Efficient Group Signatures in the Standard Model

Laila El Aimani[1] and Olivier Sanders[2] *

[1] Independent researcher
[2] Orange Labs, Applied Crypto Group, Caen, France

Abstract. In a group signature scheme, group members are able to sign on behalf of the group. Since the introduction of this cryptographic authentication mechanism, several schemes have been proposed but only few of them enjoy a security in the standard model. Moreover, those provided in the standard model suffer the recourse to non standard-assumptions, or the expensive cost and bandwidth of the resulting signature.
We provide three practical group signature schemes that are provably secure in the standard model under standard assumptions. The three schemes permit dynamic enrollment of new members while keeping a constant size for both keys and group signatures, and they improve the state-of-the art by several orders of magnitude.

Key words: Group signature, bilinear groups, standard model, non-interactive zero-knowledge.

1 Introduction

Group signatures, introduced in 1991 by Chaum and Van Heyst [16] allow members of a group to anonymously sign messages on behalf of the whole group. However, to prevent abuses, the group is controlled by a group manager that has the ability to *open* the group signature, *i.e.* to identify the signer of a message. Group signatures have proved to be extremely useful in various applications, for example keycard access to restricted areas, where it is necessary to secure areas to only employees of the group without tracking individual employee's movements.

Related works. Since their introduction, a great number of security properties that group signatures should meet have been introduced until Bellare, Micciancio and Warinschi [5] provided appropriate definitions and formalized the intuitive informal requirements of previous works. In fact, they proposed two properties for static groups, namely *full anonymity* and *full traceability*, that captured all previous requirements. Full anonymity requires that group signatures reveal no information about the signer, even in the presence of a powerful adversary who has access to an opening oracle and to all users secret keys. Full traceablity requires that the group manager is always able to identify the signer of a valid group signature or a member of the coalition that issued it. Bellare, Shi and Zhang [6] extended these notions to dynamic groups and added the notion of *non-frameability*, which requires that even a dishonest group manager and a coalition of group members cannot falsely accuse an honest user of having issued some group signature.
Boneh and Shacham [10] proposed a weaker notion of anonymity, called *selfless anonymity*, where signers can trace their own signatures. Further schemes ([11], [12]) introduced another weak version of anonymity, namely the CPA-*anonymity* where the adversary does not have access to an opening oracle. As mentioned in [7], this is a much more serious limitation because the opening functionality becomes virtually useless.

Most practical group signatures schemes ([7], [17]) that have been proposed are proven secure in the random oracle model (ROM). Indeed, despite its inherent flaws ([14], [4]), the ROM compares much better than the standard model with respect to efficiency. Ateniese *et al.* [3] gave an efficient group signature scheme in the standard model but proved its security under non-standard assumptions. Moreover, the security proof of anonymity (lemma A.2 of their paper) does not mention any access to opening oracles by the adversary and thus differs from the standard CCA-anonymity experiment. Actually, this CCA-anonymity seems hard to reach since their group signatures are partially re-randomizable, so any adversary against the selfless

* This work was done while both authors were with Technicolor.

anonymity is able to re-randomize the first part of the challenge group signature and make an opening request on the result. In 2007, Groth [19] gave the first efficient realization, achieving full anonymity in the standard model, where the size of group signatures is about 50 elements.

Our contributions. We propose three groups signatures schemes, proven secure in the standard model under standard assumptions, that improve the size of group signatures and the number of pairings required for the signature verification.
In our constructions, each user joining the group receives a certificate on a key while the group manager receives some information allowing him to open the group signatures produced by this user. To produce a group signature on a message, the user will use his key to sign, with a digital signature scheme, the message, a randomized part of the certificate and the verification key of a strong one-time signature. Then he will produce a non-interactive zero-knowledge proof (NIZK) to prove that the key used to sign the message is certified. Finally, to prevent anyone else to randomize his group signature, the user will sign some of its elements with the strong one-time signature scheme.
Our first group signature scheme, whose group signatures comprise only 22 group elements, achieves selfless anonymity without the need for encryption. Indeed, the use of the Camenish-Lysyanskaya signature scheme [13] makes it possible to disclose the digital signature on the message in the group signature, and thus to decrease the cost and size of the non-interactive proof of knowledge. The scheme resorts however to a trusted party to set up the system parameters. Moreover the cost of the opening algorithm is linear in the number of members.
Our second group signature scheme overcomes the drawbacks of the first scheme but at the expense of a slight increase in the size of the group signature (28 group elements). In fact, users certify now their verification keys (instead of their siging keys), allowing the group manager to set up the system parameters and to extract the identity of the group signature issuer using his extraction key. Since the verification keys are group elements, we now require that our certificate scheme is structure-preserving [1], which explains the extra-cost of the scheme.
Finally we propose a generic group signature scheme that generalizes Groth's scheme, and which achieves full anonymity. The construction makes use of an encryption scheme in order to be able to use any EUF-CMA digital signature scheme, which increases the size/cost of the group signature. An instantiation of this construction results in a group signature with 30 group elements.

We summarize in Figure the signature sizes (number of group elements), the verification costs (number of pairings) and the opening cost (as a function of the number of participants, denoted r) of our contributions (GS1, GS2, GS3) and compare them with the state of the art.

Scheme	Signature size	Verification cost	Opening cost	Standard model?
BCNSW [7]	5	2	O(r)	No
DP [17]	9	1	O(1)	No
Groth [19]	50	246	O(1)	Yes
GS1 (Section 4)	22	73	O(r)	Yes
GS2 (Section 5)	28	124	O(1)	Yes
GS3 (Section 6)	30	92	O(r)	Yes

Fig. 1. Comparison between group signatures performances

Outline of the paper. The rest of this paper is organized as follows. In section 2 we recall definitions of the notions that we use in our schemes. Section 3 defines the syntax and the security model of a secure group signature scheme. Following sections present our group signature schemes. We defer the security proofs to the appendix.

2 Preliminaries

In this section we recall some necessary bricks that will be used throughout the document, namely bilinear maps, Camenisch-Lysyanskaya's signatures, and signatures on committed values. Note that we defer in Appendix A the recall of further primitives such as (one-time, structure-preserving) signatures, tag-based encryption, and non-interactive proofs of knowledge.

2.1 Bilinear groups

Our constructions use cyclic groups of prime order that have a non-degenerate efficiently computable bilinear map e. We use the following notations:

- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order q.
- We write the group operations multiplicatively in each group.
- A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has the following properties:
  1. For all $x \in \mathbb{G}_1, y \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q$ we have $e(x^a, y^b) = e(x, y)^{ab}$.
  2. For $x \neq 1_{\mathbb{G}_1}$ or $y \neq 1_{\mathbb{G}_2}$, $e(x, y) \neq 1_{\mathbb{G}_T}$.
  3. e is efficiently computable.

The Symmetric External Diffie-Hellman (SXDH) assumption: We say $SXDH$ holds in the bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ if DDH is hard in both groups, where DDH (Decisional Diffie-Hellman) is the following problem:

$$\text{Given } (g, g^a, g^b, g^c) \text{ it is hard to decide whether } c = ab \bmod q \text{ or random.}$$

2.2 Camenisch-Lysyanskaya Signatures

Our group signature schemes make use of the pairing-based Camenish-Lysyankaya signature schemes [13] (scheme A and scheme C in their paper), which are provably secure under the LSRW assumption [21]. The two schemes operate in three cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ of prime order q, equipped with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and two generators $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$.

Lysyanskaya-Sahai-Rivest-Wolf (LSRW) assumption:
Let $\widetilde{g} \in \mathbb{G}_2$, $\widetilde{x} = \widetilde{g}^{\alpha}$, $\widetilde{y} = \widetilde{g}^{\beta}$ and $O_{\widetilde{x},\widetilde{y}}(.)$ be an oracle that, on input a value $u \in \mathbb{Z}_q$, outputs $A = (a, a^{\beta}, a^{\alpha+u\alpha\beta}) \in \mathbb{G}_1^3$ for a randomly chosen $a \in \mathbb{G}_1$. Then for all probabilistic polynomial time adversaries $\mathcal{A}$, the quantity ϵ is negligible:

$$\epsilon = \mathtt{Pr}[\alpha, \beta \leftarrow \mathbb{Z}_q; \widetilde{x} \leftarrow \widetilde{g}^{\alpha}, \widetilde{y} \leftarrow \widetilde{g}^{\beta}; (u, a, b, c) \leftarrow \mathcal{A}^{O_{\widetilde{x},\widetilde{y}}}(\widetilde{x}, \widetilde{y}) :$$

$$u \notin Q \wedge a \in \mathbb{G}_1 \wedge b = a^{\beta} \wedge c^{\alpha+u\alpha\beta}]$$

where Q is the set of queries asked by $\mathcal{A}$ to his oracle $O_{\widetilde{x},\widetilde{y}}(.)$.

[Key generation]	Choose $\alpha, \beta \xleftarrow{R} \mathbb{Z}_d$ then compute $X \leftarrow \widetilde{g}^{\alpha}$ and $Y \leftarrow \widetilde{g}^{\beta}$
	set $pk \leftarrow \{X, Y\}$ and $sk \leftarrow \{\alpha, \beta\}$.
[Signature on m**]**	Choose a random $a \in \mathbb{G}_1$
	and outputs : $\sigma = (a, a^{\beta}, a^{\alpha+m\alpha\beta})$
[Verification]	Given pk, m and $\sigma = (a, b, c)$
	check if the following equations holds:
	$e(a, Y) = e(b, \widetilde{g})$ and $e(a, X).e(b, X)^m = e(c, \widetilde{g})$

Fig. 2. Camenisch-Lysyanskaya's signature A

[Key generation]	Choose $\alpha, \beta, z_i \xleftarrow{R} \mathbb{Z}_d$ then compute $X \leftarrow \widetilde{g}^{\alpha}$, $Y \leftarrow \widetilde{g}^{\beta}$ and $Z_i \leftarrow \widetilde{g}^{z_i}$ for $1 \leq i \leq l$
	set $pk \leftarrow \{X, Y, Z_i\}_{1 \leq i \leq l}$ and $sk \leftarrow \{\alpha, \beta, z_i\}_{1 \leq i \leq l}$.
[Signature on $(m_0, ..., m_l)$**]**	Choose a random $a \in \mathbb{G}_1$
	$A_i = a^{z_i}$ for $1 \leq i \leq l$
	$b = a^{\beta}$, $B_i = (A_i)^{\beta}$
	$c = a^{\alpha + \alpha\beta m_0} \prod_{i=1}^{l} A^{\alpha\beta m_i}$
	and outputs the signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$
[Verification]	Given pk, m and σ
	check if the following verification equations holds:
	$e(a, Z_i) = e(A_i, \widetilde{g})$
	$e(a, Y) = e(b, \widetilde{g})$ and $e(A_i, Y) = e(B_i, \widetilde{g})$
	$e(a, X).e(b, X)^{m_0} \prod_{i=1}^{l} e(B_i, X)^{m_i} = e(c, \widetilde{g})$

Fig. 3. Camenisch-Lysyanskaya's signature C

2.3 Re-randomizable signatures on committed values

Our constructions use *i.e.* digital signature schemes Σ with the following properties:

- The signature scheme is *re-randomizable* i.e. it admits a function `SigRand` such that for each signature σ on a message m, $\texttt{SigRand}(\sigma) = \sigma'$, where σ' is a valid signature on m and such that no probabilistic polynomial time adversary $\mathcal{A}$, with access to a signing oracle, is able, given a signature σ, to distinguish a randomized version of σ from a signature on a random element of the message space.
- Σ admits an algorithm `comSign` to obtain signatures on committed values which, on input c (where $(c, r) \leftarrow \texttt{Commit}(m)$) and $\pi \leftarrow \text{POK}\{(m, r) : (c, r) = \texttt{Commit}(m)\}(c)$, outputs σ such that $\Sigma.\texttt{verify}(\sigma, (m, r)) = 1$.

