Group Signature without Random Oracles from Randomizable Signatures

Rémi Clarisse[1,2] and Olivier Sanders[1]

[1] Orange Labs, Applied Crypto Group, Cesson-Sévigné, France
[2] Univ Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France

Abstract. Group signature is a central tool for privacy-preserving protocols, ensuring authentication, anonymity and accountability. It has been massively used in cryptography, either directly or through variants such as direct anonymous attestations. However, it remains a complex tool, especially if one wants to avoid proving security in the random oracle model.
In this work, we propose a new group signature scheme proven secure without random oracles which significantly decreases the complexity in comparison with the state-of-the-art. More specifically, we halve both the size and the computational cost compared to the most efficient alternative in the same model. Moreover, our construction is also competitive against the most efficient ones in the random oracle model.
Our construction is based on a tailored combination of two popular signatures, which avoids the explicit use of encryption schemes or zero-knowledge proofs while signing. It is flexible enough to achieve security in different models and is thus suitable for most contexts.

1 Introduction

Group Signature, introduced by Chaum and van Heyst [17], enables members of a group to sign on behalf of the group. The point is that the signature is anonymous, *i.e.* it cannot be traced back to its issuer, except for a specific entity, the opening authority, which can "open" any valid group signature.

Related Works. Combining seemingly contradictory properties such as authentication and anonymity has proved tricky, the first really practical solution being provided by Ateniese *et al.* [2]. Few years later, Bellare, Micciancio and Warinschi [5] proposed the first security model (BMW model) for static group signature, which was later extended to the case of dynamic group signature by Bellare, Shi and Zhang [6] (BSZ model). Besides providing a way to assess existing schemes, these seminal works have introduced a generic construction that has become the implicit framework for most of the following group signatures.

Informally, a group member of this generic construction receives from a so-called group manager a certificate (a digital signature) τ on his public key pk when he joins the group. To compute a group signature on some message m, he first generates a digital signature σ on m (using the corresponding signing key sk) and then encrypts σ and τ. Finally, he provides a non-interactive zero-knowledge (NIZK) proof that every element is well formed. This three-steps approach is usually known as Sign-Encrypt-Prove (SEP) in the literature.

The strength of the SEP paradigm is that it is based on standard cryptographic primitives for which many instantiations exist. Unfortunately, it leads to quite complex constructions because of the security requirements placed on each building block, but primarily because of the complexity of the resulting NIZK proof. Indeed, the signer

must prove, without revealing σ and τ, that the group signature is a valid encryption of the signature σ that has been generated using keys certified by the group manager.

Such a statement is difficult to prove and this becomes worse if one wants to achieve security without relying on the random oracle model (ROM). Indeed, NIZK proofs are much more complex outside this setting and even by using the Groth-Sahai methodology [24], group signatures still contain dozens of elements (see *e.g.* [22]).

A natural question arising from this observation is whether it is possible to construct more efficient schemes by using a different paradigm. Bichsel *et al.* [7] proposed an interesting answer to this question. They indeed introduced a very efficient alternative, at the cost of a slightly weaker notion of anonymity. This allows them to circumvent the result of Abdalla and Warinschi [1] and thus to avoid encryption. More specifically, their idea was to remove encryption by using re-randomizable [14] certificates τ and by merging σ with the NIZK proofs, leading to a signature of knowledge. The resulting construction is very efficient (see Table 4 at the end of the paper) and can be further improved by instantiating it with the randomizable signature scheme of Pointcheval and Sanders (PS) [28].

Another alternative based on equivalence-class signature [20] has recently been proposed by Derler and Slamanig [18]. It shares commonalities with [7], such as the absence of explicit encryption, but manages to achieve full anonymity at the cost of increased complexity. Unfortunately, both [7] and [18] inherently rely on signature of knowledge and so rather fit the random oracle model.

Very recently, Backes *et al.* [3] proposed a different framework based on a new primitive called signatures with flexible public keys. It yields secure constructions without random oracles with improved efficiency compared to the state-of-the-art in this setting. However, the resulting group signatures are three times larger than the ones in the ROM and require more computations to be generated.

More generally, designers of group signature schemes are confronted with the choice of either proving security without random oracles or favoring efficiency by relying on the random oracle model whose limits are known [16].

Our Contribution. In this work, we propose a new group signature scheme avoiding the ROM that halves the size and the computational complexity compared to the state-of-the-art [3]. More specifically, our group signature only consists of 2304 bits which makes it very competitive, even against constructions in the ROM (see section 6 for more details).

As [3, 7, 18], our construction departs from the SEP framework and heavily relies on the randomizability of its components. However, contrarily to those works that assemble different building blocks (digital signature, NIZK, etc.) and so achieve some level of genericity, we are here interested in optimizing the combination to avoid NIZK proofs in the signature, so as to get the best possible efficiency.

Our work results from the observation that the equivalence-class signature of Fuchsbauer, Hanser and Slamanig (FHS) [20] nicely interacts with the Pointcheval and Sanders (PS) signature scheme [28]. More specifically, assuming very slight modifications of the FHS public key and of the PS signatures, we are able to merge the verification equations of FHS signatures with the one of PS signatures. Such a merge is crucial for our construction: it indeed means that it is no longer necessary to provide a NIZK proof that the signatures are valid and related. Thus, verifying our group signatures is essentially verifying FHS signatures.

Intuitively, we modify the PS signature scheme in such a way that each signature is of the form $(g^r, g^{y \cdot r} X^{r/h_m})$ where g^y is the user's secret key, r is a random scalar, h_m is a public element that depends on the message m to be signed and X is a public element. Leaving out the term in X, one can note that each signature contains a different representative of the same projective equivalence class as (g, g^y) and so it is quite easy, given a FHS signature on this pair, to prove that the PS signature was generated using certified keys.

Our group signature thus only consists of a PS signature and a FHS one which are both re-randomizable, leading to an anonymity proof under the DDH assumption. Moreover, we can prove that a non-registered user cannot generate a valid group signature unless he is able to forge FHS signatures. We only pay the price for our tailored construction in the proof of non-frameability, where we want to prove that no one can issue a forged group signature that can be traced back to an honest user. Indeed, we would like to directly rely on the security of PS signatures but this is impossible due to the modifications we introduced: a PS signature is not enough to answer adversary queries in our security proof. However, we show that we can tweak the original assumption underlying the security of PS signatures to suit our construction and so that we can rely on similar arguments to prove non-frameability.

While being non-generic, our construction remains flexible enough to comply with different group signature models. Interestingly, the different variants we consider achieve the same efficiency with respect to the group signature but mostly differ in the registration procedure. This concretely means that the most suitable setting can be chosen without any impact on the group signature itself. This also allows us in Table 4 to fairly compare our construction with the most relevant ones of the state-of-the-art and so to highlight the benefits of our group signature in all cases.

Organisation. We describe in section 2 the building blocks that we need to construct our group signature. The section 4 recalls the standard security model of group signatures. We describe our construction in section 5 and compare it with the most relevant alternatives of the state-of-the-art in section 6. Due to space limitation, the security proofs are provided in appendix A.

2 Preliminaries

Notations. The identity element of a group $\mathbb{G}$ is denoted $1_{\mathbb{G}}$ and $\mathbb{G}^*$ means $\mathbb{G} \backslash \{1_{\mathbb{G}}\}$. If the group $\mathbb{G}$ is of order p, then we may say interchangeably that $a \in \mathbb{Z}/p\mathbb{Z}$ or that a is a scalar. For a finite set X, the notation $x \xleftarrow{\$} X$ means that x is an element of X uniformly sampled.

2.1 Bilinear Groups

Definition 1. *Bilinear groups are a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of order p along with a map, called pairing, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is*

- *bilinear: for any $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}/p\mathbb{Z}$, $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$;*
- *non-degenerate: for any $g \in \mathbb{G}_1^*$ and $\widetilde{g} \in \mathbb{G}_2^*$, $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$;*
- *efficient: for any $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$, $e(g, \widetilde{g})$ can be efficiently computed.*

We will only consider bilinear groups of prime order with *type*-3 pairings, *i.e.* there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. We stress that this yields the most efficient parameters [25]. To highlight the differences between $\mathbb{G}_1$ and $\mathbb{G}_2$, we will denote elements of the latter with a tilde (*e.g.* $\widetilde{g}$).

3 Digital Signature

A digital signature scheme Σ is defined by four algorithms:

- $\texttt{Setup}(1^\lambda)$: Outputs public parameters pp for security parameter λ.
- $\texttt{Keygen}(pp)$: On input pp, outputs signing and verification keys $(\mathsf{sk}, \mathsf{pk})$.
- $\texttt{Sign}(\mathsf{sk}, m)$: Outputs a signature σ of message m under signing key sk.
- $\texttt{Verify}(\mathsf{pk}, m, \sigma)$: On input verification key pk, message m and its alleged signature σ, outputs 1 if σ is a valid signature on m under pk, and 0 otherwise.

The standard security notion for a signature scheme is *existential unforgeability under chosen message attacks* (EUF-CMA) [21]: it means that it is hard, even given access to a signing oracle, to output a valid pair (m, σ) for a message m never asked to the signing oracle.

PS Signature In [28], Pointcheval and Sanders propose a *randomizable* signature scheme, *i.e.* a scheme enabling to derive re-randomized versions σ' of any valid signature σ. An interesting feature of their signatures is that one cannot link σ and σ' without knowing the corresponding message. They describe several versions of their signature scheme, offering different features. In this work we will use their variant supporting aggregation because it enables to decrease the size of the public key but we will not use this aggregation feature.

