^{anon-1}_{(\mathcal{GSS},\mathcal{A})}(k) = 1] - \Pr[\mathsf{Exp}^{anon-0}_{(\mathcal{GSS},\mathcal{A})}(k) = 1].$$

A dynamic group signature scheme $\mathcal{GSS}$ is *anonymous* if the function $\mathsf{Adv}^{anon}_{(\mathcal{GSS},\mathcal{A})}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

A.3 Traceability

To any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$ we associate the experiment $\mathsf{Exp}^{trace}_{(\mathcal{GSS},\mathcal{A})}(k)$ (see figure 4), and define

$$\mathsf{Adv}^{trace}_{(\mathcal{GSS},\mathcal{A})}(k) = \Pr[\mathsf{Exp}^{trace}_{(\mathcal{GSS},\mathcal{A})}(k) = 1].$$

A dynamic group signature scheme $\mathcal{GSS}$ is *traceable* if the function $\mathsf{Adv}^{trace}_{(\mathcal{GSS},\mathcal{A})}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

Experiment $\mathsf{Exp}^{trace}_{(\mathcal{GSS},\mathcal{A})}(k)$

$(\mathsf{gpk},\mathsf{ik},\mathsf{ok}) \xleftarrow{R} \mathsf{GKg}(1^k)$;$\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$

$(m,\sigma) \xleftarrow{R} \mathcal{A}(\mathsf{gpk},\mathsf{ok} : \mathsf{SndToI}(\cdot,\cdot),\mathsf{AddU}(\cdot,\cdot),\mathsf{RReg}(\cdot,\cdot),\mathsf{usk}(\cdot),\mathsf{CrptU}(\cdot,\cdot))$

If $\mathsf{GVf}(\mathsf{gpk},m,\sigma)=0$ then return 0; $(i,\tau) \leftarrow \mathsf{Open}(\mathsf{gpk},\mathsf{ok},\mathsf{reg},m,\sigma)$

If $i=0$ or $\mathsf{Judge}(\mathsf{gpk},i,\mathsf{upk}[i],m,\sigma,\tau)=0$ then return 1; else return 0

Fig. 4. Traceability Experiment

Experiment $\mathsf{Exp}^{nf}_{(\mathcal{GSS},\mathcal{A})}(k)$

$(\mathsf{gpk},\mathsf{ik},\mathsf{ok}) \xleftarrow{R} \mathsf{GKg}(1^k)$;$\mathsf{CU} \leftarrow \emptyset$; $\mathsf{HU} \leftarrow \emptyset$

$(m,\sigma,i,\tau) \xleftarrow{R} \mathcal{A}(\mathsf{gpk},\mathsf{ok},\mathsf{ik} : \mathsf{SndToU}(\cdot,\cdot),\mathsf{WReg}(\cdot,\cdot),\mathsf{GSig}(\cdot,\cdot),\mathsf{usk}(\cdot),\mathsf{CrptU}(\cdot,\cdot))$

If $\mathsf{GVf}(\mathsf{gpk},m,\sigma)=0$ then return 0

If the following are all true then return 1 else return 0:

- $i \in \mathsf{HU}$ and $\mathsf{gsk}[i] \neq \epsilon$ and $\mathsf{Judge}(\mathsf{gpk},i,\mathsf{upk}[i],m,\sigma,\tau)=1$
- $\mathcal{A}$ did not query $\mathsf{usk}(i)$ or $\mathsf{GSig}(i,m)$

Fig. 5. Non Frameability Experiment

A.4 Non-frameability

To any adversary $\mathcal{A}$ and any $k \in \mathbb{N}$ we associate the experiment $\mathsf{Exp}^{nf}_{(\mathcal{GSS},\mathcal{A})}(k)$ (see figure 5), and define

$$\mathsf{Adv}^{nf}_{(\mathcal{GSS},\mathcal{A})}(k) = \Pr[\mathsf{Exp}^{nf}_{(\mathcal{GSS},\mathcal{A})}(k)=1].$$

A dynamic group signature scheme $\mathcal{GSS}$ is *non-frameable* if the function $\mathsf{Adv}^{nf}_{(\mathcal{GSS},\mathcal{A})}(\cdot)$ is negligible for any polynomial-time adversary $\mathcal{A}$.

B Revocation

Formula in [6] can be extended to certificates of the form (A,x,y): for a revocated user $\mathcal{U}_i$, The group manager publishes $(B_i^\star, x_i, H_i^\star, K_i^\star)$, with

$$B_i^\star = \tfrac{1}{\gamma+x_i}G_2 \qquad H_i^\star = \tfrac{1}{\gamma+x_i}H \qquad K_i^\star = \tfrac{1}{\gamma+x_i}K.$$

The new public key can be constructed (by anyone) as follows:

$$\tilde{G_1} = \psi(B_i^\star) \qquad \tilde{G_2} = B_i^\star \qquad \tilde{W} = G_2 - x_i B_i^\star \qquad \tilde{H} = H_i^\star \quad \tilde{K} = K_i^\star$$

One can note that $\log_K H = \log_{\tilde{K}} \tilde{H}$. Then unrevoked users can update their certificates: let $\mathcal{U}$ be an unrevoked user, whose certificate is (A,x,y). We describe below how $\mathcal{U}$ updates his certificate, after the revocation of $\mathcal{U}_i$ (and thus from the above public information): $\mathcal{U}$ computes:

$$\tilde{A} = \tfrac{1}{x-x_i}(\psi(B_i^\star) + yH_i^\star) - \tfrac{1}{x-x_i}A.$$