We define EUF-CMA security the same way as the standard digital signature schemes (a re-randomizable signature scheme cannot obviously reach the SEUF-CMA security). As shown in [13] the signature scheme described in figure 3 is EUF-CMA secure and admits an efficient protocol (described in Fig. 4) for obtaining signature on committed value. Moreover, using a proof similar to that of [7], this scheme is re-randomizable if the DDH-assumption holds in $\mathbb{G}_1$.

3 Defining Group Signatures

3.1 Syntax

A group signature scheme consists of a set of users, identified by a unique index i, that can produce signatures on behalf of the group. Users must interact with a certification authority and the group manager to join the group. At the end of this interaction, each user obtains a signature key pair and the group manager obtains some information that will allow him to identify the group signatures issuers. The syntax that we require is as follows.

- `Keygen`(λ): This algorithm inputs a security parameter λ and outputs (gpk, sk_{cert}, sk_M,**Sreg**,**reg**), where gpk is the group public key, sk_{cert} is the issuer's secret key, sk_M is the group manager's secret key, **Sreg** is the group manager's secret register and **reg** is a public register.

- `Join`(gpk, upk_i): The `Join` protocol is used to add a new user to the group. It consists of one interactive protocol between the user and the issuer and another one between the user and the group manager. The

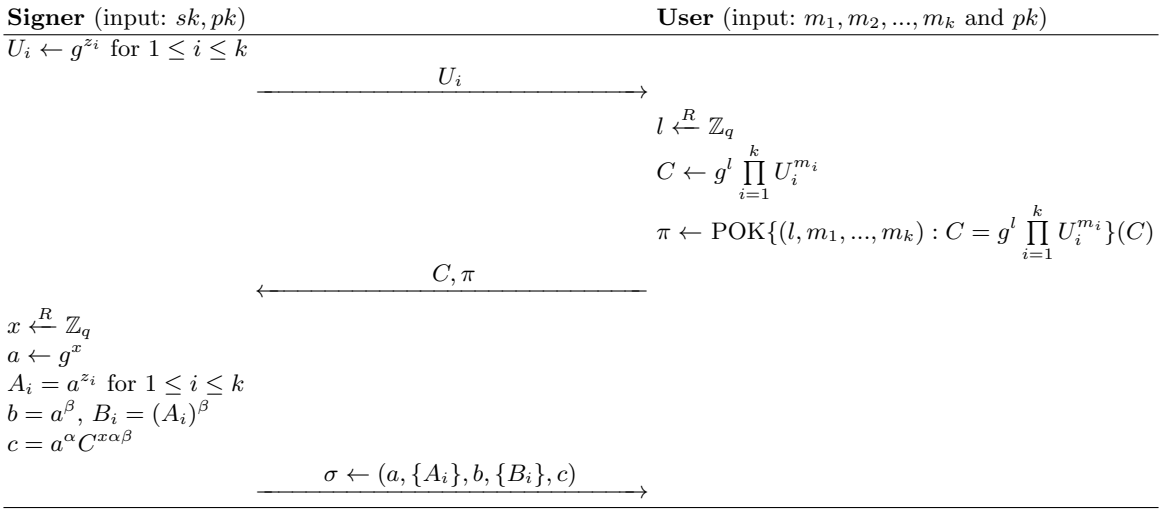

Fig. 4. Protocol for obtaining a CL signature C on committed values

common inputs of this protocol is gpk and the user's public key upk_i. The private input of the user is usk_i whereas the private inputs of the issuer and the group manager are sk_{cert} and sk_M respectively. As a result of the first interaction, the user obtains his group signing key sk_i, the verification key vk_i and a certificate, $\sigma_{cert}(i)$, proving membership of the group. Then he uses usk_i to compute σ_i, a digital signature on vk_i. At the end of the second interaction, the group manager obtains and stores vk_i and σ_i in **Sreg**[i] and publishes upk_i in **reg**[i].

- `Sign`$(gpk, sk_i, \sigma_{cert}(i), m)$: This algorithm inputs a message m, the user's secret key and certificate and outputs a group signature μ on m.

- `Verify`(gpk, μ, m) : This algorithm inputs a message m and a group signature μ and outputs 1 if μ is a valid group signature on m, and 0 otherwise.

- `Open`(gpk, m, μ, sk_M) : This algorithm inputs a message m, a group signature μ and the group manager's secret data sk_M and **Sreg**. If μ is a valid group signature on m from user i it returns a proof τ of this statement and the identity i, else it returns $\perp$.

- `Judge`(gpk, m, μ, i, τ) : This algorithm inputs a message m, a group signature μ, an identity i and a proof of knowledge τ. It returns 1 if τ is a valid proof that μ is a group signature on m, issued by user i. Otherwise it returns 0.

3.2 Security model

In this section we give the security definitions that we require for group signature schemes. We adhere to the model defined by [6], except that we use the *selfless anonymity* notion from [10] instead of the *full anonymity* for our first two group signature schemes. Informally, selfless anonymity, contrarily to full anonymity, does not protect dishonest users, i.e. the adversary does not know the private keys of the identities he wishes to be challenged on. As in [7], we consider a setting with n users divided statically into sets $\mathcal{HU}$ and $\mathcal{DU}$ of honest and dishonest users respectively (*i.e* an honest user cannot be corrupted during the experiment). This static division implies, by guessing the indices of "target" users in the anonymity and non-frameability experiment, security in the dynamic corruption case.
The security notions make use of the following oracles:

– $\mathcal{O}\texttt{Join}_{UD}(gpk, upk_i, sk_M)$ is an oracle that executes the user's side of the join protocol for the input user $i \in \mathcal{HU}$. This oracle will be used by an adversary playing the role of the corrupted group manager.

– $\mathcal{O}\texttt{Join}_{DM}(gpk, upk_i, usk_i)$ is an oracle that executes the join protocol with the honest group manager. This oracle will be used by an adversary to register a corrupted user.

– $\mathcal{O}\texttt{Sign}(i, m)$ is an oracle that accepts as input an identity i and a message m and returns a group signature μ if user i is honest and registered.

– $\mathcal{O}\texttt{Open}(m, \mu)$ inputs a message-signature pair (m, μ) and returns the result of the function call $\texttt{Open}(\mathbf{Sreg}, sk_M, \mu, m)$. We use the notation $\mathcal{O}\texttt{Open}\neg(m, \mu)$ when the query (m, μ) is not allowed.

Correctness: We define the correctness of a group signature scheme through a game in which an adversary is allowed to request a signature on some message by any of the honest group members. The adversary wins if either the resulting signature does not pass the verification test, the signature is opened as if it were produced by a different user, or the judging algorithm returns 0. The group signature scheme is correct if for any adversary $\mathcal{A}$ and any security parameter λ (we keep this notation in the following experiments), $\texttt{Pr}[\mathbf{Exp}^{corr}_{\mathcal{A}}(\lambda) = 1]$ is negligible in λ, where $\mathbf{Exp}^{corr}_{\mathcal{A}}(\lambda)$ is defined as follows:

1. $\mathcal{HU} \leftarrow \{1..n\}$.
2. $(gpk, sk_{cert}, sk_M) \leftarrow \texttt{Keygen}(\lambda)$.
3. $(sk_i, vk_i, \sigma_{cert}(i)) \leftarrow \texttt{Join}(gpk, sk_{cert}, sk_M)$ for each user $i \in \mathcal{HU}$.
4. $(i, m) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Sign}, \mathcal{O}\texttt{Open}}(gpk)$.
5. If $i \notin \mathcal{HU}$ then return 0.
6. $\mu \leftarrow \texttt{Sign}(sk_i, m)$.
7. If $\texttt{Verify}(gpk, m, \mu) = 0$ then return 1.
8. If $\texttt{Open}(\mathbf{Sreg}, sk_M, \mu, m) = (j, \tau)$ and $j \neq i$ then return 1.
9. If $\texttt{Judge}(m, \mu, \tau, i) = 0$ then return 1.

Anonymity: Informally, anonymity requires that group signatures do not reveal the signer's identity. In the selfless-anonymity game the adversary's goal is to determine which of the two users generated the challenge signature. The difference with the full anonymity of [6] consists in not giving the adversary $\mathcal{A}$ access to either private key. As in [10] and [7] we define the selfless-anonymity experiment $\mathbf{Exp}^{anon-b}_{\mathcal{A}}(\lambda)$ as follows:

1. $\mathcal{DU} \leftarrow \mathcal{A}(\lambda)$.
2. $\mathcal{HU} \leftarrow \{1..n\} \setminus \mathcal{DU}$.
3. $(gpk, sk_{cert}, sk_M) \leftarrow \texttt{Keygen}(\lambda)$.
4. $(sk_i, vk_i, \sigma_{cert}(i)) \leftarrow \texttt{Join}(gpk, sk_{cert}, sk_M)$ for each user $i \in \mathcal{HU}$.
5. $(m, i_0, i_1) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Sign}, \mathcal{O}\texttt{Open}, \mathcal{O}\texttt{Join}_{DM}}(gpk)$ with $(i_0, i_1 \in \mathcal{HU})$.
6. $\mu \leftarrow \texttt{Sign}(sk_{i_b}, m)$ for $b \xleftarrow{R} \{0, 1\}$.
7. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Sign}, \mathcal{O}\texttt{Open}\neg(m,\mu), \mathcal{O}\texttt{Join}_{DM}}(gpk)$.
8. Return b^*.

We define $\mathbf{Adv}^{anon-b}_{\mathcal{A}}(\lambda) = |\texttt{Pr}[b = b^*] - \frac{1}{2}|$. The group signature scheme is anonymous if for any probabilistic polynomial time adversary, this advantage is a negligible function of λ. In the full-anonymity experiment, we no longer require that $i_0, i_1 \in \mathcal{HU}$, we then say that the group signature scheme is fully-anonymous.

Traceability: Traceability requires that no adversary is able to create a valid signature that cannot be traced to some user already registered. We define the traceability experiment as follows:

1. $\mathcal{DU} \leftarrow \{1..n\}$.
2. $(gpk, sk_{cert}, sk_M) \leftarrow \texttt{Keygen}(\lambda)$.
3. $(m, \mu) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Open}, \mathcal{O}\texttt{Join}_{DM}}(gpk)$.
4. If $\texttt{Verify}(gpk, m, \mu) = 1$ and $\texttt{Open}(\mathbf{Sreg}, sk_M, \mu, m) = \perp$ then return 1.

5. Return 0.

We define $\mathbf{Adv}^{trace}_{\mathcal{A}}(\lambda) = \mathtt{Pr}[\mathbf{Exp}^{trace}_{\mathcal{A}}(\lambda) = 1]$. The group signature scheme is traceable if for any probabilistic polynomial time adversary, this advantage is a negligible function of λ.

Non-frameability: Informally, non-frameability requires that a cheating group manager cannot falsely accuse an honest user of having signed a given message m. We define the non-frameability experiment as follows:

1. $\mathcal{DU} \leftarrow \mathcal{A}(\lambda)$.
2. $\mathcal{HU} \leftarrow \{1..n\} \setminus \mathcal{DU}$.
3. $(gpk, sk_{cert}, sk_M) \leftarrow \mathtt{Keygen}(\lambda)$.
4. $(sk_i, vk_i, \sigma_{cert}(i)) \leftarrow \mathtt{Join}(gpk, sk_{cert}, sk_M)$ for each user $i \in \mathcal{HU}$.
5. $(i, m, \mu) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Sign}, \mathcal{O}\mathtt{Join}_{UD}}(sk_M, gpk)$.
6. If $i \notin \mathcal{HU}$ or $\mathtt{Verify}(gpk, m, \mu) = 0$ then return 0.
7. If m was queried to $\mathcal{O}\mathtt{Sign}$ then return 0.
8. If $\mathtt{Judge}(m, \mu, \tau, i) = 0$ then return 0.
9. Return 1.