- $\texttt{Setup}(1^\lambda)$: Outputs the parameters pp containing the description of type-3 bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ along with a set of generators $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$ and a pair $(X, \widetilde{X}) \leftarrow (g^x, \widetilde{g}^x)$ for some random scalar x.
- $\texttt{Keygen}(pp)$: Generates a random scalar y and sets $(\mathsf{sk}, \mathsf{pk})$ as $(g^y, \widetilde{Y} = \widetilde{g}^y)$.
- $\texttt{Sign}(\mathsf{sk}, m)$: On message m, generates a signature $(\sigma_1, \sigma_2) \leftarrow (g^r, X^r \cdot g^{r \cdot y \cdot m})$ for some random scalar r.
- $\texttt{Verify}(\mathsf{pk}, m, (\sigma_1, \sigma_2))$: Accepts signature (σ_1, σ_2) on m if the following equality holds: $e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^m) = e(\sigma_2, \widetilde{g})$.

One can note that anyone can re-randomize a signature by raising σ_1 and σ_2 to the same power t. The PS signature scheme is proven EUF-CMA-secure under a LRSW assumption customized for type-3 pairing, that we recall in subsection 3.1.

FHS Signature In [20], Fuchsbauer, Hanser and Slamanig introduce a signature on equivalence-class for the following equivalence relation on tuples in $\mathbb{G}_1^n$: $(M_1, \ldots, M_n)$ is in the same equivalence class as $(N_1, \ldots, N_n)$ if there exists a scalar a such that $N_i = M_i^a$ for all $i \in [1, n]$. In this paper, we will only consider the case $n = 2$.

- $\texttt{Setup}(1^\lambda)$: Outputs parameters pp containing the description of type-3 bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with generators $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$.
- $\texttt{Keygen}(pp)$: Generates two random scalars α_1 and α_2 and sets sk as (α_1, α_2) and pk as $(\widetilde{A}_1, \widetilde{A}_2) = (\widetilde{g}^{\alpha_1}, \widetilde{g}^{\alpha_2})$.
- $\texttt{Sign}(\mathsf{sk}, (M_1, M_2))$: Selects a random scalar t and computes the signature $(\tau_1, \tau_2, \widetilde{\tau})$ $\leftarrow ((M_1^{\alpha_1} M_2^{\alpha_2})^t, g^{1/t}, \widetilde{g}^{1/t})$ on the representative $(M_1, M_2) \in \mathbb{G}_1^2$.
- $\texttt{Verify}(\mathsf{pk}, (M_1, M_2), (\tau_1, \tau_2, \widetilde{\tau}))$: Accepts $(\tau_1, \tau_2, \widetilde{\tau}) \in \mathbb{G}_1^2 \times \mathbb{G}_2$, a signature on (M_1, M_2), if $e(\tau_1, \widetilde{\tau}) = e(M_1, \widetilde{A}_1) \cdot e(M_2, \widetilde{A}_2)$ and $e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau})$ hold.

We note that the signature $(\tau_1, \tau_2, \widetilde{\tau})$ is only valid on the representative (M_1, M_2). However, we can easily derive a signature on other representatives (M_1^r, M_2^r) of the same equivalence class, while re-randomizing the signature, by generating a random scalar t' and computing $(\tau_1^{r \cdot t'}, \tau_2^{1/t'}, \widetilde{\tau}^{1/t'})$.

3.1 Computational Assumptions

SXDH assumption. For $i \in \{1,2\}$, the DDH problem is hard in $\mathbb{G}_i$ if, given $(g, g^x, g^y, g^z) \in \mathbb{G}_i^4$, it is hard to distinguish whether $z = x \cdot y$ or z is random. The SXDH assumption holds if DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

PS assumption. Pointcheval and Sanders [28] introduce "Assumption 1", here referred to as PS assumption, to prove the security of their construction.

PS Assumption: Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear group setting of type-3, with g (resp. $\widetilde{g}$) a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$). For $(\widetilde{X} = \widetilde{g}^x, \widetilde{Y} = \widetilde{g}^y)$, where x and y are random scalars, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}/p\mathbb{Z}$ that chooses a random $r \in \mathbb{Z}/p\mathbb{Z}$ and outputs the pair $P = (g^r, g^{r(x+m\cdot y)})$. Given $(g, g^y, \widetilde{g}, \widetilde{X}, \widetilde{Y})$ and unlimited access to this oracle, no adversary can efficiently generate $(m^*, g^r, g^{r(x+m^*\cdot y)})$, with $r \neq 0$, for a new scalar m^*, not asked to $\mathcal{O}$.

MPS assumption. As we explain in the introduction, we would like to directly rely on the PS assumption but this is not possible. In particular, in our group signature construction, the user incorporates parts of the group signature in the message to be signed. This is done by using a suitable map h that must be taken into account by the assumption. We therefore introduce a variant of the PS assumption that we call MPS assumption (M stands for modified).

MPS Assumption: Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ a bilinear group setting of type-3, with g (resp. $\widetilde{g}$) a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$), and $h : \{0,1\}^* \to \mathbb{Z}/p\mathbb{Z}$ a function. For random scalars x, y and z, we define the oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}/p\mathbb{Z}$ that picks random $r, s \in \mathbb{Z}/p\mathbb{Z}$ and outputs the tuple $P = (s, \widetilde{g}^{r\cdot s}, g^r, g^{r\cdot z}, g^{r(x+t\cdot y)})$ with $t = h(\widetilde{g}^{r\cdot s}||g^r||m)$. Given $(g, g^y, g^z, g^{z\cdot x}, \widetilde{g}, \widetilde{X}, \widetilde{Y})$ and unlimited access to this oracle, no adversary can efficiently generate $(t^*, g^r, g^{r(x+t^*\cdot y)})$ with $r \neq 0$ and a value t^* different from those involved in the answers from $\mathcal{O}$.

The validity of the output can easily be checked thanks to the pairing e: $e(g^{r(x+t^*\cdot y)}, \widetilde{g}) \stackrel{?}{=} e(g^r, \widetilde{X} \cdot \widetilde{Y}^{t^*})$. We note that the goal of the adversary is still to output a valid PS signature but it now has access to additional elements that do not seem helpful to create forgeries, as we discuss below. We also give a proof that the MPS assumption holds in the generic group model in Section A.4.

Remark 2.
- Our new oracle still returns a PS signature $(g^r, g^{r(x+ty)})$ but on a scalar $t = h(\widetilde{g}^{r\cdot s}||g^r||m)$ instead of m. However, we define much harder success conditions for the adversary: it can only win if the scalar t used in its forgery is different from the ones used by $\mathcal{O}$ (in particular a forgery on a new message m^* is not valid if it leads to an already used t). Intuitively, this rules out any strategy based on the properties of h (such as collisions). We will therefore assume (and prove in the generic group model) that the MPS assumption holds for *any*[1] function $h : \{0,1\}^* \to \mathbb{Z}/p\mathbb{Z}$. From the security point of view, this therefore does not change anything compared to an assumption where $\mathcal{O}$ would return $(g^r, g^{r(x+my)})$.
- This slight modification induces another one: we now need to provide the pair $(g^z, g^{z\cdot x})$, for some random scalar z, in the assumption. In [28], this pair is exactly

[1] We nevertheless note that the hardness of the corresponding problem depends on the function h. For example, if h is constant then no adversary can succeed as soon as it makes (at least) one query to $\mathcal{O}$.

a signature on 0 and so is directly generated by the reduction in the security proof by running $\mathcal{O}$ on 0. This is no longer possible here, and we then need to explicitly add these elements in the definition of the assumption. In any case, this does not provide more power to the adversary than in the PS assumption.
- The element $g^{r \cdot z}$ is the only one involving the secret z in P. It seems therefore useless to combine it with the other elements of P to derive a new valid tuple.
- The last difference with the PS assumption is the pair $(s, \widetilde{g}^{r \cdot s})$ that must be added to the oracle answers. However, we note that $\widetilde{g}^{r \cdot s}$ is an element of $\mathbb{G}_2$ and so is intuitively useless to forge a PS signature $(g^r, g^{r(x+ty)}) \in \mathbb{G}_1^2$, thanks to the asymmetry of the pairing. The same holds true for s that is not one of the secret values used to compute the PS signature.

4 Group Signature

For completeness, we recall here the security model for dynamic group signature from the BSZ model [6]. We introduce some minor syntactic changes and discuss popular variants of the original security notions introduced by Bellare *et al.* [6]. A reader familiar with group signature can safely jump to Remark 3.

Syntax. A group signature scheme is defined by the following algorithms that involve three types of entities: a group manager, an opening authority and users. Each of the latter is identified by a public index $i \in \mathbb{N}^*$.

- $\mathtt{Setup}(1^\lambda)$: Outputs public parameters pp for security parameter λ.
- $\mathtt{UKeygen}(pp)$: Returns a user's key pair $(\mathsf{sk}, \mathsf{pk})$ on public parameter pp. We assume that pk is public and anyone can get an authentic copy of it.
- $\mathtt{OKeygen}(pp)$: Returns the opening authority's key pair $(\mathsf{osk}, \mathsf{opk})$ under pp.
- $\mathtt{GKeygen}(pp)$: Returns the group manager's key pair $(\mathsf{gsk}, \mathsf{gpk})$ along with a public register $\mathbf{Reg}$, on public parameters pp.
- $\mathtt{Join}$: This is a two-party interactive protocol between the group manager and a user i who wants to join the group. The input of the former is $(\mathsf{gsk}, \mathbf{Reg}, \mathsf{opk}, \mathsf{pk}_i)$ whereas the user takes as input $(\mathsf{gpk}, \mathsf{opk}, \mathsf{sk}_i)$. If the protocol does not fail, then the user gets a group signing key usk_i whereas the group manager updates $\mathbf{Reg}$. Else, both parties return $\perp$.
- $\mathtt{Sign}(\mathsf{usk}_i, m)$: Returns a group signature σ of m under signing key usk_i.
- $\mathtt{Verify}(\mathsf{gpk}, \sigma, m)$: On input the group manager's public key, a group signature σ and a message m, returns a bit $b \in \{0, 1\}$.
- $\mathtt{Open}(\mathsf{osk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m)$: On input the opening authority's secret key, the group manager's public key, the register $\mathbf{Reg}$, a group signature σ and a message m, returns either 0, $\perp$ or an index $i \in \mathbb{N}^*$ along with a proof π.
- $\mathtt{Judge}(\mathsf{gpk}, \mathbf{Reg}, \sigma, m, i, \pi)$: On input the group manager's public key, the register $\mathbf{Reg}$, a group signature σ, a message m, an index $i \in \mathbb{N}^*$ and a proof π, returns a bit $b \in \{0, 1\}$.