One can verify that $(\tilde{A},x,y)$ is a valid certificate for the new group public key:

$$\begin{aligned}
(\gamma+x)\tilde{A} &= \frac{\gamma+x}{x-x_i}(\psi(B_i^\star) + yH_i^\star) - \frac{\gamma+x}{x-x_i}A \\
&= \frac{\gamma+x}{x-x_i}(\psi(B_i^\star) + yH_i^\star) - \frac{1}{x-x_i}(G_1 + yH) \\
&= \frac{\gamma+x}{x-x_i}(\psi(B_i^\star) + yH_i^\star) - \frac{1}{x-x_i}((\gamma+x_i)\psi(B_i^\star) + (\gamma+x_i)yH_i^\star) \\
&= \psi(B_i^\star) + yH_i^\star = \tilde{G_1} + y\tilde{H}.
\end{aligned}$$

The group manager can also update (keeping track of all the modifications for the verification of the signature) the users' database: from $(\mathsf{upk}, A, x, yH, S)$, it can compute the new values (note that y is not needed, but yH only.) □

C Honest-Verifier Zero-Knowledge Proof of Knowledge – Proof of Lemma 5

Such a proof of knowledge must satisfy the following properties:

C.1 Completeness.

An honest prover, who owns a valid triple (A, x, y), will be accepted: Let

$$\mathsf{gpk} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \psi; G_1, K, H = \xi_1 K, G = \xi_2 K; G_2, W = \gamma G_2)$$

be the common group public key, and $\mathcal{U}$ an honest prover, whose triple is (A, x, y). From the generation of all the elements, one can easily check the completeness.

C.2 Soundness.

Let us assume that an algorithm $\mathcal{A}$ succeeds with non-negligible probability: after having sent the commitments $(T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4)$, its success probability is non-negligible. Therefore, it can answer correctly, with non-negligible probability, to the two challenges c and c', by $(s_\alpha, s_\beta, s_x, s_z)$ and $(s'_\alpha, s'_\beta, s'_x, s'_z)$ respectively. Let us denote by c'' the difference between c and c' (which is non-zero modulo p, since $0 \leq c \neq c' < 2^k < p$: We also denote by s''_α, s''_β, s''_x and s''_z the relative differences of the answers:

$$s''_\alpha K = c'' T_1 \quad s''_\beta K = c'' T_3 \quad s''_\alpha H - s''_\beta G = c''(T_2 - T_4)$$
$$(T_2, G_2)^{s''_x} \cdot e(H, W)^{-s''_\alpha} \cdot e(H, G_2)^{-s''_z} = (e(G_1, G_2)/e(T_2, W))^{c''}$$

If we define

$$\tilde{x} = s''_x/c'' \quad \tilde{z} = s''_y/c'' \quad \tilde{\alpha} = s''_\alpha/c'' \quad \tilde{\beta} = s''_\beta/c'',$$

we have

$$T_1 = \tilde{\alpha} K \qquad T_3 = \tilde{\beta} K \qquad T_2 - T_4 = \tilde{\alpha} H - \tilde{\beta} G$$

and

$$e(T_2, G_2)^{\tilde{x}} \cdot e(H, W)^{-\tilde{\alpha}} \cdot e(H, G_2)^{-\tilde{z}} = e(G_1, G_2)/e(T_2, W).$$

Therefore, with $\tilde{A} = T_2 - \tilde{\alpha} H$ and $\tilde{y} = \tilde{z} - \tilde{\alpha}\tilde{x}$, one gets

$$e(\tilde{A}, G_2)^{\tilde{x}} \cdot e(\tilde{A}, W) = e(G_1, G_2) \cdot e(H, G_2)^{\tilde{y}}$$

which means that $(\tilde{A}, \tilde{x}, \tilde{y})$ is a valid certificate: $(\tilde{x} + \gamma)\tilde{A} = G_1 + \tilde{y} H$. Indeed, e is non-degenerated, and thus $e(\cdot, G_2)$ is an injection.

C.3 Honest-Verifier Zero-Knowledge.

For an honest verifier, the transcripts (in name of any user) can be simulated in an indistinguishable way, without knowing any certification, and namely the one of the target user, under the XDH assumption: One needs to simulate a transcript (T_1, T_2, T_3, T_4), (R_1, R_2, R_3, R_4), c and $(s_\alpha, s_\beta, s_x, s_z)$, without knowing any valid certificate. We first need to simulate the quadruple (T_1, T_2, T_3, T_4), which can be done by randomly choosing A, α and β in the appropriate groups, and compute

$$T_1 = \alpha K \quad T_2 = A + \alpha H \quad T_3 = \beta K \quad T_4 = A + \beta G.$$

Under the XDH assumption, this quadruple is indistinguishable from the output on any prover. The following will not assume any knowledge about the data used to generate this quadruple (T_1, T_2, T_3, T_4). And the simulation will be perfect, as for any Schnorr-like proof: one first randomly chooses the challenge $c \in \{0,1\}^k$ and then s_α, s_β, s_x, s_z in $\mathbb{Z}_p$. one simply computes (R_1, R_2, R_3, R_4) using the verification equations.