We define $\mathbf{Adv}^{nf}_{\mathcal{A}}(\lambda) = \mathtt{Pr}[\mathbf{Exp}^{nf}_{\mathcal{A}}(\lambda) = 1]$. The group signature scheme is non-frameable if for any probabilistic polynomial time adversary, this advantage is a negligible function of λ.

Remark: We can also define the strong non-frameability experiment in which the adversary's goal is to accuse an honest user of having created a given signature μ (we replace the line 7 by: if μ was produced by $\mathcal{O}\mathtt{Sign}$ then returns 0).

4 A group signature without encryption

4.1 Description

The core of this group signature scheme is the combination of a re-randomizable signature scheme Σ_{cert} and the signature scheme Σ_1 described in Figure 2. We assume that the system parameters are set up by a trusted party and that each user i has a key pair (usk_i, upk_i) for a signature scheme Σ_0 where upk_i is public and associated to i. At the end of the $\mathtt{Keygen}$ algorithm, the issuer receives sk_{cert}. When a new member joins the group, he creates a key pair (sk_i, vk_i) for Σ_1, then gets a certificate $\sigma_{cert}(i)$ on sk_i (using the protocol for obtaining signatures on committed values) and finally sends vk_i and a signature on it (using usk_i) to the group manager who records it in his secret register **Sreg**. When making a group signature on a message m, the member will first re-randomize his certificate and generate a key pair (sk_{ots}, vk_{ots}) for a strong one-time signature. Then he will use his secret key sk_i to sign the concatenation of m, the certificate and vk_{ots} and give a non-interactive zero-knowledge proof of knowledge that the certificate is a valid signature on sk_i. To prevent an adversary from randomizing the signature or the non-interactive proof, the user will sign them with the strong one-time signature. We use the following notations to describe our group signature scheme:

Notations:

- λ is the security parameter.
- $\mathcal{G}$ is a probabilistic polynomial time algorithm that, on input λ, generates $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ where $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$.
- K_{NI} is a probabilistic polynomial time algorithm that, on input $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$, generates a common reference string crs for the Groth-Sahai proof system.
- **reg** is a public register.
- **Sreg** is the secret register of the group manager.
- gpk is the group public key.

The algorithms defining our group signature scheme are described in figure 5.

Keygen(λ)	**Verify**(m, μ)
$gk \leftarrow \mathcal{G}(\lambda)$	If $V_{NI}(crs, \mu, m, \pi) = 1 \wedge$
$crs \leftarrow K_{NI}(gk)$	$\Sigma_{ots}.\texttt{verify}(vk_{ots}, \sigma_{ots}, a \| \pi) = 1$
$(sk_{cert}, pk_{cert}) \leftarrow \Sigma_{cert}.\texttt{keygen}(gk)$	then return 1
$gpk \leftarrow (gk, pk_{cert}, crs, \mathbf{reg})$	Else return 0.
Join(user: usk_i, issuer: sk_{cert}, group manager: **Sreg**)	**Open**(**Sreg**$, m, \mu)$
$(sk_i, vk_i) \leftarrow \Sigma_1.\texttt{Keygen}(gk)$	If $\texttt{Verify}(m, \mu) = 0$
$\sigma_{cert}(i) \leftarrow \Sigma_{cert}.\texttt{comSign}(sk_{cert}, \texttt{Commit}(sk_i), \pi)$	then return 0.
$\sigma_i \leftarrow \Sigma_0.\texttt{sign}(usk_i, vk_i)$	Parse μ as $(m, vk_{ots}, \sigma_{ots}, \sigma_1, \sigma_{cert}, \pi)$
$\mathbf{Sreg}[i] \leftarrow (vk_i, \sigma_i)$	for vk_i in **Sreg**$[i]$:
$\mathbf{reg}[i] \leftarrow (i, upk_i)$	if $\Sigma_1.\texttt{Verify}(vk_i, m \| vk_{ots} \| \sigma_{cert}, \sigma_1) = 1$
	return i and a proof τ.
Sign$(sk_i, \sigma_{cert}(i), m)$	with $\tau \leftarrow \text{POK}\{(vk_i, \sigma_i) :$
$\sigma_{cert}(i) \leftarrow \Sigma_{cert}.\texttt{sigRand}(\sigma_{cert}(i))$	$\Sigma_1.\texttt{Verify}(vk_i, m \| vk_{ots} \| \sigma_{cert}, \sigma_1) = 1 \wedge$
$(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}.\texttt{keygen}$	$\Sigma_0.\texttt{Verify}(upk_i, vk_i, \sigma_i) = 1$
$(a, b, c) \leftarrow \Sigma_1.\texttt{sign}(sk_i, m \| vk_{ots} \| \sigma_{cert}(i))$	$\}(m, \mu, \mathbf{reg})$
$\pi \leftarrow \text{POK}\{(sk_i) :$	
$(a, b, c) = \Sigma_1.\texttt{sign}(sk_i, m \| vk_{ots} \| \sigma_{cert}(i)) \wedge$	**Judge**(m, μ, τ, i)
$\sigma_{cert}(i) = \Sigma_{cert}.\texttt{sign}(sk_{cert}, sk_i)$	If $V_{NIZK}(crs, m, \mu, \tau, i) = 1$
$\}(m, (a, b, c), \sigma_{cert}(i))$	return 1.
$\sigma_{ots} \leftarrow \Sigma_{ots}.\texttt{sign}(sk_{ots}, a \| \pi)$	Else return 0.
$\mu \leftarrow (m, vk_{ots}, \sigma_{ots}, (a, b, c), \sigma_{cert}(i), \pi)$	

Fig. 5. Group signature scheme 1

4.2 Security proofs

Theorem 1. *If Σ_{cert} is a EUF-CMA secure signature scheme and (P, V) is a sound zero-knowledge proof system, then the group signature is fully traceable.*

Theorem 2. *If Σ_0 and Σ_1 are EUF-CMA secure signature schemes, Σ_{ots} is a strong one-time signature scheme and (P, V) is a sound, zero-knowledge proof system, then the group signature scheme is strongly non-frameable.*

Theorem 3. *If Σ_0 and Σ_1 are EUF-CMA secure signature schemes, Σ_{ots} is a strong one-time signature and (P, V) is a sound, zero-knowledge proof system, then the group signature scheme is anonymous under the SXDH assumption.*

The proofs are provided in appendix B.

4.3 A concrete realization

We instantiate this system with the Λ_{sxdh} setting which refers to the case of asymmetric pairings for which the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.
We use the CL signature scheme described in Fig. 3 for Σ_{cert}, the weakly secure Boneh-Boyen signature scheme described in Fig. 8 for Σ_{ots} and the structure-preserving signature described in Fig. 9 for Σ_0.
We use the following notations to describe the proofs of knowledge:

Notations:

- $(sk_i, pk_i) \leftarrow ((\alpha_i, \beta_i), (\widetilde{g}^{\alpha_i}, \widetilde{g}^{\beta_i}))$
- $(sk_{cert}, pk_{cert}) \leftarrow ((u, v, z_1, z_2), (U, V, Z_1, Z_2))$
- $(sk_M, pk_M) \leftarrow ((x_1, \widetilde{x}_1, x_2, \widetilde{x}_2), (X_1, \widetilde{X}_1, X_2, \widetilde{X}_2))$
- $\sigma_1 \leftarrow (a_1, b_1, c_1)$; $\sigma_{cert} \leftarrow (a, A_1, A_2, b, B_1, B_2, c)$

– $\eta \leftarrow m \| vk_{ots} \| \sigma_{cert}$

A group signature, $(vk_{ots}, \sigma_{ots}, (a, b, c), \sigma_{cert}(i))$ involves 13 group elements, and a proof π of the following equations (the variables are underlined):

$$e(a_1, \widetilde{g})^{\underline{\beta_i}} = e(b_1, \widetilde{g}) \quad (1)$$

$$(e(a_1, \widetilde{g}).e(b_1, \widetilde{g})^{\eta})^{\underline{\alpha_i}} = e(c_1, \widetilde{g}) \quad (2)$$

$$e(c, \widetilde{g}) = e(a, U).e(b, U)^{\underline{l}} .e(B_1, U)^{\underline{\alpha_i}} .e(B_2, U)^{\underline{\beta_i}} \quad (3)$$

where η is $m \| vk_{ots} \| \sigma_{cert}$ and l is the randomness used by user i in the join stage to obtain signature on the committed values α_i and β_i.
These equations involve 3 variables (α_i, β_i, l) and thus increase the size of the proof by 6 group elements. Equations (1) and (2) proves that (a_1, b_1, c_1) is a valid Camenisch-Lysyanskaya signature from user i on η. These two equations cost 1 group element each. Equation (3) proves that the certificate is a valid signature on α_i and β_i and costs 1 group element. The total size of the proof is then 9 group elements.

The total size of the group signature is thus 22 group elements, which amounts to 700 Bytes if we consider an implementation using 256-bit groups sizes. Using naive computations, equation (1) needs for the verification 18 pairings, equation (2), 19 pairings and equation (3), 25 pairings. Checking the consistency of the certificate and of the one-time signature adds 11 pairings. The total cost of the `Verify` algorithm is then 73 pairings.

Remark: We can further reduce the size of the group signature if the group manager and the certificate issuer are the same entity. Indeed, in the `Join` protocol we no longer need to use the hiding randomness l which decreases both the size of the certificate and the size of the proof by two elements. The size of the group signature is then 18 group elements.

5 A group signature without encryption and without trusted parties

5.1 Description

The main drawbacks of the previous scheme lie in requiring a trusted party to set up the system parameters, and having the cost of the algorithm `Open` linear in the number of users. To solve these two problems, users will now certify their verification keys (instead of their siging keys), allowing thus the group manager to set up the system parameters and to use the extraction keys ek of the Groth-Sahai proofs to extract the identity behind a group signature.

This construction is similar to the previous scheme except that we require a different signature scheme for Σ_{cert}. We now assume that any digital signature σ, generated using Σ_{cert} on an arbitrary message m can be efficiently transformed in a reversible way to a pair (r, s) where r (r may be the empty string) is information theoretically independent from m, *i.e.* there exists an algorithm `Simulate` that inputs $\Sigma_{cert}.sk$ and a message from the message space and outputs a string indistinguishable from r. We require further the partial randomizability property for Σ_{cert}, *i.e.* Σ_{cert} admits two algorithms $\texttt{SigPrand}_0$ and $\texttt{SigPrand}_1$, defined as follows:
For each signature $\sigma \rightarrow (r, s)$ on a message m:

– $(r', state) \leftarrow \texttt{SigPrand}_0(r)$
– $s' \leftarrow \texttt{SigPrand}_1(s, state)$

where r' is unlinkable to r and $\sigma' \leftarrow (r', s')$ is a valid signature on m.
We use the same notations as in the previous scheme to define in figure 6 the algorithms of our second group signature scheme. X_{NI} will be the extraction algorithm for the non-interactive Groth-Sahai proofs.