Security Model. A group signature should achieve *correctness*, *anonymity*, *traceability* and *non-frameability*. We refer to [6] for a formal definition of correctness, but informally it means that any user who has joined the group should be able to produce valid signatures σ (*i.e.* one for which $\mathtt{Verify}$ outputs 1) on any message m. Moreover, it should be possible to open such signatures, *i.e.* to recover the identity i of the

$\mathtt{Exp}^{an}_{\mathcal{A}}(1^\lambda)$ – Anonymity Security Game

1. $pp \leftarrow \mathtt{Setup}(1^\lambda)$
2. $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathtt{OKeygen}(pp)$
3. $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathtt{GKeygen}(pp)$
4. $b \xleftarrow{\$} \{0,1\}$
5. $b^* \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{J}_U,\mathcal{O}\mathtt{Cor},\mathcal{O}\mathtt{Sign},\mathcal{O}\mathtt{Open},\mathcal{O}\mathtt{Ch}_b}(\mathsf{gsk}, \mathsf{opk})$
6. If $\mathcal{O}\mathtt{Open}$ is queried on the output of $\mathcal{O}\mathtt{Ch}_b$, then return 0
7. Return $(b = b^*)$

$\mathtt{Exp}^{tra}_{\mathcal{A}}(1^\lambda)$ – Traceability Security Game

1. $pp \leftarrow \mathtt{Setup}(1^\lambda)$
2. $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathtt{OKeygen}(pp)$
3. $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathtt{GKeygen}(pp)$
4. $(\sigma, m) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{J}_{GM},\mathcal{O}\mathtt{Cor},\mathcal{O}\mathtt{Sign}}(\mathsf{gpk}, \mathsf{osk})$
5. If $\perp \leftarrow \mathtt{Open}(\mathsf{osk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m)$, then return 1
6. If $(i, \pi) \leftarrow \mathtt{Open}(\mathsf{osk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m)$ and $0 \leftarrow \mathtt{Judge}(\mathsf{gpk}, \mathbf{Reg}, \sigma, m, i, \pi)$, then return 1
7. Return 0

$\mathtt{Exp}^{nf}_{\mathcal{A}}(1^\lambda)$ – Non-Frameability Security Game

1. $pp \leftarrow \mathtt{Setup}(1^\lambda)$
2. $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathtt{OKeygen}(pp)$
3. $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow \mathtt{GKeygen}(pp)$
4. $(\sigma, m, i, \pi) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add},\mathcal{O}\mathtt{J}_U,\mathcal{O}\mathtt{Cor},\mathcal{O}\mathtt{Sign}}(\mathsf{gsk}, \mathsf{osk})$
5. If $\mathcal{O}\mathtt{Sign}$ returned σ, then return 0
6. If i is corrupt, then return 0
7. Return $\mathtt{Judge}(\mathsf{gpk}, \mathbf{Reg}, \sigma, m, i, \pi)$

Fig. 1. Security Games for Group Signature

signer, and to produce publicly verifiable proofs that user i has indeed issued these signatures.

Anonymity requires that group signatures should be anonymous, except for the opening authority. Traceability requires that no one can produced a valid signature that cannot be traced back to some user through the $\mathtt{Open}$ procedure. Finally, non-frameability means that no one can be falsely accused of having produced a signature. The corresponding security games, outlined in Figure 1, make use of the following oracles:

- $\mathcal{O}\mathtt{Add}(i)$ is an oracle that can be used to add a new user i. It then runs $\mathtt{UKeygen}(pp)$ to get $(\mathsf{sk}_i, \mathsf{pk}_i)$ and returns pk_i. If i has already been used in a previous query, then it returns $\perp$.
- $\mathcal{O}\mathtt{J}_U(i)$ is an oracle that plays the user's side of the $\mathtt{Join}$ protocol. It can be used by an adversary $\mathcal{A}$ playing the role of a corrupt group manager. It returns $\perp$ if i has already joined the group or if user i does not exist.
- $\mathcal{O}\mathtt{Cor}(i)$ is an oracle that returns all the secret keys of the user i. The user i is then said to be *corrupt*. Any non-corrupt user is considered *honest*.
- $\mathcal{O}\mathtt{J}_{GM}()$ is the counterpart of the $\mathcal{O}\mathtt{J}_U$ oracle that can be used by a corrupt user to join the group.
- $\mathcal{O}\mathtt{Sign}(i, m)$ is an oracle that returns $\mathtt{Sign}(\mathsf{usk}_i, m)$, provided that i is an honest user that has already joined the group.
- $\mathcal{O}\mathtt{Open}(\sigma, m)$ is an oracle that returns $\mathtt{Open}(\mathsf{osk}, \mathsf{gpk}, \mathbf{Reg}, \sigma, m)$.
- $\mathcal{O}\mathtt{Ch}_b(i_0, i_1, m)$ is an oracle that takes as inputs the index of two honest users and returns $\mathtt{Sign}(\mathsf{usk}_{i_b}, m)$.

Let $\mathcal{A}$ be a probabilistic polynomial adversary. A group signature scheme is

- anonymous if $\mathtt{Adv}^{an}(\mathcal{A}) = |\Pr[\mathtt{Exp}^{an}_{\mathcal{A}}(1^\lambda) = 1] - 1/2|$ is negligible for any $\mathcal{A}$;
- traceable if $\mathtt{Adv}^{tra}(\mathcal{A}) = \Pr[\mathtt{Exp}^{tra}_{\mathcal{A}}(1^\lambda) = 1]$ is negligible for any $\mathcal{A}$;
- non-frameable if $\mathtt{Adv}^{nf}(\mathcal{A}) = \Pr[\mathtt{Exp}^{nf}_{\mathcal{A}}(1^\lambda) = 1]$ is negligible for any $\mathcal{A}$.

The security model introduced by Bellare, Shi and Zhang [6] places no restriction on the $\mathcal{O}\mathtt{Cor}$ queries in the anonymity experiment. This means that the adversary is allowed to corrupt the "challenge" users (*i.e.* those that are involved in $\mathcal{O}\mathtt{Ch}$

queries). This corresponds to the strongest notion of anonymity, sometimes called *full anonymity* or *CCA-2 anonymity* (see *e.g.* [18]), where anonymity holds even if the users' secret keys are leaked.

Remark 3. The BSZ model [6] defines strong security properties that are sufficient in most contexts. However, it may be possible in some situations to relax some of them, usually leading to more efficient constructions. This is particularly true for the anonymity property for which popular variants exist, such as *CPA anonymity* [8, 13] or *selfless anonymity* [7,9,28]. The former removes the oracle $\mathcal{O}\mathtt{Open}$ in the anonymity game but the users remain anonymous even if their secret keys leak. Contrarily, selfless anonymity allows $\mathcal{O}\mathtt{Open}$ queries but users are no longer anonymous when their secret keys leak. These two notions are incomparable and so fit different contexts. The construction we describe in the next section achieves both of them. Interestingly, it also achieves full anonymity in the model introduced by Bellare, Micciancio and Warinschi [5] (BMW model), where the group manager is also the opening authority.

5 Our Construction

5.1 Intuition

A group signature usually contains two kinds of digital signatures that we will denote by σ and τ. The first one is issued on the message to be signed by the user using his own key pair $(\mathsf{usk}, \mathsf{upk})$. Intuitively, the unforgeability of the digital signature ensures that no adversary is able to produce a forged group signature which can be traced back to upk (non-frameability). The second one is issued by the group manager on usk (or upk) to differentiate key pairs of group members from those of unregistered users. Here, unforgeability ensures that only users that have joined the group can issue group signature, which is necessary to achieve traceability.

If non-frameability and traceability were the only two conditions expected from a group signature, then the latter would simply be $(\tau, \sigma, \mathsf{upk}, m)$. However, this cannot work when anonymity is also required so the standard practice has been to encrypt/commit at least τ and upk and then provide zero-knowledge proofs that these elements are well-formed.

The work of Bichsel *et al* [7] has shown that we can do better when τ is randomizable. Indeed, in such a case there is no need to encrypt τ, the latter can simply be re-randomized and sent unencrypted, leading to significant gains in efficiency. Their group signature can only achieve a weaker selfless anonymity notion in the ROM, but it seems a reasonable price to pay in view of the benefits.

Despite its novelty, [7] still shares commonalities with the standard framework of the BSZ model [6]. There is indeed still a modular composition of two signatures τ and σ with a proof of knowledge. The latter two can be merged (leading to a signature of knowledge) using the Fiat-Shamir heuristic [19] in the ROM, but the spirit remains the same. Modular systems are interesting since they can leverage any advance in the construction of their building blocks. For example, the scheme of [7] can straightforwardly be improved by using PS signatures [28] to instantiate τ, instead of Camenisch-Lysyanskaya signatures [14] in the original construction. Unfortunately, the complexity of a modular construction is the sum of all its parts, so a natural question is whether it is possible to improve efficiency by optimizing the combination of the different building blocks for some specific instantiations.

In this section we construct the most efficient group signature without random oracles by noticing that FHS equivalence-class signatures [20] nicely interact with PS signatures [28]. Indeed let us recall the latter, and more specifically its variant designed to support aggregation. A (non-aggregated) signature on a message m in this case is given by $(\sigma_1, \sigma_2) = (g^r, X^r(g^{y \cdot m})^r)$ where r is some random scalar, $X = g^x$ is a public element and y is the signer's secret key. One can note that we can alternatively define σ_2 as $\sigma_2^{1/m} = X^{r/m}(g^y)^r$: any adversary able to forge such a signature can trivially be converted into an adversary against the original PS-signature scheme.