Keygen(λ)	**Verify**(m, μ)
$gk \leftarrow \mathcal{G}(\lambda)$	If $V_{NI}(crs, \mu, m, \pi) = 1 \wedge$
$(crs, ek) \leftarrow K_{NI}(gk)$	$\Sigma_{ots}.\texttt{verify}(vk_{ots}, \sigma_{ots}, a\|\pi) = 1$
$(sk_{cert}, pk_{cert}) \leftarrow \Sigma_{cert}.\texttt{keygen}(gk)$	then return 1
$gpk \leftarrow (gk, pk_{cert}, crs, \textbf{reg})$	Else return 0.
Join(user: usk_i, issuer: sk_{cert}, group manager: **Sreg**)	**Open**(**Sreg**, ek, m, μ)
$(sk_i, vk_i) \leftarrow \Sigma_1.\texttt{Keygen}(gk)$	$vk_i \leftarrow X_{NI}(crs, \mu)$
$\sigma_{cert}(i) \leftarrow \Sigma_{cert}.\texttt{sign}(sk_{cert}, vk_i)$	If vk_i is registered in **Sreg**
$\sigma_i \leftarrow \Sigma_0.\texttt{sign}(usk_i, vk_i)$	return i and τ
Sreg$[i] \leftarrow (vk_i, \sigma_i)$	Else return $\perp$
reg$[i] \leftarrow (i, upk_i)$	with $\tau \leftarrow$ POK$\{(vk_i, \sigma_i)$:
	$\Sigma_1.\texttt{Verify}(vk_i, m\|vk_{ots}\|r', \sigma_1) = 1 \wedge$
Sign$(sk_i, \sigma_{cert}(i), m)$	$\Sigma_0.\texttt{Verify}(upk_i, vk_i, \sigma_i) = 1$
$(r, s) \leftarrow \sigma_{cert}(i)$	$\}(m, \mu, \textbf{reg})$
$(r', state) \leftarrow \texttt{SigPrand}_0(r)$	
$s' \leftarrow \texttt{SigPrand}_1(s, state)$	**Judge**(m, μ, τ, i)
$(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}.\texttt{keygen}(\lambda)$	If $V_{NIZK}(crs, m, \mu, \tau, i) = 1$
$(a, b, c) \leftarrow \Sigma_1.\texttt{sign}(sk_i, m\|vk_{ots}\|r')$	return 1.
$\pi \leftarrow$ POK$\{(vk_i, s')$:	Else return 0.
$(a, b, c) = \Sigma_1.\texttt{sign}(sk_i, m\|vk_{ots}\|r') \wedge$	
$(r', s') = \Sigma_{cert}.\texttt{sign}(sk_{cert}, vk_i)$	
$\}(m, (a, b, c), r')$	
$\sigma_{ots} \leftarrow \Sigma_{ots}.\texttt{sign}(sk_{ots}, a\|\pi)$	
$\mu \leftarrow (m, vk_{ots}, \sigma_{ots}, (a, b, c), r', \pi)$	

Fig. 6. Group signature scheme 2

5.2 Security proofs

Theorem 4. *If Σ_{cert} is a EUF-CMA secure signature scheme and (P, V) is a sound zero-knowledge proof system, then the group signature is fully traceable.*

Theorem 5. *If Σ_0 and Σ_1 are EUF-CMA secure signature schemes, Σ_{ots} is a strong one-time signature scheme and (P, V) is a sound, zero-knowledge proof system, then the group signature scheme is strongly non-frameable.*

Theorem 6. *If Σ_0 and Σ_1 are EUF-CMA secure signature schemes, Σ_{ots} is a strong one time signature and (P, V) is a sound, zero-knowledge proof system, then the group signature scheme is anonymous under the SXDH assumption.*

The proofs are provided in appendix C.

5.3 A concrete realization

We use the Λ_{sxdh} setting and the structure-preserving signature scheme described in Fig. 9 for Σ_{cert}. We further use the weakly secure Boneh-Boyen signature scheme described in Fig 8 for Σ_{ots}.
Using the same notations as in section 4 (with $(\widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^{\alpha_i}, \widetilde{g}^{\beta_i})$) and [1], π is a proof of the following equations:

$$e(a_1, \underline{Y}) = e(b_1, \widetilde{g}) \tag{1}$$

$$e(a_1, \underline{X}).e(b_1, \underline{X})^{\eta} = e(c_1, \widetilde{g}) \tag{2}$$

$$e(g_z, \underline{z'})e(g_r, \underline{r'})e(s', t')e(g_1, \underline{X})e(g_2, \underline{Y}) = A \tag{3}$$

$$e(h_z, \underline{z'})e(h_u, \underline{u'})e(v', w')e(h_1, \underline{X})e(h_2, \underline{Y}) = B \tag{4}$$

where $\sigma_{cert} \leftarrow (z', r', s', t', u', v', w')$ and η is $m \| vk_{ots} \| (s', t', v', w')$. The authors of [1] proved that (s', t', v', w') are information theoretically independant of the signature element z' and (X, Y).
These equations involve then 5 variables (X, Y, z', r', u') and thus increase the size of the proof by 10 group elements. Equations (1) and (2) prove that (a_1, b_1, c_1) is a valid Camenisch-Lysyanskaya signature from user i on η. These two equations costs 2 group elements each. Equation (3) and (4) prove that the certificate is a valid signature on X and Y and cost 2 group elements each. The total size of the proof is then 18 group elements.

A group signature, $(vk_{ots}, \sigma_{ots}, (a, b, c), (s', t', v', w'))$ involves 10 group elements, the total size is thus 28 group elements, which amounts to 900 Bytes if we consider an implementation using 256-bit groups size. Using naive computations, equation (1) involves 22 pairings, equation (2), 27 pairings and equation (3) and (4), 37 pairings each. Verifying the one-time signature adds 1 pairing. The total cost of the `Verify` algorithm is then 124 pairings.

6 A generic construction with encryption

6.1 Description

It is possible, using a tag-based encryption scheme Γ, to modify the first group signature scheme to construct a generic scheme. This modification increases the size of the group signature but the new scheme reaches the full anonymity. We now assume that Σ_1 is an EUF-CMA secure digital signature scheme. The difference with the first scheme is that the signature produced by Σ_1 is now encrypted, using Γ, under the tag vk_{ots}. The group manager owns the encryption secret key and is then able to decrypt the ciphertext and proceeds as in the `Open` algorithm of the first scheme.

We assume that any digital signature σ, generated using Σ_1 on an arbitrary message m can be efficiently transformed in a reversible way to a pair (s, r) where r (r may be the empty string) reveals no information about $\Sigma_1.vk$, i.e. there exists an algorithm that inputs a key from the key space and outputs a string indistinguishable from r. This technical detail will improve the efficiency of the construction as it will not necessitate encrypting the entire signature σ.
The algorithms defining this new group signature are described in Figure 7.

6.2 Security proofs

Theorem 7. *If Γ is an IND-st-wCCA secure tag-based encryption scheme, Σ_{ots} is a strong one-time signature scheme and (P, V) is a sound, zero-knowledge proof system, then the group signature scheme is fully anonymous.*

Theorem 8. *If Σ_{cert} is a EUF-CMA secure signature scheme and (P, V) is a sound zero-knowledge proof system, then the group signature is fully traceable.*

Theorem 9. *If Σ_0 and Σ_1 are EUF-CMA secure signature schemes, Σ_{ots} is a strong one-time signature and (P, V) is a sound, zero-knowledge proof system, then the group signature scheme is strongly non-frameable.*

The proofs are provided in appendix D.

6.3 A concrete realization

We instantiate the construction with the following bricks:

1. The tag-based encryption scheme described in Figure 10 with $G = \mathbb{G}_1$.
2. The signature scheme described in Figure 2 for Σ_1 and the signature scheme described in Figure 3 for Σ_{cert}.

Keygen(λ)	**Verify**(m, μ)
$gk \leftarrow \mathcal{G}(\lambda)$	If $V_{NI}(crs, \mu, m, \pi) = 1 \wedge$
$crs \leftarrow K_{NI}(gk)$	$\Sigma_{ots}.\texttt{verify}(vk_{ots}, \sigma_{ots}, C\|\pi) = 1$
$(sk_{cert}, pk_{cert}) \leftarrow \Sigma_{cert}.\texttt{keygen}(gk)$	then return 1
$(sk_M, pk_M) \leftarrow \Gamma.\texttt{keygen}(\lambda)$	Else return 0.
$gpk \leftarrow (gk, pk_{cert}, pk_M, crs, \mathbf{reg})$	
	Open($\mathbf{Sreg}, sk_M, m, \mu$)
Join(user: usk_i, issuer: sk_{cert}, group manager: **Sreg**)	If $\texttt{Verify}(m, \mu) = 0$
$(sk_i, vk_i) \leftarrow \Sigma_1.\texttt{Keygen}(gk)$	then return 0.
$\sigma_{cert}(i) \leftarrow \Sigma_{cert}.\texttt{comSign}(sk_{cert}, \texttt{Commit}(sk_i), \pi)$	$s \leftarrow \Gamma.\texttt{decrypt}_{sk_M}(vk_{ots}, C)$
$\sigma_i \leftarrow \Sigma_0.\texttt{sign}(uski, vk_i)$	$\sigma_1 \leftarrow (r, s)$
$\mathbf{Sreg}[i] \leftarrow (vk_i, \sigma_i)$	for vk_i in $\mathbf{Sreg}[i]$:
$\mathbf{reg}[i] \leftarrow (i, upk_i)$	if $\Sigma_1.\texttt{Verify}(vk_i, m\|vk_{ots}\|\sigma_{cert}, \sigma_1) = 1$
	return i and a proof τ.
Sign$(sk_i, \sigma_{cert}(i), m)$	with $\tau \leftarrow \text{POK}\{(vk_i, \sigma_i, \sigma_1) :$
$\sigma_{cert}(i) \leftarrow \Sigma_{cert}.\texttt{sigRand}(\sigma_{cert}(i))$	$\Sigma_1.\texttt{Verify}(vk_i, m\|vk_{ots}\|\sigma_{cert}, \sigma_1) = 1 \wedge$
$(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}.\texttt{keygen}(\lambda)$	$\Sigma_0.\texttt{Verify}(upk_i, vk_i, \sigma_i) = 1$
$\sigma_1 \leftarrow \Sigma_1.\texttt{sign}(sk_i, m\|vk_{ots}\|\sigma_{cert}(i))$	$\}(m, \mu, \mathbf{reg})$
$(r, s) \leftarrow \sigma_1$	
$C \leftarrow \Gamma.\texttt{encrypt}_{pk_M}(vk_{ots}, s)$	**Judge**(m, μ, τ, i)
$\pi \leftarrow \text{POK}\{(sk_i, s, l) :$	If $V_{NIZK}(crs, m, \mu, \tau, i) = 1$
$\sigma_1 \leftarrow (r, s)$	return 1.
$\sigma_1 = \Sigma_1.\texttt{sign}(sk_i, m\|vk_{ots}\|\sigma_{cert}(i)) \wedge$	Else return 0.
$\sigma_{cert}(i) = \Sigma_{cert}.\texttt{sign}(sk_{cert}, (l, sk_i))$	
$\}(m, r, C, \sigma_{cert}(i))$	
$\sigma_{ots} \leftarrow \Sigma_{ots}.\texttt{sign}(sk_{ots}, C\|r\|\pi)$	
$\mu \leftarrow (m, vk_{ots}, \sigma_{ots}, r, C, \sigma_{cert}(i), \pi)$	

Fig. 7. Group signature scheme 3

3. The structure-preserving signature scheme described in Fig. 9 for Σ_0.
4. We use the weakly-secure Boneh-Boyen signature scheme descibed in Fig. 8 for Σ_{ots}.

In a group signature μ, $(vk_{ots}, \sigma_{ots}, r, C, \sigma_{cert})$ involve 16 group elements and π is a proof for the following equations (we use the notations of section 4 and $C \leftarrow (C_1, C_2, C_3, C_4)$):

$$(C_1, C_2, C_3) = (g^{\underline{k}}, (X_1^{vk_{ots}} \widetilde{X}_1)^{\underline{k}}, (X_2^{vk_{ots}} \widetilde{X}_2)^{\underline{k}}) \tag{1}$$

$$(e(a_1, \widetilde{g}).e(b_1, \widetilde{g})^{\eta})^{\underline{\alpha_i}} = e(C_4, \widetilde{g}) \ .e(X_1^{-1}, \widetilde{g})^{\underline{k}} \tag{2}$$

$$e(a_1, \widetilde{g})^{\underline{\beta_i}} = e(b_1, \widetilde{g}) \tag{3}$$

$$e(c_1, \widetilde{g}) = e(a, U).e(b, U)^{\underline{l}} \ .e(B_1, U)^{\underline{\alpha_i}} \ .e(B_2, U)^{\underline{\beta_i}} \tag{4}$$

where l is the randomness used by user i in the join stage to obtain signature on the committed value α_i and β_i.
These equations involve 4 variables $(\alpha_i, \beta_i, l, k)$ and thus increase the size of the proof by 8 group elements. Equations (1), (2) and (3) prove that r and the value encrypted in C is a valid Camenisch-Lysyanskaya signature from user i on η. Equation (1) comprises three sub-equations which cost 1 group element each (linear equation). Equations (2) and (3) cost 1 group element each. Equation (5) proves that the certificate is a valid signature on α_i and β_i and costs 1 group element. The total size of the proof is then 14 group elements.

The size of the group signature is thus 30 group elements, which amounts to 1 kB if we consider an implementation using 256-bit groups size, which is more compact that the realization in [19] where the total size of the group signature is 50 elements.
We use naive the computation [8] for the proof verification. Equation (1) involves three sub-equations requiring 16 pairing each. Equation (2) involves 4 pairings + 18 pairings for the proof verification. Equation (3)

requires $2 + 16 = 18$ pairings and equation (4), 25 pairings. The verification cost for these equations is then 81 pairings but the `Verify` algorithm requires 11 more pairings (to check the consistency of the certificate and the one-time signature). The total cost of the `Verify` algorithm is then 92 pairings.

References

1. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, 2010.
2. Laila El Aimani and Marc Joye. Toward practical group encryption. Cryptology ePrint Archive, Report 2012/155, 2012.
3. Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. *IACR Cryptology ePrint Archive*, 2005:385, 2005.
4. Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2004.
5. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
6. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
7. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, 2010.
8. Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch groth-sahai. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *Lecture Notes in Computer Science*, pages 218–235, 2010.
9. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
10. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick Drew McDaniel, editors, *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004.
11. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2006.
12. Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.
13. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.
14. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In Jeffrey Scott Vitter, editor, *STOC*, pages 209–218. ACM, 1998.
15. David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. In Smart [23], pages 127–145.
16. David Chaum. Some weaknesses of "weaknesses of undeniable signatures". In Donald W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 554–556. Springer, 1991.
17. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.
18. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
19. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.
20. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Smart [23], pages 415–432.
21. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *Proceedings of the Sixth Annual International Workshop on Selected Areas in Cryptography 1999*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer.

22. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *J. Cryptology*, 15(2):75–96, 2002.
23. Nigel P. Smart, editor. *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*. Springer, 2008.

A Building Blocks

A.1 Digital signature schemes

A digital signature scheme Σ comprises three algorithms, namely the key generation algorithm `keygen`, the signing algorithm `sign`, and the verification algorithm `verify`. The standard security notion for a signature scheme is existential unforgeability under chosen message attacks (EUF-CMA), which was introduced in [18]. Informally, this notion refers to the hardness of, given a signing oracle, producing a valid pair of message and corresponding signature such that message has not been queried to the signing oracle. There exists also a stronger notion, SEUF-CMA (strong existential unforgeability under chosen message attack), which allows the adversary to produce a forgery on a previously queried message, however the corresponding signature must not be obtained from the signing oracle.

A.2 Boneh-Boyen's weakly secure signature scheme

We recall in Fig. 8 the Boneh-Boyen's weakly secure signature scheme [9], proven secure under the q-SDH (Strong Diffie-Hellman) assumption:

The q-SDH assumption: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q, respectively generated by g_1 and g_2. The q-SDH problem is stated as follows:

$$\text{Given as input a } (q+3)\text{-tuple of elements } (g_1, g_1^x, g_1^{x^2}, ..., g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2,$$
$$\text{output a pair } (c, g_1^{\frac{1}{x+c}}) \in \mathbb{Z}_q \times \mathbb{G}_1 \text{ for a freely chosen value } c \in \mathbb{Z}_q \setminus \{-x\}.$$

We say that the (q, ϵ)-SDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$ if no polynomial-time algorithm solves the q-SDH problem with probability greater than ϵ.

[**Key generation**]	Select random generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. Choose $x \xleftarrow{R} \mathbb{Z}_q$ then compute $v \leftarrow g_2^x$ and $z \leftarrow\in e(g_1, g_2)$. set $pk \leftarrow \{g_1, g_2, v, z\}$ and $sk \leftarrow \{x\}$.
[**Signature on** m]	Return $\sigma \leftarrow g^{\frac{1}{x+m}}$
[**Verification**]	Given pk, m and σ, check if the following verification equation holds: $e(\sigma, v.g_2^m) = z$. If the equality holds, or if $\sigma = 1$ and $v.g_2^m = 1$, return 1.

Fig. 8. Boneh-Boyen weakly secure signature scheme

A.3 Structure preserving signature

We recall in Fig. 9 the digital signature scheme proposed by Abe *et al* [1], allowing to sign messages of n group elements. The scheme was proved EUF-CMA secure under the q-SFP (Simultaneous Flexible Pairing) assumption:

The q-SFP assumption: Let g_z, h_z, g_r and h_u be random generators of $\mathbb{G}_1$. Let $(a, \widetilde{a}), (b, \widetilde{b})$ be random pairs in $\mathbb{G}_1 \times \mathbb{G}_2$. For $j = 1, .., q$, let $R_j = (z, r, s, tu, v, w)$ that satisfies:

$$e(a,\widetilde{a}) = e(g_z,z)e(g_r,r)e(s,t) \text{ and } e(b,\widetilde{b}) = e(h_z,z)e(h_u,u)e(v,w)$$

Given $g_z, h_z, g_r, h_u, (a,\widetilde{a}), (b,\widetilde{b})$ and $R_1, ..., R_q$ it is hard to find $(z^*, r^*, s^*, t^*, u^*, v^*, w^*)$ that fulfill the previous relations under the restriction that $z^* \neq 1$ and $z^* \neq z \in R_j$ for $1 \leq j \leq q$.

[**Key generation**]	$g, g_r, g_u \xleftarrow{R} \mathbb{G}_1$ and $\widetilde{g} \xleftarrow{R} \mathbb{G}_2$.
	$\gamma_i, \delta_i \xleftarrow{R} \mathbb{Z}_q$ and $g_i \leftarrow g_r^{\gamma_i}$, $h_i \leftarrow h_u^{\delta_i}$ for $i = 1, .., k$.
	$\gamma_z, \delta_z \xleftarrow{R} \mathbb{Z}_q$ and $g_z \leftarrow g_r^{\gamma_z}$, $h_z \leftarrow h_u^{\delta_z}$.
	$\alpha, \beta \xleftarrow{R} \mathbb{Z}_q$ and $A \leftarrow e(g_r, \widetilde{g}^{\alpha}), B \leftarrow e(h_u, \widetilde{g}^{\beta})$
	$pk \leftarrow (g_z, h_z, g_r, h_u, \{g_i, h_i\}_{i=1}^k, A, B)$
	$sk \leftarrow (\alpha, \beta, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^k)$.
[**Signature on** $(m_1, ..., m_k)$]	$\zeta, \rho, \tau, \phi, \omega \xleftarrow{R} \mathbb{Z}_q$.
	$z \leftarrow \widetilde{g}^{\zeta}, r \leftarrow \widetilde{g}^{\alpha-\rho\tau-\gamma_z\zeta} \prod_{i=1}^k m_i^{-\gamma_i}$, $s \leftarrow g_r^{\rho}$, $t \leftarrow \widetilde{g}^{\tau}$
	$u \leftarrow \widetilde{g}^{\beta-\phi\omega-\delta_z\zeta} \prod_{i=1}^k m_i^{-\delta_i}$, $v \leftarrow h_u^{\phi}$, $w \leftarrow \widetilde{g}^{\omega}$
	Return $\sigma \leftarrow (z, r, s, t, u, v, w)$
[**Verification**]	Check if the following verification equations hold:
	$A = e(g_z, z)e(g_r, r)e(s, t) \prod_{i=1}^k e(g_i, m_i)$.
	$B = e(h_z, z)e(h_u, u)e(v, w) \prod_{i=1}^k e(h_i, m_i)$.

Fig. 9. Structure preserving signature

A.4 Strong one-time signatures

Strong one-time digital signature schemes can be used to sign, at most, one message; otherwise, signatures can be forged. A new public key is required for each message that is signed. They are, similarly to normal digital signatures, defined by the key generation algorithm, the signing algorithm, and the verification algorithm. We say the one-time signature is strong, if the adversary can neither forge a signature on different message nor create a different signature on the message already signed.

A.5 Tag-based encryption (TBE)

Tag-based encryption, also referred to as encryption with labels, was first introduced in [22]. In these schemes, the encryption algorithm takes as input, in addition to the public key pk and the message m intended to be encrypted, a tag t which specifies information related to the message m and its encryption context. Similarly, the decryption algorithm takes additionally to the ciphertext and the private key the tag under which the ciphertext was created. Security notions are then defined as usual except that the adversary specifies to his challenger the tag to be used in the challenge ciphertext, and in case he (the adversary) is allowed to query oracles, then he cannot query them on the pair formed by the challenge ciphertext and the tag used to form it. There are also weakened security models for this type of encryption where the adversary specifies the challenge tag before getting the parameters of the scheme, and during the game, he (the adversary) is not allowed to query the allowed oracles w.r.t. the challenge tag; we talk in this case about selective tag security. We specify in the following the formal definition of IND-st-wCCA security for tag-based encryption:

Let Γ be a tag-based encryption scheme scheme. Let further $\mathcal{A}$ denote a probabilistic polynomial time algorithm. We write $\mathcal{A}^{\mathtt{decrypt}(sk,.)}$ to denote that $\mathcal{A}$ has access to oracle $\mathtt{decrypt}(sk, .)$. When a query q is not allowed, we use the symbol $\neg$: $\mathcal{A}^{\mathtt{decrypt}^{\neg q}(sk,.)}$. We define the experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{IND-st-wCCA}}(\lambda)$:

1. $param \leftarrow \mathtt{setup}(\lambda)$;
2. $t \leftarrow \mathcal{A}(param)$;
3. $(sk, pk) \leftarrow \Gamma.\mathtt{keygen}(param, \lambda)$;

4. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk,.)}(pk)$;
5. $b \xleftarrow{R} \{0,1\}$; $e_b \leftarrow \Gamma.\mathtt{encrypt}_{pk}(m_b, t)$;
6. $b^\star \leftarrow \mathcal{A}^{\mathtt{decrypt}^{\neg(-,t)}(sk,.)}(e_b)$;
7. If $b = b^\star$ return 1 else 0.