Therefore, any signature issued by a user will be of the form $(\sigma_1, \sigma_2) = (g^r, X^{r/m}(g^y)^r)$. If we applied the standard methodology here, we would provide a signature τ on y (or g^y) and then prove in a zero-knowledge way that τ is valid on the key that has been used to generate (σ_1, σ_2). However, we can do better if we directly use the FHS-signature scheme [20].

Indeed, for all r, if we discard the term $X^{r/m}$ in σ_2, it only remains $(g^r, g^{y \cdot r})$ which are different representatives of the same equivalent class. Thus, if we provide a FHS signature on $(g^r, g^{r \cdot y})$ one can *directly* check that (σ_1, σ_2) was generated using a certified key, without any proof of knowledge. Anonymity of the resulting construction simply follows from the ability to re-randomize FHS signature while changing the representative of the class.

It then only remains to explain how to remove $X^{r/m}$. Recall that a FHS signature on $(g^r, g^{r \cdot y})$ is a tuple $(\tau_1, \tau_2, \widetilde{\tau})$ such that

$$e(\tau_1, \widetilde{\tau}) = e(g^r, \widetilde{A}_1) \cdot e(g^{r \cdot y}, \widetilde{A}_2) \qquad \text{and} \qquad e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau}),$$

where $(\widetilde{A}_1, \widetilde{A}_2) = (\widetilde{g}^{\alpha_1}, \widetilde{g}^{\alpha_2})$ is the public key. Assume that we add $\widetilde{B} = \widetilde{X}^{\alpha_2}$ to this public key ($\widetilde{X} = \widetilde{g}^x$ is a part of the public key of the PS-signature scheme). Then, $e(\sigma_1, \widetilde{A}_1) \cdot e(\sigma_2, \widetilde{A}_2) \cdot e(\sigma_1, \widetilde{B}^{-1/m}) = e(g^r, \widetilde{A}_1) \cdot e(g^{y \cdot r}, \widetilde{A}_2) = e(\tau_1, \widetilde{\tau})$, and the second equation remains unchanged. This means that we can check the validity of both FHS and PS signatures at essentially the cost of verifying a FHS signature. Moreover, the fact that we merge the verification of these signatures makes zero-knowledge proofs unnecessary. Concretely, this means that our group signature only consists of $(\sigma_1, \sigma_2, \tau_1, \tau_2, \widetilde{\tau})$, *i.e.* four elements of $\mathbb{G}_1$ and one element of $\mathbb{G}_2$, and can be verified with merely two pairing equations.

Interestingly, the fact that we avoid the classical signature of knowledge of y allows to achieve both CPA anonymity and selfless anonymity. Indeed, schemes based on randomizable signatures (see *e.g.* [14, 28]) are usually proven anonymous under the DDH assumption in $\mathbb{G}_1$. Therefore, to enable opening, they usually force the users to provide some "trapdoor" $\widetilde{g}^y \in \mathbb{G}_2$ that allows the opening authority to break DDH on their specific signatures. When y is part of the user's signing key usk (which is necessary for a signature of knowledge of y), leakage of the latter means that the adversary can recover y and thus $\widetilde{g}^y$. Anonymity can then no longer hold in this case leading to the selfless anonymity notion.

In our case, we note that $g^y \in \mathbb{G}_1$ is enough to issue group signatures, meaning that users can discard y after generating their keys. In case usk leaks, the adversary now recovers g^y, which is useless to break DDH. We can thus retain some level of anonymity (at least CPA anonymity) in this case.

5.2 The Protocol

We now formalize the previous intuition by describing all the algorithms that constitute our scheme. As we explain above, we manage to avoid NIZK proofs and explicit

encryption in our signature. However, we still need such primitives for some algorithms such as `Join` and `Open`. Fortunately, the latter are in practice subject to less constraints that `Sign` as they have less impact on the user's experience (in particular because they are run far less often than `Sign`).

Our construction therefore also makes use of a public key encryption scheme Γ and of a NIZK proof system. The latter will concretely be the Groth-Sahai proof system [24] that allows to prove most common relations in bilinear groups by using a common reference string *crs*. Additional details on these two primitives are provided in Appendix B.

- $\texttt{Setup}(1^\lambda)$: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be the description of type-3 bilinear groups of prime order p, this algorithm first selects $g \xleftarrow{\$} \mathbb{G}_1^*$ and $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2^*$, and then computes $(X, \widetilde{X}) \leftarrow (g^x, \widetilde{g}^x)$ for some random scalar x. It also generates the public parameters pp_Σ for a digital signature scheme[2] Σ and selects a hash function $h : \{0,1\}^* \to \mathbb{Z}/p\mathbb{Z}$. Finally, it generates a common reference string *crs* for the Groth-Sahai proof system [24] in the SXDH setting and then sets the public parameters as $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g}, X, \widetilde{X}, crs, pp_\Sigma, h)$.
- $\texttt{UKeygen}(pp)$: The user defines his own key pair as $(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\texttt{Keygen}(pp_\Sigma)$.
- $\texttt{OKeygen}(pp)$: The opening authority generates a key pair $(\mathsf{osk}, \mathsf{opk})$ for a public key encryption scheme Γ.
- $\texttt{GKeygen}(pp)$: The group manager selects two random scalars α_1 and α_2 and then computes $(\widetilde{A}_1, \widetilde{A}_2, \widetilde{B}) \leftarrow (\widetilde{g}^{\alpha_1}, \widetilde{g}^{\alpha_2}, \widetilde{X}^{\alpha_2})$. He then initializes a public register $\mathbf{Reg}$ and returns $(\mathsf{gsk}, \mathsf{gpk}) \leftarrow ((\alpha_1, \alpha_2), (\widetilde{A}_1, \widetilde{A}_2, \widetilde{B}))$.
- `Join`: To join the group, a user i first selects two random scalars, u and y, and computes $(g^u, g^{u \cdot y})$ along with $C \leftarrow \Gamma.\texttt{Encrypt}(\mathsf{opk}, \widetilde{g}^y)$. He then generates a NIZK proof π that C encrypts an element $\widetilde{g}^y \in \mathbb{G}_2$ such that $e(g^u, \widetilde{g}^y) = e(g^{u \cdot y}, \widetilde{g})$. Finally, he generates $\mu \leftarrow \Sigma.\texttt{Sign}(\mathsf{sk}_i, (g^u || g^{u \cdot y} || C || \pi))$ and sends it, along with $(g^u, g^{u \cdot y}, C, \pi)$, to the group manager.
  Upon receiving these elements, the group manager checks the validity of the proof π and that $\Sigma.\texttt{Verify}(\mathsf{pk}_i, \mu, (g^u || g^{u \cdot y} || C || \pi)) = 1$. If π and μ are both valid, then he stores $(g^u, g^{u \cdot y}, C, \pi, \mathsf{pk}_i, \mu)$ in $\mathbf{Reg}[i]$, generates a $t \xleftarrow{\$} \mathbb{Z}/p\mathbb{Z}$ and returns $\tau_1' \leftarrow ((g^u)^{\alpha_1} (g^{u \cdot y})^{\alpha_2})^t$, $\tau_2 \leftarrow g^{1/t}$ and $\widetilde{\tau} \leftarrow \widetilde{g}^{1/t}$.
  Finally, the user computes $\tau_1 \leftarrow (\tau_1')^{1/u}$ and sets $\mathsf{usk}_i = (\tau_1, \tau_2, \widetilde{\tau}, g^y)$.
- $\texttt{Sign}(\mathsf{usk}_i, m)$: To sign a message m, the user first selects two random scalars r and s, and generates the following elements:

$$\tau_1' \leftarrow \tau_1^{r \cdot s}, \quad (\tau_2', \widetilde{\tau}') \leftarrow (\tau_2^{1/s}, \widetilde{\tau}^{1/s}), \quad (\sigma_1, \sigma_2) \leftarrow (g^r, X^{r/h(\widetilde{\tau}' || \sigma_1 || m)} \cdot (g^y)^r).$$

  The group signature σ on m is then defined as $\sigma = (\tau_1', \tau_2', \widetilde{\tau}', \sigma_1, \sigma_2)$.
- $\texttt{Verify}(\mathsf{gpk}, \sigma, m)$: To verify a group signature σ on m, one checks that none of its elements is $1_{\mathbb{G}_1}$ or $1_{\mathbb{G}_2}$ and that the following equalities hold:

$$e(\sigma_1, \widetilde{A}_1 \widetilde{B}^{-1/h(\widetilde{\tau} || \sigma_1 || m)}) \cdot e(\sigma_2, \widetilde{A}_2) = e(\tau_1, \widetilde{\tau}) \quad \text{and} \quad e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau}),$$

  in which case one outputs 1. Otherwise, one returns 0.
- $\texttt{Open}(\mathsf{osk}, \mathsf{gpk}, \sigma, m)$: Before opening a signature, the opening authority first checks that it is valid. Otherwise, he returns 0. By using its secret key osk, the opening authority has the ability to decrypt any ciphertext C_i stored in $\mathbf{Reg}[i]$ and thus

[2] Any EUF-CMA signature scheme can be selected here, without any impact on the complexity of the group signatures.

recover the elements $\widetilde{g}^{y_i} \in \mathbb{G}_2$ for all registered users. He can then check, for each of them, whether the following equality holds:

$$e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1, \widetilde{X}^{-1/h(\widetilde{\tau}||\sigma_1||m)}) = e(\sigma_1, \widetilde{g}^{y_i}).$$

If there is no match, then the opening authority returns $\perp$. Otherwise, let j be the corresponding user. The opening authority recovers the data $(g^{u_j}, g^{u_j \cdot y_j}, C_j, \pi_j, \mathsf{pk}_j, \mu_j)$ stored in $\mathbf{Reg}[j]$, commits to $\widetilde{g}^{y_j}$ and then outputs j along with a Groth-Sahai proof π that:

$$e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1, \widetilde{X}^{-1/h(\widetilde{\tau}||\sigma_1||m)}) = e(\sigma_1, \widetilde{g}^{y_j}) \quad \text{and} \quad e(g^{u_j \cdot y_j}, \widetilde{g}) = e(g^{u_j}, \widetilde{g}^{y_j}).$$

– $\mathtt{Judge}(\mathsf{gpk}, \sigma, m, i, \pi)$: To verify an opening, one checks that π_i is valid, $\mathtt{Verify}(\mathsf{gpk}, \sigma, m) = 1$ and $\Sigma.\mathtt{Verify}(\mathsf{pk}_i, \mu_i, (g^{u_i}||g^{u_i \cdot y_i}||C_i||\pi_i)) = 1$. If all conditions are satisfied, then one returns 1. Otherwise, one returns 0.