We define $\mathcal{A}$'s advantage as $\mathbf{Adv}^{\mathsf{ind\text{-}st\text{-}wCCA}}_{\Gamma,\mathcal{A}}(\lambda) = \left|\Pr\left[\mathbf{Exp}^{\mathsf{ind\text{-}st\text{-}wCCA}}_{\Gamma,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$.
Given $(t, q_d) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a (t, ε, q_d)-ind-st-wCCA adversary against Γ if, running in time t and issuing q_d decryption queries, $\mathcal{A}$ has $\mathbf{Adv}^{\mathsf{ind-st-wCCA}}_{\Gamma,\mathcal{A}}(\lambda) \geq \varepsilon$. The scheme Γ is said to be (t, ε, q_d)-ind-st-wCCA secure if no (t, ε, q_d)-ind-st-wCCA adversary against it exists.

The Twin Diffie-Hellman problem was introduced in [15] as a slight modification to the original Diffie-Hellman (DH) problem which is as hard as the ordinary DH problem, *even given access to a corresponding decision oracle*. It is defined as follows.

Let $\mathcal{G}$ be a prime-order cyclic group generated by some g. Let dh further denotes the Diffie-Hellman function (it inputs $(X, Y) \in \mathcal{G}^2$ and returns Z such that (g, X, Y, Z) is Diffie-Hellman quadruple). The *Twin DH function is defined as follows:*

$$\begin{aligned} \mathrm{2dh}\colon \; & \mathcal{G}^3 \to \mathcal{G}^2 \\ & (X_1, X_2, Y) \to (\mathrm{dh}(X_1, Y), \mathrm{dh}(X_2, Y)) \end{aligned}$$

Cash *et al.* define further a corresponding *twin DH predicate*:

$$\mathrm{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) := \mathrm{2dh}(X_1, X_2, \hat{Y}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2).$$

Finally the *Strong Twin Diffie-Hellman* assumption states that distinguishing the two distributions $(X_1, X_2, Y, \mathrm{dh}(X_1, Y))$ and (X_1, X_2, Y, Z) for random $X_1, X_2, Y, Z \in \mathcal{G}$ is hard even in the presence of a decision oracle for the predicate $\mathrm{2dhp}(X_1, X_2, -, -, -)$ which on input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2))$ returns $\mathrm{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

Theorem 10. *The DDH assumption holds if and only if the Strong Twin DDH assumptions holds.*

Proof. The proof is given in [15].

[Setup]	Choose a group $(\mathbb{G}, \cdot)$ generated by g with prime order d.
[Key generation]	Choose $x_1, \widetilde{x}_1, x_2, \widetilde{x}_2 \xleftarrow{R} \mathbb{Z}_d$ then compute $X_i \leftarrow g^{x_i}$ and $\widetilde{X}_i \leftarrow g^{\widetilde{x}_i}$ for $i = 1, 2$ set $pk \leftarrow \{X_i, \widetilde{X}_i\}_{i=1,2}$ and $sk \leftarrow \{x_i, \widetilde{x}_i\}_{i=1,2}$.
[Encryption]	For a message $m \in \mathbb{G}$ and a tag $t \in \mathbb{Z}_d$: choose $r \xleftarrow{R} \mathbb{Z}_d$, compute $c_1 \leftarrow g^r$, $c_2 \leftarrow (X_1^t \widetilde{X}_1)^r$, $c_3 \leftarrow (X_2^t \widetilde{X}_2)^r$, and $c_4 = m X_1^r$, set the ciphertext to (c_1, c_2, c_3, c_4).
[Decryption]	Given a ciphertext $c = (c_1, c_2, c_3, c_4)$ and a tag t: check that $c_2 = c_1^{tx_1 + \widetilde{x}_1}$ and that $c_3 = c_1^{tx_2 + \widetilde{x}_2}$ if it is not the case, return $\perp$, otherwise: compute the plaintext as $m \leftarrow c_4 c_1^{-x_1}$.

Fig. 10. TBE variant of the Modified Cramer-Shoup [15]

Theorem 11. *The encryption scheme in Figure 10 is IND-st-wCCA secure under the Strong Twin Diffie-Hellman assumption.*

Proof. The proof is given in [2].

A.6 Non-interactive proofs of knowledge

Let R be an efficiently computable ternary relation. Given some gk we let L be the language consisting of statements x that have a witness w so $(gk, x, w) \in R$.
As in [20], such a system is modeled by four probabilistic polynomial time algorithms: a setup algorithm $\mathcal{G}_{NI}$, a common reference string (crs) generation algorithm K_{NI}, a prover P_{NI} and a verifier V_{NI}. The setup outputs a setup (gk, ek) (ek, the trapdoor, may be the empty string) that K_{NI} takes as input to produce the common reference string crs. The prover P_{NI} takes as input (gk, crs, x, w) and produces a proof π. The verifier takes as inputs (gk, crs, x, π) and outputs 1 if the proof is valid and 0 if rejecting the proof.
In the following, we use the notation $POK\{(w) : (gk, x, w) \in R\}(gk, crs, x)$ for $P_{NI}(gk, crs, x, w)$.

A non-interactive proof should satisfy completeness which denotes that an honest prover is able to convince an honest verifier whenever the statement belongs to the language and the prover holds a witness testifying to this fact. A further required property is soundness which captures the inability of a cheating prover P to convince the verifier V with an invalid statement.

Informally, a zero-knowledge proof is a proof that shows the statement is true without revealing anything else. This is defined by having a simulator (S_1, S_2) whose first part, S_1, outputs a simulated crs and a simulation trapdoor t and whose second part, S_2, simulates, using t, proofs for statements without knowing the corresponding witnesses. Real proofs on a real crs have to be computationally indistinguishable from simulated proofs on a simulated crs.

The Groth-Sahai (GS) proof system [20] gives efficient non-interactive zero knowledge (NIZK) and non-interactive witness-indistinguishable (NIWI) proofs for algebraic statements involving elements in groups equipped with a bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \leftarrow \mathbb{G}_T$ (we assume that), e.g.

A pairing-product equation over variables $X_1, \ldots, X_m \in \mathbb{G}_1$ and $Y_1, \ldots, Y_n \in \mathbb{G}_2$

$$\prod_{i=1}^{n} e(A_i, Y_i) \cdot \prod_{i=1}^{m} e(X_i, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(X_i, Y_i)^{\gamma_{i,j}} = t_T,$$

for constants $A_1, \ldots, A_n \in \mathbb{G}_1$, $B_1, \ldots, B_m) \in \mathbb{G}_1$, and $\Gamma = (\gamma_{i,j})_{\substack{1 \le i \le m \\ 1 \le j \le n}} \in \mathbb{Z}_d^{m \times n}$, and $t_T \in \mathbb{G}_T$.

A multi-scalar multiplication over variables $y_1, \ldots, y_m \in \mathbb{Z}_d$ and $X_1, \ldots, X_m \in \mathbb{G}_1$

$$\prod_{i=1}^{n} A_i^{y_i} \cdot \prod_{i=1}^{m} X_i^{b_i} \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} X_j^{y_i \gamma_{i,j}} = T,$$

for constants $A_1, \ldots, A_n \in \mathbb{G}_1$, $b_1, \ldots, b_m \in \mathbb{Z}_d$, $\Gamma \in \mathbb{Z}_d^{m \times n}$, and $T \in \mathbb{G}_1$. A multi-scalar multiplication equation over variables in $\mathbb{G}_2$ is defined analogously.

A quadratic equation over variables $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ in $\mathbb{Z}_d$

$$\sum_{i=1}^{n} a_i y_i + \sum_{i=1}^{m} x_i b_i + \sum_{i=1}^{m}\sum_{j=1}^{n} \gamma_{i,j} x_i y_j = t,$$

For constants $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ in $\mathbb{Z}_d$, constant matrix $\Gamma \in \mathbb{Z}_d^{m \times n}$, and $t \in \mathbb{Z}_d$.

The principle of the GS proof system consists in using the setup parameters and the common reference string (CRS) to commit to the witness components, then generate proof elements that these values committed to satisfy the equations underlying the statement to be proved. Note that proofs for pairing-product equations are only (WI) when $t_T \in \mathbb{G}_T$ has no particular structure. They can be however transformed into ZK proofs (at the expense of an additional cost) if t_T is equal to $1_{\mathbb{G}_T}$ or its representation as a pairing product equation is known.

The GS proof system could be instantiated under different (mild) assumptions, in particular the Λ_{sxdh} setting which refers to the case of asymmetric pairings for which the DDH assumptions holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{Z}_d$
Variables $x \in \mathbb{Z}_d, X \in \mathbb{G}_1$	2	0	0
Variables $y \in \mathbb{Z}_d, Y \in \mathbb{G}_2$	0	2	0
Pairing product equations	4	4	0
- Linear equations: $\prod_{i=1}^{n} e(A_i, Y_i) = t_T$	2	0	0
- Linear equations: $\prod_{i=1}^{n} e(X_i, B_i) = t_T$	0	2	0
Multi-scalar multiplication equations in $\mathbb{G}_1$	2	4	0
- Linear equations: $\prod_{i=1}^{n} A_i^{y_i} = T_1$	1	0	0
- Linear equations: $\prod_{i=1}^{n} X_i^{b_i} = T_1$	0	0	2
Multi-scalar multiplication equations in $\mathbb{G}_2$	4	2	0
- Linear equations: $\prod_{i=1}^{n} Y_i^{a_i} = T_2$	0	0	2
- Linear equations: $\prod_{i=1}^{n} B_i^{x_i} = T_2$	0	1	0
Quadratic equations in $\mathbb{Z}_d$	2	2	0
- Linear equations: $\sum_{i=1}^{n} a_i y_i = t$	0	0	1
- Linear equations: $\sum_{i=1}^{m} x_i b_i = t$	0	0	1

Fig. 11. Cost of each variable and equation in terms of group elements from $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{Z}_d$

	Naive computation	Batch computation
Pairing product equations	$5m + 3n + 16$	$m + 2n + 8$
Multi-scalar multiplication equations in $\mathbb{G}_1$	$8m + 2n + 14$	$\min(2n + 9, 2m + n + 7)$
Multi-scalar multiplication equations in $\mathbb{G}_2$	$8n + 2m + 14$	$\min(2m + 9, 2n + m + 7)$
Quadratic equations in $\mathbb{Z}_d$	$8m + 8n + 12$	$2\min(m, n) + 8$

Fig. 12. Number of pairings per proof verification (m and n refer to the number of different types of involved variables - see [8])

B Security proofs of section 4

B.1 Proof of theorem 1

Proof. Let $\mathcal{A}$ be an adversary against the traceability with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ that EUF-CMA breaks the underlying digital signature scheme Σ_{cert}.
[`keygen`] $\mathcal{R}$ runs the `Keygen` algorithm as usual, except that she does not run Σ_{cert}.`keygen` and gets the issuer public key pk_{cert} from her EUF-CMA challenger.
[**`opening` queries**] $\mathcal{R}$ proceeds as the standard algorithm.
[**`joining` queries**] When $\mathcal{R}$ receives a joining query, she uses her signing oracle to produce the user's certificate. Then, she proceeds as the standard algorithm to simulate the interaction between the group manager and the new user.
[**Final output**] $\mathcal{A}$ returns an untraceable group signature μ on a message m, which contains a certificate σ_{cert} on sk, the secret key used to sign with Σ_1, such that $sk \neq sk_i$ for all registered user i (else, the group signature is traceable). Using the assumption that (P, V) is a sound proof system, $\mathcal{A}$ has to forge a new certificate on sk, and $\mathcal{R}$ will run the extractor on the NIZK proof π to get sk and returns the pair (sk, σ_{cert}) to her challenger.
$\mathcal{R}$ EUF-CMA breaks the digital signature scheme Σ_{cert} with advantage ϵ.

B.2 Proof of theorem 2

Proof. Let $\mathcal{A}$ be an adversary against the non-frameability with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ that EUF-CMA breaks the underlying digital signature scheme Σ_1.

[`keygen`] $\mathcal{R}$ runs the `Keygen` algorithm as usual, except that she computes a simulated common reference string crs. $\mathcal{R}$ chooses at random $i \in \mathcal{HU}$ and computes $(upk_j, usk_j) \leftarrow \Sigma_0$.`keygen` for $j \in \mathcal{HU}$.

[`joining` **queries**] $\mathcal{R}$ has to simulate the user part of the join protocol with the group manager (simulated by $\mathcal{A}$). If $j \neq i$, $\mathcal{R}$ generates a key pair $(pk_j, sk_j) \leftarrow \Sigma_1.\texttt{keygen}$ and proceeds as in the standard algorithm. If $j = i$ then $\mathcal{R}$ gets the verification key from her challenger and uses the simulated common reference string to produce the proofs of knowledge.
[`signing` **queries**] $\mathcal{R}$ proceeds as the standard algorithm if $j \neq i$, else, she uses her signing oracle to produce a signature and uses the simulated common reference string to produce the proofs of knowledge.
[**Final output**] $\mathcal{A}$ returns a group signature $\mu = (\sigma_{ots}, vk_{ots}, \sigma_1, \sigma_{cert}, \pi)$ on a message m and an identity j. If $j \neq i$, then $\mathcal{R}$ aborts the experiment. Otherwise we proceed as follows We first assume that σ_1 is a valid signature $w.r.t\ vk_{i_b}$. We distinguish the following two cases:

1. $\mathcal{O}\texttt{Sign}$ oracle was already requested on (i, m), returning μ^*. Then there are two possibilities:
   - vk_{ots} or σ_{cert} are different from those in μ^*, which implies that $\mathcal{A}$ successfully forge a new signature σ_1. Then, $\mathcal{R}$ forwards $(m\|vk_{ots}\|\sigma_{cert}, \sigma_1)$ to her challenger.
   - σ_1 or π are different from those in μ^*, then σ_1 or π have been re-randomized and $\mathcal{A}$ has to forge a new one-time signature Σ_{ots}. Then $\mathcal{A}$ can be turned in an attacker against the security of Σ_{ots}.
2. There were no oracle request on (i, m), then σ_1 is a forged signature on $m\|vk_{ots}\|\sigma_{cert}$.

If σ_1 is not a valid signature $w.r.t\ vk_{i_b}$ then $\mathcal{A}$ has produced a signature σ_1 on $m\|vk_{ots}\|t$ and a verification key vk such that $\Sigma_1.\texttt{Verify}(vk, m\|vk_{ots}\|t, \sigma_1) = 1$ and wants to prove in the `Judge` algorithm that vk is the verification key for user i. Then he has to prove knowledge of a signature $\sigma_{0,i}$ on vk such that $\Sigma_0.\texttt{Verify}(upk_i, vk, \sigma_{0,i}) = 1$. Because of the soundness of the proof, it means that the adversary has produced a forgery on Σ_0 and then, can be turned in an attacker against the EUF-CMA security of Σ_0.

Let m be the number of honest users, $\mathcal{R}$ is able to break the EUF-CMA security of Σ_1 with advantage $\frac{1}{m}\epsilon$. $\mathcal{A}$'s advantage ϵ is then negligible.

B.3 Proof of theorem 3

Proof. Let $\mathcal{A}$ be an adversary against the anonymity with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ against the DDH problem in $\mathbb{G}_1$.

$\mathcal{R}$ gets the DDH challenge (g, g^x, g^y, g^z) from her challenger and has to decide whether $z = xy$ or random.

[`keygen`] $\mathcal{R}$ runs $\mathcal{K}.\texttt{keygen}$ algorithm as usual and generates a pair (α_i, β_i) , for each honest user i. In the experiment $\mathcal{R}$ will act as if $sk_i = (\alpha_i, y\beta_i)$, then has to generate a certificate $\sigma_{cert}(i)$ on it. Let (u, v, z_1, z_2) be the secret certificate key, $\mathcal{R}$ proceeds as follows:

$$(a, A_1, A_2, b, B_1, B_2, c) \leftarrow (g^\rho, g^{\rho z_1}, g^{\rho z_2}, g^v, g^{\rho v z_1}, g^{\rho v z_2}, a^{u+uvr} A_1^{uv\alpha_i} (g^y)^{\rho z_2 uv\beta_i})$$

which is a valid signature on $(\alpha_i, y\beta_i)$ (for random ρ and r).
[`joining` **queries**] $\mathcal{R}$ proceeds as the standard algorithm.
[`signing` **queries**] Receiving a request $\mathcal{O}\textbf{Sign}(i, m)$ from $\mathcal{A}$, $\mathcal{R}$ proceeds as follows:

1. $(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}.\texttt{keygen}$
2. $\sigma \leftarrow \Sigma_{cert}.\texttt{sigRand}(\sigma_{cert}(i))$
3. $\eta \leftarrow m\|vk_{ots}\|\sigma$
4. $(a, b, c) \leftarrow (g^\rho, (g^y)^{\rho\beta_i}, a_i^\alpha (g^y)^{\rho\alpha_i\beta_i\eta})$
5. $\mathcal{R}$ simulates the proof π.
6. $\sigma_{ots} \leftarrow \Sigma_{ots}.\texttt{sign}(sk_{ots}, a\|\pi)$
7. $\mu \leftarrow (vk_{ots}, \sigma_{ots}, (a, b, c), \sigma, \pi)$

for a random ρ. The produced signature (a, b, c) is valid and $\mathcal{R}$ records (i, m, μ) then returns it to $\mathcal{A}$.
[`open` **queries**] $\mathcal{R}$ distinguishes two cases:

- If the signature originates from a dishonest user, $\mathcal{R}$ proceeds as the standard algorithm.
- In case the signature was created on the behalf of an honest user then, because of the non-frameability property of our scheme, the signature is a previous output of the $\mathcal{O}\texttt{Sign}$ oracle. $\mathcal{R}$ looks in her record and outputs the corresponding identity i with a simulated proof π'.

[**Challenge**] $\mathcal{A}$ sends m, i_0, i_1 to $\mathcal{R}$ who computes $b \leftarrow \{0,1\}$. To produce a signature on behalf of user i_b, $\mathcal{R}$ proceeds as follows:

1. $(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}.\texttt{keygen}$
2. $\sigma \leftarrow \Sigma_{cert}.\texttt{sigRand}(\sigma_{cert}(i_b))$
3. $(a, b, c) \leftarrow (g^x, (g^z)^{\beta_{i_b}}, a^{\alpha_{i_b}}(g^z)^{\alpha_{i_b}\beta_{i_b}}\eta)$
4. $\mathcal{R}$ simulates the proof π.
5. $\sigma_{ots} \leftarrow \Sigma_{ots}.\texttt{sign}(sk_{ots}, a\|\pi)$
6. $\mu \leftarrow (vk_{ots}, \sigma_{ots}, (a, b, c), \sigma, \pi)$

[**Post challenge phase**] $\mathcal{R}$ answers signing, joining and opening queries as previously.
[**Final output**] Finally, $\mathcal{A}$ outputs a bit b_0. If $b = b_0$ $\mathcal{R}$ returns 1, else she returns 0.

If (g, g^x, g^y, g^z) is a DDH tuple, the group signature is valid, else μ is an element indistinguishable from a random element in the group signature space since z is random, the certificate is generated with the re-randomizable signature scheme Σ_{cert} and the proof is zero-knowledge. Let b^* be the bit computed by the challenger, then $\mathcal{R}$'s advantage in breaking the DDH problem is:

$$\begin{aligned}\texttt{Adv}(\mathcal{R}) &= |\texttt{Pr}[b = b_0 \mid b^* = 1].\texttt{Pr}[b^* = 1] + \texttt{Pr}[b \neq b_0 \mid b^* = 0].\texttt{Pr}[b^* = 0] - \frac{1}{2}| \\ &= |\frac{1}{4} + \frac{\epsilon}{2} + \frac{1}{4} - \frac{1}{2}| \\ &= \frac{\epsilon}{2}\end{aligned}$$

C Security proofs of section 5

C.1 Proof of theorem 4

Proof. Let $\mathcal{A}$ be an adversary against the traceability with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ that EUF-CMA breaks the underlying digital signature scheme Σ_{cert}.

[`keygen`] $\mathcal{R}$ runs the `Keygen` algorithm as usual, except that she does not run $\Sigma_{cert}.\texttt{keygen}$ and gets the issuer public key pk_{cert} from her EUF-CMA challenger.
[`opening` **queries**] $\mathcal{R}$ proceeds as the standard algorithm.
[`joining` **queries**] When $\mathcal{R}$ receives a joining query, she uses his signing oracle to produce the user's certificate. Then, she proceeds as the standard algorithm to simulate the interaction between the group manager and the new user.
[**Final output**] $\mathcal{A}$ returns an untraceable group signature μ on a message m, which contains a proof of knowledge of a certificate σ_{cert} on vk, the verification key corresponding to the signature produced by Σ_1, such that $vk \neq vk_i$ for all registered user i (else, the group signature is traceable). Using the assumption that (P, V) is a sound proof system, $\mathcal{A}$ has to forge a new certificate on vk, and $\mathcal{R}$ will run the extractor on the NIZK proof π to get vk and σ_{cert} and returns the pair (vk, σ_{cert}) to her challenger.
$\mathcal{R}$ EUF-CMA breaks the digital signature scheme Σ_{cert} with advantage ϵ.

C.2 Proof of theorem 5

Proof. Let $\mathcal{A}$ be an adversary against the non-frameability with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ that EUF-CMA breaks the underlying digital signature scheme Σ_1.

[`keygen`] $\mathcal{R}$ runs the `Keygen` algorithm as usual, except that she chooses $i \in \mathcal{HU}$ and computes $(upk_j, usk_j) \leftarrow \Sigma_0.\texttt{keygen}$ for $j \in \mathcal{HU}$.
[`joining` **queries**] $\mathcal{R}$ has to simulate the user part of the join protocol with the group manager (simulated by $\mathcal{A}$). If $j \neq i$, $\mathcal{R}$ generates a key pair $(pk_j, sk_j) \leftarrow \Sigma_1.\texttt{keygen}$ and proceeds as in the standard algorithm. If $j = i$ then $\mathcal{R}$ gets the verification key vk from her challenger, signs it with usk_i, and sends vk and the

signature to $\mathcal{A}$.
[**signing queries**] $\mathcal{R}$ proceeds as the standard algorithm if $j \neq i$, else, she uses her signing oracle to produce the signature.
[**Final output**] $\mathcal{A}$ returns a group signature $\mu = (\sigma_{ots}, vk_{ots}, \sigma_1, r, \pi)$ on a message m and an identity j. If $j \neq i$, then $\mathcal{R}$ aborts the experiment. Else we assume that σ_1 is a valid signature *w.r.t* vk (else, as in the previous scheme, it involves a forgery on Σ_0). We distinguish the two following cases:

1. $\mathcal{O}$`Sign` oracle was already requested on (i, m), returning μ^*. Then there are two possibilities:
   - vk_{ots} or r are different from those in μ^*, which implies that $\mathcal{A}$ successfully forge a new signature σ_1. Then, $\mathcal{R}$ forwards $(m\|vk_{ots}\|r, \sigma_1)$ to her challenger.
   - σ_1 or π are different from the one in μ^*, then σ_1 or π has been re-randomized and $\mathcal{A}$ has to forge a new one-time signature Σ_{ots}. Then $\mathcal{A}$ can be turned in an attacker against the security of Σ_{ots}.
2. There were no oracle request on (i, m), then σ_1 is a forged signature on $m\|vk_{ots}\|r$.

Let m be the number of honest users, $\mathcal{R}$ is able to break the EUF-CMA security of Σ_1 with advantage $\frac{1}{m}\epsilon$. $\mathcal{A}$'s advantage ϵ is then negligible.