Correctness. First note that at the end of the `Join` protocol, the user gets a FHS equivalence-class signature [20] on the representative (g, g^y). Indeed, $(\tau_1, \tau_2, \widetilde{\tau}) = ((g^{\alpha_1} g^{y \cdot \alpha_2})^t, g^{1/t}, \widetilde{g}^{1/t})$. To issue a group signature on m, the user first re-randomizes $(\tau_1, \tau_2, \widetilde{\tau})$ using s while updating the representative to $(g^r, g^{r \cdot y})$. The resulting tuple $(\tau_1', \tau_2', \widetilde{\tau}')$ is $((g^{r \cdot \alpha_1} g^{r \cdot y \cdot \alpha_2})^{t \cdot s}, g^{1/(t \cdot s)}, \widetilde{g}^{1/(t \cdot s)})$ and is still a FHS signature on the same equivalent class. He then generates a pair (σ_1, σ_2) where $(\sigma_1, \sigma_2^{m'})$ is a PS signature [28] on $m' = h(\widetilde{\tau}'||\sigma_1||m)$ using the same randomness r. Therefore, such a group signature satisfies:

$$\begin{aligned} e\left(\sigma_1, \widetilde{A}_1 \widetilde{B}^{-1/m'}\right) \cdot e(\sigma_2, \widetilde{A}_2) &= e\left(g^r, \widetilde{g}^{\alpha_1 - \frac{x \cdot \alpha_2}{m'}}\right) \cdot e\left(g^{r(\frac{x}{m'} + y)}, \widetilde{g}^{\alpha_2}\right), \\ &= e(g, \widetilde{g})^{r(\alpha_1 + y \cdot \alpha_2)} = e(\tau_1', \widetilde{\tau}'), \end{aligned}$$

and $e(\tau_2', \widetilde{g}) = e(g^{1/(t \cdot s)}, \widetilde{g}) = e(g, \widetilde{g}^{1/(t \cdot s)}) = e(g, \widetilde{\tau}')$.

Remark 4. Group signatures following the classical Sign-Encrypt-Prove framework usually provide an efficient opening procedure. Indeed, the opening authority knows the corresponding decryption key and so can decrypt the ciphertext included in the group signature and then identify the signer. Unfortunately, there is no equivalent for constructions without encryption and in particular there is no longer a "master" key that the opening authority can use to break anonymity.

Constructions based on randomizable signatures [7, 18] circumvent this issue by forcing each user to provide to the opening authority a way to open their signatures. Concretely, during the `Join` protocol, each user must transmit some elements depending on their secret keys to this authority. Unfortunately this requirement does not fit the BSZ model [6] where `Join` is a two-party protocol between the user and the group manager. There are then two ways to solve this problem. Either we add the opening authority as an acting party in `Join` or we require that the user sends these elements to the group manager. The first solution is conceptually the simplest but modifies the original BSZ model. The second one does not but requires additional primitives to ensure security. Indeed, the user cannot transmit such elements in clear (otherwise the group manager could break anonymity) so he must send them encrypted and prove (in a zero-knowledge way) that the resulting ciphertext is well-formed.

In this paper we choose to describe the most complex (second) solution since one can easily derive from it a group signature scheme complying with the first option. We

will then need an IND-CCA2 secure public key encryption scheme that is compatible with NIZK proofs. In practice, one can choose for instance [27] that nicely interacts with Groth-Sahai proofs [24]. We note that efficiency is not really a concern here since this step of the `Join` protocol has no impact on the group signatures themselves.

Remark 5. We note that the security model of Bellare *et al* [6] already assumes a trusted `Setup` phase, so our construction perfectly fits this model on this point. However, this does not explain how to generate the public parameters in real-world conditions. In practice, it would be natural that the opening authority generates them. Regarding security, it would only be problematic for non-frameability if corruption of this entity occurred *before* `Setup`, but this is excluded by the model of [6]. We can also mitigate the risks by relying on a cooperative generation of the parameters, as in [15].

5.3 Security Results.

Theorem 6. *Our group signature is:*

- *traceable under the EUF-CMA security of the FHS signature scheme;*
- *non-frameable under the* MPS *assumption, the collision-resistance of the function h and the EUF-CMA security of Σ;*
- *CPA anonymous under the* SXDH *assumption and the IND-CCA2 of Γ;*
- *selfless anonymous if it is non-frameable, if Γ is IND-CCA2 secure and if the* SXDH *assumption holds;*
- *fully anonymous, with merged opening authority and group manager, if it is traceable and if the* SXDH *assumption holds.*

Theorem 6, proved in appendix A, shows that our scheme retains some security properties (namely CPA security) even when users' secret keys are leaked, contrarily to the ones of [7,28]. The fact that selfless anonymity depends on the non-frameability may seem surprising but this is due to the special opening process that the reduction $\mathcal{R}$ uses in our security proof. Informally, $\mathcal{R}$ is able to open all signatures but the ones generated by the "challenge" user. To circumvent this problem $\mathcal{R}$ stores all the signatures it has produced on behalf of this user so that it will be able to recognize them if they are later submitted to the $\mathcal{O}$`Open` oracle. However, this works as long as the adversary is unable to forge signatures for this user, hence the non-frameability requirement.

The last statement of the theorem shows that we can achieve the strongest notion of anonymity if we additionally assume that the opening authority is also the group manager, as in the model of Bellare, Micciancio and Warinschi [5].

6 Efficiency comparison

We compare the signing algorithm of our scheme with the ones of other constructions of the state-of-the-art. All of them are proven under interactive assumptions (or directly in the generic group model) so we do not take this point into account in our comparison.

In Table 4, we enumerate the number of expensive operations, *i.e.* exponentiations in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ (denoted by e_1, e_2 and e_T respectively). Regarding signing cost, our scheme is the most efficient one whenever computing $3\mathsf{e}_1 + 1\mathsf{e}_2$ is cheaper than computing $1\mathsf{e}_T$.

To compare these operations, we choose a common metric. We aim at the 128-bit security. Following the incentive of Barbulescu and Duquesne [4], we select Barreto-Lynn-Scott curves with $k = 12$ that now seem more appropriate than Barreto-Naehrig ones, considering the recent attacks on pairings [26]. Moreover, they are getting involved in more implementations, *e.g.* in Zexe [12], a ledger-based system, and in ZCash [11] for zk-SNARKs.

Like in [4], we select a prime $q \equiv 3 \pmod 4$ and construct the tower of fields:

$$\mathbb{F}_{q^2} = \frac{\mathbb{F}_q[U]}{(U^2+1)}, \qquad \mathbb{F}_{q^6} = \frac{\mathbb{F}_{q^2}[V]}{(V^3-U-1)} \qquad \text{and} \qquad \mathbb{F}_{q^{12}} = \frac{\mathbb{F}_{q^6}[W]}{(W^2-V)}.$$

This choice yields the costs in Table 1, where $\mathbf{m}$ is the cost of one multiplication in $\mathbb{F}_q$ (we make the rough assumption that squaring is the same cost as multiplying in $\mathbb{F}_q$). The last line of Table 1 represents costs in the so-called cyclotomic subgroup $\mathbb{G}_{\Phi_{12}(q)} \subset \mathbb{F}_{q^{12}}$ of order $\Phi_{12}(q)$ (where $\Phi_{12}(q)$ is the 12^{th} cyclotomic polynomial evaluated at q, see [4]). This is of interest to us since $\mathbb{G}_T \subset \mathbb{G}_{\Phi_{12}(q)}$ and squaring in $\mathbb{G}_{\Phi_{12}(q)}$ are twice faster.

Field	Mult. (M)	Squaring (S)
$\mathbb{F}_q$	$\mathbf{m}$	$\mathbf{m}$
$\mathbb{F}_{q^2}$	$3\mathbf{m}$	$2\mathbf{m}$
$\mathbb{F}_{q^{12}}$	$54\mathbf{m}$	$36\mathbf{m}$
$\mathbb{G}_{\Phi_{12}(q)}$	$54\mathbf{m}$	$18\mathbf{m}$

Table 1. Costs of arithmetic operations in the tower extension as in [4]

For simplicity, we take our BLS12 curve in the short Weierstrass model, that is $y^2 = x^3 + b$ with $b \in \mathbb{F}_q$, and use the Jacobian coordinate system: representing (x, y) as (X, Y, Z) and satisfying the equations $x = X/Z^2$ and $y = Y/Z^3$. This is the most efficient for pairings, without changing models (see [10]): it takes 11 field multiplications and 5 field squarings to add two distinct points and 2 field multiplications and 5 field squarings to double a point. After converting squarings to multiplications (see Table 1), we end up with Table 2. Note that a point in $\mathbb{G}_2$ is on the degree-6 twist curve, *i.e.* over $\mathbb{F}_{q^2}$.

Group	Addition $11M + 5S$	Doubling $2M + 5S$
$\mathbb{G}_1$	$16\mathbf{m}$	$7\mathbf{m}$
$\mathbb{G}_2$	$43\mathbf{m}$	$16\mathbf{m}$

Table 2. Costs of arithmetic operations in the pairing groups $\mathbb{G}_1/\mathbb{F}_q$ and $\mathbb{G}_2/\mathbb{F}_{q^2}$, when modeling the curve with a short Weierstrass equation and using Jacobian coordinates

Now, to compare exponentiation, let n be a positive integer. Think of n as one of the random scalars in our `Sign` procedure. A square-and-multiply algorithm will, on average, "square" $\log_2 n$ times and "multiply" $(\log_2 n)/2$ times. It means, for instance, that an exponentiation by n in $\mathbb{G}_1$ costs $15(\log_2 n)$ field multiplications. In the following, we bound n by p (the order of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$) and so get Table 3, where $\mathbf{k} = \log_2 p$.

Group	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$
Cost	$15\mathbf{k} \cdot \mathbf{m}$	$38\mathbf{k} \cdot \mathbf{m}$	$45\mathbf{k} \cdot \mathbf{m}$

Table 3. Normalized cost of one group exponentiation

Using BLS12 curves leads to a 256-bit representation of scalars, at least 384-bit for the elements of $\mathbb{G}_1$ and 768-bit for those of $\mathbb{G}_2$. We summarize our comparison

in Table 4, where the BMW model is the one of [5] whereas the BSZ model is the one of [6] that we recall in section 4. The last line of the table corresponds to our construction where the opening authority and the group manager are merged, which impacts security but not efficiency.

Scheme	Size in bit	Cost in grp. exp.	Cost with BLS12	ROM?	GS model	Anonymity
BCNSW [7]	1664	3 e_1 + 1 e_T	90$\mathbf{k} \cdot \mathbf{m}$	yes	BMW	selfless
PS [28]	1280	2 e_1 + 1 e_T	75$\mathbf{k} \cdot \mathbf{m}$	yes	BMW	selfless
DS [18]	2816	5 e_1 + 1 e_2	113$\mathbf{k} \cdot \mathbf{m}$	yes	BSZ	CPA
DS* [18]	4608	5 e_1 + 6 e_2	303$\mathbf{k} \cdot \mathbf{m}$	yes	BSZ	full
BHKS [3]	4992	9 e_1 + 2 e_2	211$\mathbf{k} \cdot \mathbf{m}$	no	BMW	full
Ours	2304	5 e_1 + 1 e_2	113$\mathbf{k} \cdot \mathbf{m}$	no	BSZ	CPA & selfless
Ours*	2304	5 e_1 + 1 e_2	113$\mathbf{k} \cdot \mathbf{m}$	no	BMW	full

Table 4. Efficiency and security comparisons using BLS12 curves ($\mathbf{m}$ represents the cost of one multiplication in the base field of the curve and $\mathbf{k} = \lceil \log_2 p \rceil$)

If we focus on constructions without random oracles, our group signature outperforms the recent construction of [3]: it halves both the signature size and the signature cost. We also note that it is competitive against the most efficient construction [28] in the random oracle model (ROM). Indeed, while the signature size remains larger (double the size), the computational cost is quite similar and, more importantly, our signer no longer needs to perform operation in $\mathbb{G}_T$ and so, does not need to implement the arithmetic in $\mathbb{F}_{q^{12}}$, which is noticeable.

We would like to add that this comparison was made targeting the 128-bit security level. At higher security levels, BLS12 curves might not be relevant anymore. For instance, at 256 bits of security, the authors from [10] choose a BLS24 curve and different curve models for $\mathbb{G}_1$ and $\mathbb{G}_2$, thus satisfying the condition $3\mathsf{e}_1 + 1\mathsf{e}_2 < 1\mathsf{e}_T$. In that case, our group signature scheme is computationally the most efficient, even compared to the best alternative in the ROM [28].

Conclusion

In this paper, we have introduced the most efficient group signature scheme proved secure without random oracles. Our construction is based on a tailored combination of the PS signature scheme and the FHS equivalence-class signature scheme, leading to a group signature consisting only of four elements in $\mathbb{G}_1$ and one in $\mathbb{G}_2$. Its security mostly relies on the one of these signature schemes which have been widely used in cryptographic protocols, although we need to adapt the proof of PS signature to fit our construction.

Our scheme halves both the size and the computational cost compared to the most efficient alternative in the same model. It also significantly closes the gap with constructions in the ROM, showing that we can avoid this model without dramatically increasing complexity.

References

1. Michel Abdalla and Bogdan Warinschi. On the minimal assumptions of group signature schemes. *ICICS 04*.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. *CRYPTO 2000*.

3. Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. *ASIACRYPT 2018, Part II*.
4. Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *J. Cryptology*, 2019.
5. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. *EUROCRYPT 2003*.
6. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. *CT-RSA 2005*.
7. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. *SCN 10*.
8. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. *CRYPTO 2004*.
9. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. *ACM CCS 2004*.
10. Joppe W. Bos, Craig Costello, and Michael Naehrig. Exponentiating in pairing groups. *SAC 2013*.
11. Sean Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. https://electriccoin.co/blog/new-snark-curve/, 2017.
12. Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. Zexe: Enabling decentralized private computation. *IACR Cryptology ePrint Archive*, 2018.
13. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. *EUROCRYPT 2006*.
14. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. *CRYPTO 2004*.
15. Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible E-cash made practical. *PKC 2015*.
16. Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. *TCC 2004*.
17. David Chaum and Eugène van Heyst. Group signatures. *EUROCRYPT'91*.
18. David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. *ASIACCS 18*.
19. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. *CRYPTO'86*.
20. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 2019.
21. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 1988.
22. Jens Groth. Fully anonymous group signatures without random oracles. *ASIACRYPT 2007*.
23. Jens Groth. On the size of pairing-based non-interactive arguments. *EUROCRYPT 2016, Part II*.
24. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. *EUROCRYPT 2008*.
25. Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. *ACNS 13*
26. Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. *CRYPTO 2016, Part I*.
27. Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. Non-malleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. *EUROCRYPT 2014*.
28. David Pointcheval and Olivier Sanders. Short randomizable signatures. *CT-RSA 2016*.

A Security Proofs

A.1 Proof of Anonymity

Our proofs of CPA anonymity and selfless anonymity are very similar and only differ by one game. We will then consider an adversary against "anonymity" without specifying which property we consider except in Game 5 where the distinction is necessary. We discuss the case of full anonymity in Remark 7.

Let $\mathcal{A}$ be an adversary against the anonymity of our construction succeeding with probability ε. We define a sequence of games to show that this advantage is negligible. For each Game i we define $\mathtt{Adv}_i = |\Pr(S_i) - 1/2|$, where S_i is the event "$\mathcal{A}$ succeeds in Game i". We additionally define $\mathtt{Adv}_{\mathsf{SXDH}}$ as the advantage against the SXDH problem.

Game 1. Our first game is exactly the one of anonymity of Figure 1 where the reduction $\mathcal{R}$ generates normally all the secret values and so is able to answer any oracle query. By definition, we have $\mathtt{Adv}_1 = \varepsilon$.

Game 2. In our second game, $\mathcal{R}$ selects a random index $i^* \in [1, q_A]$, where q_A is a bound on the number of $\mathcal{O}\mathtt{Add}$ queries. $\mathcal{R}$ proceeds as usual but aborts if $\mathcal{A}$ queries (i_0, i_1, m) to the $\mathcal{O}\mathtt{Ch}$ oracle with $i_b \neq i^*$. The advantage of $\mathcal{A}$ in this new game is then at least $\frac{\varepsilon}{q_A}$.

Game 3. In the third game, $\mathcal{R}$ generates a simulated common reference string crs and simulates all the zero-knowledge proofs. Any change in the behaviour of $\mathcal{A}$ can then be used against the zero-knowledge property of these proofs, which rely on SXDH in our setting. Therefore, $\mathtt{Adv}_3 \geq \mathtt{Adv}_2 - \mathtt{Adv}_{\mathsf{SXDH}}$.

Game 4. In the fourth game, $\mathcal{R}$ sets opk as the public key of a IND-CCA 2 experiment. It then uses the decryption oracle to decrypt the ciphertext C_i stored in $\mathbf{Reg}[i]$ for all users i and so can answer any query as usual. However, upon receiving the $\mathcal{O}\mathtt{J}_U$ query on i^* (this query necessarily occurs because of Game 2), it proceeds normally except that it generates C as an encryption of a random element of $\mathbb{G}_2$ and simulates the proof. A change in the behaviour of $\mathcal{A}$ would imply an attack against the IND-CCA2 security of Γ, so we get $\mathtt{Adv}_4 \geq \mathtt{Adv}_3 - \mathtt{Adv}_{\mathsf{IND-CCA2}}$.

Game 5. In the fifth game, $\mathcal{R}$ stores every signature it generates on behalf of i^* in some register **Sig**. Upon receiving a $\mathtt{Open}$ query for some pair (σ, m), it first checks whether $\sigma \in \mathbf{Sig}$ in which case it returns i^* along with a simulated proof. Otherwise, it returns $\mathtt{Open}(\sigma, m)$.

We note that Game 5 is the same as Game 4 when we consider CPA anonymity since there is no $\mathcal{O}\mathtt{Open}$ query in this case. For selfless anonymity, a difference only occurs when the adversary manages to submit a forged signature that can be traced back to i^*. However, such an adversary can straightforwardly be converted into an adversary against non-frameability. We then have $\mathtt{Adv}_5 \geq \mathtt{Adv}_4 - \mathtt{Adv}^{nf}$.

Game 6. In the sixth game, $\mathcal{R}$ proceeds as in the previous game except that it answers to the $\mathcal{O}\mathtt{Ch}$ query by returning a signature generated using a random key. The advantage of $\mathcal{A}$ can then only be 0. We prove below that the Games 5 and 6 cannot be distinguished under the SXDH assumption and we then have $\mathtt{Adv}_6 \geq \mathtt{Adv}_5 - \mathtt{Adv}_{\mathsf{SXDH}}$.