C.3 Proof of theorem 6

Proof. Let $\mathcal{A}$ be an adversary against the anonymity with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ against the DDH problem.

$\mathcal{R}$ gets the DDH challenge (g, g^x, g^y, g^z) from her challenger and has to decide whether $z = xy$ or random.

[`keygen`] $\mathcal{R}$ runs $\mathcal{K}$.`keygen` algorithm as usual, generates a pair (α_i, β_i) and computes $r_i \leftarrow$ `Simulate`(sk_{cert}) for each honest user i. In the experiment $\mathcal{R}$ will act as if $sk_i = (\alpha_i, y\beta_i)$.
[`joining` **queries**] $\mathcal{R}$ proceeds as the standard algorithm.
[`signing` **queries**] Receiving a request $\mathcal{O}$**Sign**(i, m) from $\mathcal{A}$, $\mathcal{R}$ proceeds as follows:

1. $(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}$.`keygen`
2. $r \leftarrow$ `Simulate`(sk_{cert})
3. $\eta \leftarrow m\|vk_{ots}\|r$
4. $(a, b, c) \leftarrow (g^\rho, (g^y)^{\rho\beta_i}, a_i^\alpha (g^y)^{\rho\alpha_i\beta_i\eta})$
5. $\mathcal{R}$ simulates the proof π.
6. $\sigma_{ots} \leftarrow \Sigma_{ots}$.`sign`$(sk_{ots}, a\|\pi)$
7. $\mu \leftarrow (vk_{ots}, \sigma_{ots}, (a, b, c), r, \pi)$

for a random ρ. The produced signature (a, b, c) is valid and $\mathcal{R}$ records (i, m, μ) then returns it to $\mathcal{A}$.
[`open` **queries**] $\mathcal{R}$ distinguishes two cases:

- If the signature originate from a dishonest user, $\mathcal{R}$ proceeds as the standard algorithm.
- In case the signature was created on the behalf of an honest user then, because of the non-frameability property of our scheme, the signature is a previous output of the $\mathcal{O}$`Sign` oracle. $\mathcal{R}$ looks in her record and outputs the corresponding identity i with a simulated proof π'.

[**Challenge**] $\mathcal{A}$ sends m, i_0, i_1 to $\mathcal{R}$ who computes $b \leftarrow \{0, 1\}$. To produce a signature on behalf of user i_b, $\mathcal{R}$ proceeds as follows:

1. $(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}$.`keygen`
2. $\sigma \leftarrow$ `Simulate`(sk_{cert})
3. $\eta \leftarrow m\|vk_{ots}\|r$
4. $(a, b, c) \leftarrow (g^x, (g^z)^{\beta_i}, a^{\alpha_i} (g^z)^{\alpha_i\beta_i\eta})$
5. $\mathcal{R}$ simulates the proof π.
6. $\sigma_{ots} \leftarrow \Sigma_{ots}$.`sign`$(sk_{ots}, a\|\pi)$
7. $\mu \leftarrow (vk_{ots}, \sigma_{ots}, (a, b, c), r, \pi)$

then simulates the proof π'.
[**Post challenge phase**] $\mathcal{R}$ answers signing, joining and opening queries as previously.
[**Final output**] Finally, $\mathcal{A}$ outputs a bit b_0. If $b = b_0$ $\mathcal{R}$ returns 1, else she returns 0.

If (g, g^x, g^y, g^z) is a DDH tuple, the group signature is valid, else μ is an element indistinguishable from a random element in the group signature space since z is random and r is information theoretically independent from σ_{cert} and m. Let b^* be the bit computed by the challenger, then $\mathcal{R}$'s advantage in breaking the DDH problem is:

$$\begin{aligned}\texttt{Adv}(\mathcal{R}) &= |\texttt{Pr}[b = b_0 \mid b^* = 1].\texttt{Pr}[b^* = 1] + \texttt{Pr}[b \neq b_0 \mid b^* = 0].\texttt{Pr}[b^* = 0] - \frac{1}{2}| \\ &= |\frac{1}{4} + \frac{\epsilon}{2} + \frac{1}{4} - \frac{1}{2}| \\ &= \frac{\epsilon}{2}\end{aligned}$$

D Security proofs of section 7

D.1 Proof of theorem 9

Proof. Let $\mathcal{A}$ be an adversary against the anonymity with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ against the IND-st-wCCA security of Γ.

[`keygen`] $\mathcal{R}$ gets the parameters of Γ, computes $(sk_{ots}, vk_{ots}) \leftarrow \Sigma_{ots}.\texttt{keygen}$ and sends $t^* \leftarrow vk_{ots}$ to his challenger. Then, she receives pk, the public key of Γ, and runs $\mathcal{K}.\texttt{keygen}$ algorithm as usual, except that she sets $pk_M = pk$ and computes a simulated common reference string.
[{`signing`, `joining`} **queries**] $\mathcal{R}$ proceeds as the standard algorithms.
[`opening` **queries**] $\mathcal{R}$ runs `Verify` on the group signature. If it outputs 1, then she checks the tag t used for encryption. If $t \neq t^*$, $\mathcal{R}$ uses her decryption oracle on C and then proceeds as the standard `Open` algorithm, else, she returns $\perp$. This simulation differs if $t^* = t$, but this event appears for each query with probability $\frac{1}{|\mathcal{T}|}$ (where $\mathcal{T}$ is the tag space) because $\mathcal{A}$ does not know t^*.
[**Challenge**] $\mathcal{A}$ sends m, i_0, i_1 to $\mathcal{R}$ who computes $b_0 \xleftarrow{R} \{0,1\}$, $\sigma_{cert} \leftarrow \Sigma_{cert}.\texttt{sigRand}(\sigma_{cert}(i_{b_0})$, $\sigma_b \leftarrow \Sigma_1.\texttt{sign}(sk_{i_b}, m \| t^* \| \sigma_{cert})$ and $(r_b, s_b) \leftarrow \sigma_b$ for $b \in \{0,1\}$. She sends s_0 and s_1 to her challenger who computes $b^* \xleftarrow{R} \{0,1\}$ and returns C^*, an encryption of s_{b^*} under the tag t^*. $\mathcal{R}$ sets $r \leftarrow r_{b_0}$ and uses the simulated common reference string to produce the proof π, then computes $\sigma_{ots} \leftarrow \Sigma_{ots}.\texttt{sign}(sk_{ots}, C^* \| r \| \pi)$ and sends $\mu \leftarrow (m, t^*, \sigma_{ots}, r, C^*, \sigma_{cert}, \pi)$ to $\mathcal{A}$.
Recalling the properties of Σ_{cert} and Σ_1, σ_{cert} is computationally indistinguishable from $\sigma_{cert}(i_0)$ or $\sigma_{cert}(i_1)$ (we denote by ϵ_0 the probability that $\mathcal{A}$ is able to distinguish $\sigma_{cert}(i_0)$ from $\sigma_{cert}(i_1)$, ϵ_0 is negligible under the DDH assumption) and r is indistinguishable from r_0 or r_1. Moreover, since the proof is zero-knowledge, the distribution of this challenge is the same as in the full anonymity experiment.
[**Post challenge phase**] $\mathcal{R}$ can answer signing, joining and opening (with a tag different from t^*) queries as previously. If the tag is t^* then she returns $\perp$. Indeed, to produce a valid group signature with the tag $t^* = vk_{ots}$, $\mathcal{A}$ has to forge a new signature σ_{ots} because each modification on the challenge group signature involves a different C^* or a different proof π. Since Σ_{ots} is a strong one-time signature, this event appears with negligible probability.
[**Final output**] Finally, $\mathcal{A}$ outputs a bit b' that $\mathcal{R}$ forwards to her challenger.

$\mathcal{R}$'s behavior is the same as the challenger in the full anonymity experiment, his advantage in breaking the IND-st-wCCA security of Γ is then $(1 - \epsilon_0)\epsilon$.

D.2 Proof of theorem 8

Proof. Let $\mathcal{A}$ be an adversary against the traceability with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ that EUF-CMA breaks the underlying digital signature scheme Σ_{cert}.

[keygen] $\mathcal{R}$ runs the Keygen algorithm as usual, except that she does not run Σ_{cert}.keygen and gets the issuer public key pk_{cert} from her EUF-CMA challenger.
[**opening queries**] $\mathcal{R}$ proceeds as the standard algorithm.
[**joining queries**] When $\mathcal{R}$ receives a joining query, she uses his signing oracle to produce the user's certificate. Then, she proceeds as the standard algorithm to simulate the interaction between the group manager and the new user.
[**Final output**] $\mathcal{A}$ returns an untraceable group signature μ on a message m, which contains a certificate σ_{cert} on sk, the secret key used to sign with Σ_1, such that $sk \neq sk_i$ for all registered user i (else, the group signature is traceable). Using the assumption that (P, V) is a sound proof system, $\mathcal{A}$ has to forge a new certificate, and $\mathcal{R}$ will run the extractor on the NIZK proof π to get sk and returns the pair (sk, σ_{cert}) to her challenger.
$\mathcal{R}$ EUF-CMA breaks the digital signature scheme Σ_{cert} with advantage ϵ.

D.3 Proof of theorem 9

Proof. Let $\mathcal{A}$ be an adversary against the non-frameability with advantage ϵ, we construct $\mathcal{R}$ a reduction using $\mathcal{A}$ that EUF-CMA breaks the underlying digital signature scheme Σ_1.

[keygen] $\mathcal{R}$ runs the Keygen algorithm as usual, except that she computes a simulated common reference string crs. $\mathcal{R}$ chooses $i \in \mathcal{HU}$ and computes $(upk_j, usk_j) \leftarrow \Sigma_0$.keygen for $j \in \mathcal{HU}$.
[**joining queries**] $\mathcal{R}$ has to simulate the user part of the join protocol with the group manager (simulated by $\mathcal{A}$). If $j \neq i$, $\mathcal{R}$ generates a key pair $(pk_j, sk_j) \leftarrow \Sigma_1$.keygen and proceeds as in the standard algorithm. If $j = i$ then $\mathcal{R}$ gets the verification key from her challenger and uses the simulated common reference string to produce the proofs of knowledge.
[**signing queries**] $\mathcal{R}$ proceeds as the standard algorithm if $j \neq i$, else, she uses her signing oracle to produce a signature and uses the simulated common reference string to produce the proofs of knowledge.
[**Final output**] $\mathcal{A}$ returns a group signature μ on a message m and an identity j. If $j \neq i$, then $\mathcal{R}$ aborts the experiment. Else we assume that σ_1 is a valid signature *w.r.t* vk (else, as in the previous scheme, it involves a forgery on Σ_0). We distinguish the two following cases:

1. $\mathcal{O}$Sign oracle was already requested on (i, m), returning μ^*. Then there are two possibilities:
   - vk_{ots} or σ_{cert} are different from those in μ^*, which implies that $\mathcal{A}$ successfully forge a new signature σ_1. Then, $\mathcal{R}$ forwards $(m \| vk_{ots} \| \sigma_{cert}, \sigma_1)$ to her challenger.
   - C or π are different from those in μ^*, then C or π have been re-randomized and $\mathcal{A}$ has to forge a new one-time signature Σ_{ots}. Then $\mathcal{A}$ can be turned in an attacker against the security of Σ_{ots}.
2. There were no oracle request on (i, m), then σ_1 is a forged signature on $m \| vk_{ots} \| \sigma_{cert}$.

Let m be the number of honest users, $\mathcal{R}$ is able to break the EUF-CMA security of Σ_1 with advantage $\frac{1}{m}\epsilon$. $\mathcal{A}$'s advantage ϵ is then negligible.