Proof (of indistinguishability between anonymity Games 5 and 6). $\mathcal{R}$ receives a DDH challenge (g, g^a, g^b, g^z) in $\mathbb{G}_1$ and must then decide whether $z = a \cdot b$. It will then act as if $y = a$ for the secret key usk_{i^*} of user i^*. This is not a problem since g^a is sufficient to issue group signatures and to join the group since Game 4. Moreover Game 5 ensures $\mathcal{R}$ is able to answer any $\mathcal{O}\mathtt{Open}$ query, even without knowing $\widetilde{g}^a$.

To answer the $\mathcal{O}\mathtt{Ch}$ query for a message m, it selects a random scalar t and computes a group signature σ as follows:

- $\tau_1 \leftarrow ((g^b)^{\alpha_1} \cdot (g^z)^{\alpha_2})^t$;
- $(\tau_2, \widetilde{\tau}) \leftarrow (g^{1/t}, \widetilde{g}^{1/t})$;
- $(\sigma_1, \sigma_2) \leftarrow (g^b, (g^b)^{x/h(\widetilde{\tau}||\sigma_1||m)} \cdot (g^z))$.

In any case, σ is a valid group signature on m, *i.e.* $\mathtt{Verify}(\mathsf{gpk}, \sigma, m)$ outputs the bit 1. If $z = a \cdot b$, then σ s a valid signature issued by user i^* and $\mathcal{A}$ is still playing Game 5. Else, σ is a signature issued with a random key, independent of a and $\mathcal{A}$ is playing Game 6. Any change of behavior of $\mathcal{A}$ between these two games can then be used against the DDH assumption in $\mathbb{G}_1$ and so against the SXDH assumption. □

We get the following result:

- $\varepsilon/q_A \leq 2\ \mathtt{Adv}_{\mathsf{SXDH}} + \mathtt{Adv}_{\mathsf{IND-CCA2}}$ for any adversary succeeding against CPA anonymity with probability ε;
- $\varepsilon/q_A \leq 2\ \mathtt{Adv}_{\mathsf{SXDH}} + \mathtt{Adv}_{\mathsf{IND-CCA2}} + \mathtt{Adv}^{nf}$ for any adversary succeeding against selfless anonymity with probability ε;

which proves both CPA anonymity and selfless anonymity of our construction.

Remark 7. Let us now consider the case where the group manager and the opening authority are merged, as in the BMW model [5]. As explained in Remark 4, the use of IND-CCA2 encryption during the `Join` protocol is only necessary when the opening authority is not involved in this process, which is no longer the case here. We can then discard Γ and remove Game 4 in the security proof.

In Game 5, $\mathcal{R}$ proceeds as follows. It still stores the signatures generated on behalf of i^* in **Sig** but now answers $\mathcal{O}\mathtt{Open}$ queries on (σ, m) as follows:

- if $\sigma \in$ **Sig**, then it returns i^* along with a simulated proof π;
- if $\mathtt{Open}(\sigma, m)$ returns (i, π) or 0, then it forwards this answer to the adversary;
- if $\mathtt{Open}(\sigma, m)$ returns $\perp$, then it returns i^* along with a simulated proof π.

We note that a problem may only occur in the third case if the adversary managed to submit a group signature that cannot be traced back to a registered user. However, this would mean that $\mathcal{A}$ is a valid adversary against traceability, which is unlikely.

All the other games remain unchanged so $\varepsilon/q_A \leq 2\ \mathtt{Adv}_{\mathsf{SXDH}} + \mathtt{Adv}^{tra}$ for any adversary succeeding against full anonymity with probability ε in BMW [5].

A.2 Proof of Traceability

We prove here that any untraceable group signature can be used to construct a forgery against the FHS equivalence-class signature scheme. More specifically, let $\mathcal{A}$ be an adversary against the traceability of our group signature succeeding with probability ε, then $\mathcal{A}$ can be converted into an adversary succeeding against the EUF-CMA security of FHS signature with the same probability.

Technically, $\mathcal{A}$ can succeed by returning a valid signature σ on m that either foils the opening process or that can be opened but for which it is impossible to produce a valid proof of opening. We can exclude the latter in our construction because of the correctness and of the soundness of Groth-Sahai proofs.

Our reduction $\mathcal{R}$ generates the public parameters as usual except that it does not discard x after generating X and $\widetilde{X}$. It then gets the public key $\widetilde{A}_1$ and $\widetilde{A}_2$ from the EUF-CMA challenger and sets the group manager's public key as $(\widetilde{A}_1, \widetilde{A}_2, \widetilde{A}_2^x)$. By using its signing oracle, it is able to handle `Join` query so the simulation is perfect. At the end of the game, $\mathcal{A}$ then outputs with probability ε an untraceable group signature σ on m. If we parse σ as $(\tau_1, \tau_2, \widetilde{\tau}, \sigma_1, \sigma_2)$, this means that:

(1.) $e\big(\sigma_1, \widetilde{A}_1 \widetilde{B}^{-1/h(\widetilde{\tau}||\sigma_1||m)}\big) \cdot e(\sigma_2, \widetilde{A}_2) = e(\tau_1, \widetilde{\tau})$;

(2.) $e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau})$;
(3.) $e(\sigma_2, \widetilde{g}) \cdot e\big(\sigma_1, \widetilde{X}^{-1/h(\widetilde{\tau}||\sigma_1||m)}\big) \neq e(\sigma_1, \widetilde{g}^{y_i})$ for all $\widetilde{g}^{y_i}$ stored (encrypted).

Equation (1.) is equivalent to:

$$e(\sigma_1, \widetilde{A}_1) \cdot e\big(\sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}, \widetilde{A}_2\big) = e(\tau_1, \widetilde{\tau}),$$

which means (together with equation (2.)) that $(\tau_1, \tau_2, \widetilde{\tau})$ is a valid FHS signature on $\big(\sigma_1, \sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}\big)$. However, $(\tau_1, \tau_2, \widetilde{\tau})$ will be considered as a valid forgery only if $\big(\sigma_1, \sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}\big)$ does not belong to the equivalence class of a message submitted to the signing oracle.

Let $\mathcal{S} = \{(h_i, h_i^{y_i})\}_{i=1}^{q}$, for $h_i \in \mathbb{G}_1$ be the set of queried messages. All these messages were involved in a `Join` query during which the group manager received (and stored) $\widetilde{g}^{y_i}$ (encrypted). Therefore, if (μ_1, μ_2) belongs to the equivalent class of an element of $\mathcal{S}$, then there exists $i \in [1, q]$ such that $e(\mu_1, \widetilde{g}^{y_i}) = e(\mu_2, \widetilde{g})$.

Let us then assume that $\big(\sigma_1, \sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}\big)$ satisfies the previous condition, *i.e.* that there is i such that: $e(\sigma_1, \widetilde{g}^{y_i}) = e\big(\sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}, \widetilde{g}\big)$. We then have $e(\sigma_1, \widetilde{g}^{y_i}) = e(\sigma_2, \widetilde{g}) \cdot e\big(\sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}, \widetilde{g}\big) = e(\sigma_2, \widetilde{g}) \cdot e\big(\sigma_1, \widetilde{X}^{-1/h(\widetilde{\tau}||\sigma_1||m)}\big)$, which contradicts equation (3.). The pair $\big(\sigma_1, \sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}\big)$ has then never been signed, nor any representative of the same equivalence class, which means that $(\tau_1, \tau_2, \widetilde{\tau})$ along with $\big(\sigma_1, \sigma_2 \cdot \sigma_1^{-x/h(\widetilde{\tau}||\sigma_1||m)}\big)$ is a valid forgery against the EUF-CMA security of the FHS scheme.

A.3 Proof of Non-Frameability

A successful adversary $\mathcal{A}$ against the non-frameability of our scheme is able to forge a valid group signature $\sigma = (\tau_1, \tau_2, \widetilde{\tau}, \sigma_1, \sigma_2)$ on a message m that can be traced back to some honest user i. We then distinguish four cases:

- Case 1: the pair $(g^u, g^{u \cdot y})$ stored in $\mathbf{Reg}[i]$ is not the one sent by user i when he joined the group;
- Case 2: the user i has issued a signature $(\tau_1', \tau_2', \widetilde{\tau}', \sigma_1', \sigma_2')$ on m' such that $(\widetilde{\tau}', \sigma_1', m') \neq (\widetilde{\tau}, \sigma_1, m)$ but $h(\widetilde{\tau}||\sigma_1||m) = h(\widetilde{\tau}'||\sigma_1'||m')$;
- Case 3: the user i has issued a signature $(\tau_1', \tau_2', \widetilde{\tau}', \sigma_1', \sigma_2')$ on m' such that $(\widetilde{\tau}', \sigma_1', m') = (\widetilde{\tau}, \sigma_1, m)$;
- Case 4: none of the previous events occurred.

We recall that a user can be accused of having produced a signature only if the opening is valid, *i.e.* only if the `Judge` algorithm outputs 1. One step of the `Judge` algorithm is to check that the user i has indeed signed the elements stored in $\mathbf{Reg}[i]$, *i.e.* that there is a signature μ in $\mathbf{Reg}[i]$ such that $\Sigma.\mathtt{Verify}(\mathsf{pk}_i, \mu, (g^u||g^{u \cdot y}||C_i||\pi_i)) = 1$. Therefore, Case 1 straightforwardly implies a forgery against Σ, which is assumed to be EUF-CMA.

Similarly, Case 2 directly implies a collision of the hash function h.

Case 3. Let us now consider the third case. As the algorithm `Judge` returns 1, σ is valid, which means that:

(1.) $e(\sigma_1, \widetilde{A}_1 \widetilde{B}^{-1/h(\widetilde{\tau}||\sigma_1||m)}) \cdot e(\sigma_2, \widetilde{A}_2) = e(\tau_1, \widetilde{\tau})$,

(2.) $e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau})$.

We then have $e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau}) = e(g, \widetilde{\tau}') = e(\tau_2', \widetilde{g})$ and thus $\tau_2 = \tau_2'$. Moreover, we know that $e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1, \widetilde{X}^{-1/h(\widetilde{\tau}||\sigma_1||m)}) = e(\sigma_1, \widetilde{g}^{y_i})$ because σ is traced back to i. If we rewrite this equation, we have

$$e(\sigma_2, \widetilde{g}) = e(\sigma_1, \widetilde{X}^{1/h(\widetilde{\tau}||\sigma_1||m)} \cdot \widetilde{g}^{y_i}) = e(\sigma_1', \widetilde{X}^{1/h(\widetilde{\tau}'||\sigma_1'||m')} \cdot \widetilde{g}^{y_i}) = e(\sigma_2', \widetilde{g})$$

and so $\sigma_2 = \sigma_2'$. Combining this equality with (1.), we also get $\tau_1 = \tau_1'$. Therefore, σ is exactly the signature $(\tau_1', \tau_2', \widetilde{\tau}', \sigma_1', \sigma_2')$ returned by the $\mathcal{O}\mathtt{Sign}$ algorithm and so cannot constitute a valid attack against non-frameability.

Case 4. We now consider Case 4 and prove that a successful adversary breaks the MPS assumption. Let $(\mathsf{g}, \mathsf{g}^y, \mathsf{g}^z, \mathsf{g}^{z \cdot x}, \widetilde{\mathsf{g}}, \widetilde{\mathsf{g}}^x, \widetilde{\mathsf{g}}^y)$ be a MPS instance for the function h defined in pp. The reduction $\mathcal{R}$ generates the public parameters pp as usual except that it sets $(g, X) = (\mathsf{g}^z, \mathsf{g}^{z \cdot x})$ and $(\widetilde{g}, \widetilde{X}) = (\widetilde{\mathsf{g}}, \widetilde{\mathsf{g}}^x)$. It then generates the keys $(\mathsf{osk}, \mathsf{opk})$ and $(\mathsf{gsk}, \mathsf{gpk})$ as in section 5, selects $i^* \in [1, q_A]$ (where q_A is a bound on the number of $\mathcal{O}\mathtt{Add}$ queries) and handles the oracles queries as follows:

- $\mathcal{O}\mathtt{Add}(i)$: $\mathcal{R}$ answers to every query normally with $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \Sigma.\mathtt{Keygen}$.
- $\mathcal{O}\mathtt{Cor}(i)$: $\mathcal{R}$ returns requested secret keys if $i \neq i^*$ and aborts otherwise.
- $\mathcal{O}\mathtt{J}_U(i)$: $\mathcal{R}$ proceeds normally if $i \neq i^*$ by generating the necessary secret values. Otherwise, it encrypts $\widetilde{\mathsf{g}}^y$ as C and sends $(\mathsf{g}, \mathsf{g}^y)$ along with the corresponding NIZK proof π. We note that $(\mathsf{g}, \mathsf{g}^y)$ is a valid pair for $\widetilde{\mathsf{g}}^y$ (so the NIZK proof can be generated normally) which implicitly defines u as z^{-1}.
- $\mathcal{O}\mathtt{Sign}(i, m)$: here we only consider the case where $i = i^*$ since $\mathcal{R}$ can normally generate this signature otherwise. $\mathcal{R}$ forwards the message m to the MPS oracle $\mathcal{O}$ which returns $(s, \widetilde{\mathsf{g}}^{r \cdot s}, \mathsf{g}^r, \mathsf{g}^{r \cdot z}, \mathsf{g}^{r(x+t \cdot y)})$ with $t = h(\widetilde{\mathsf{g}}^{r \cdot s}||\mathsf{g}^r||m)$. $\mathcal{R}$ then constructs the following group signature σ:
  - $(\sigma_1', \sigma_2') \leftarrow (\mathsf{g}^r, (\mathsf{g}^{r(x+t \cdot y)})^{1/t})$;
  - $(\tau_2', \widetilde{\tau}') \leftarrow ((\mathsf{g}^{r \cdot z})^s, (\widetilde{\mathsf{g}}^r)^s)$;
  - $\tau_1' \leftarrow (\mathsf{g}^{\alpha_1} \cdot (\mathsf{g}^y)^{\alpha_2})^{1/s})$;

  where $(\alpha_1, \alpha_2) = \mathsf{osk}$. This is indeed a valid group signature since:

$$\begin{aligned} & e(\sigma_1', \widetilde{A}_1 \widetilde{B}^{-1/h(\widetilde{\tau}'||\sigma_1'||m)}) \cdot e(\sigma_2', \widetilde{A}_2) \\ & \quad = e(\mathsf{g}^r, \widetilde{\mathsf{g}}^{\alpha_1 - (\alpha_2 \cdot x)/h(\widetilde{\mathsf{g}}^{r \cdot s}||\mathsf{g}^r||m)}) \cdot e(\mathsf{g}^{r(x/h(\widetilde{\mathsf{g}}^{r \cdot s}||\mathsf{g}^r||m)+y)}, \widetilde{\mathsf{g}}^{\alpha_2}), \\ & \quad = e(\mathsf{g}^r, \widetilde{\mathsf{g}}^{\alpha_1}) \cdot e(\mathsf{g}^{r \cdot y}, \widetilde{\mathsf{g}}^{\alpha_2}), \\ & \quad = e(\tau_1', \widetilde{\tau}'), \end{aligned}$$

  and $e(\tau_2', \widetilde{g}) = e((\mathsf{g}^{r \cdot z})^s, \widetilde{\mathsf{g}}) = (\mathsf{g}^z, \widetilde{\mathsf{g}}^{r \cdot s}) = e(g, \widetilde{\tau}')$. Moreover, σ is correctly distributed as this construction implicitly defines r and s of the $\mathtt{Sign}$ algorithm as r/z and $1/rs$ from the MPS instance, where r and s are random.

We note that $\mathcal{R}$ is perfectly able to answer any adversary query as long as its guess on i^* is correct, which occurs with probability at least $1/q_A$. In this case, $\mathcal{A}$ returns, with some probability ε, a valid signature $\sigma = (\tau_1, \tau_2, \widetilde{\tau}, \sigma_1, \sigma_2)$ on some message m that can be traced back to i^*. Concretely, this means that σ satisfies the following relations:

- $e(\sigma_1, \widetilde{A}_1 \widetilde{B}^{-1/t}) \cdot e(\sigma_2, \widetilde{A}_2) = e(\tau_1, \widetilde{\tau})$;

- $e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau})$;
- $e(\sigma_2, \widetilde{g}) \cdot e(\sigma_1, \widetilde{X}^{-1/t}) = e(\sigma_1, \widetilde{g}^y)$;

with $t = h(\widetilde{\tau}||\sigma_1||m)$. Since we here consider the Case 4, we know that the value of t is different from the ones used by $\mathcal{R}$ to answer $\mathcal{O}\texttt{Sign}$ queries. Moreover, the third equation means that:

$$e(\sigma_2^t, \widetilde{\mathsf{g}}) = e(\sigma_2^t, \widetilde{g}) = e(\sigma_1, \widetilde{X} \cdot (\widetilde{g}^y)^t) = e(\sigma_1, \widetilde{\mathsf{g}}^x \cdot (\widetilde{\mathsf{g}}^y)^t)$$

which implies that $(t, \sigma_1, \sigma_2^t)$ is a valid solution to the MPS problem. Any adversary corresponding to Case 4, succeeding with probability ε, can then be converted into an adversary against the MPS assumption succeeding with probability at least ε/q_A. Case 4 thus occurs with negligible probability under the MPS assumption.

A.4 Assessment of the MPS Assumption

The goal of an adversary $\mathcal{A}$ against the MPS assumption is to forge a new PS signature with the help of an oracle $\mathcal{O}$ that outputs more elements than in the original PS assumption. As we discuss in subsection 3.1, these new elements do not seem provide any advantage to $\mathcal{A}$. We formalize this intuition with an evaluation of the MPS assumption in the generic group model.

Lemma 8. *In the generic group model, no adversary can succeed against the* MPS *assumption with probability greater than* $O((4q_{\mathcal{O}} + 7 + q_G)^2/p)$*, where* q_G *is a bound on the group oracle queries and* $q_{\mathcal{O}}$ *is a bound on the number of queries to* $\mathcal{O}$*.*

Proof. The adversary has access to $(g, g^y, g^z, g^{z \cdot x}, \widetilde{g}, \widetilde{X}, \widetilde{Y})$ along with the outputs $(s_i, \widetilde{g}^{r_i \cdot s_i}, g^{r_i}, g^{r_i \cdot z_i}, g^{r_i(x_i + t_i \cdot y_i)})$, with $t_i = h(\widetilde{g}^{r_i \cdot s_i}||g^{r_i}||m_i)$, of the oracle $\mathcal{O}$. We will first prove that $\mathcal{A}$ cannot symbolically produce a valid tuple $(t^*, \sigma_1, \sigma_2) \in \mathbb{Z}_p \times \mathbb{G}_1^2$ by associating each group element with a polynomial whose formal variable are x, y, z and r_i, $\forall i \in [1, q_{\mathcal{O}}]$.

In the generic group model, any group element must have been generated through queries to the oracle of the internal law of the group or to $\mathcal{O}$. Since the goal of $\mathcal{A}$ is to output elements σ_1 and σ_2 of $\mathbb{G}_1$, the latter can only be combinations of elements from the same group. This means that there are known coefficients $\{a, b, c, d, \alpha_i, \beta_i, \gamma_i\}$ and $\{a', b', c', d', \alpha_i', \beta_i', \gamma_i'\}$, for $i \in [1, q_{\mathcal{O}}]$, such that:

$$\tau_1 = g^a \cdot (g^y)^b \cdot (g^z)^c \cdot (g^{z \cdot x})^d \cdot \prod_i (g^{r_i})^{\alpha_i} \cdot (g^{r_i \cdot z_i})^{\beta_i} \cdot (g^{r_i(x + t_i \cdot y)})^{\gamma_i};$$
$$\tau_2 = g^{a'} \cdot (g^y)^{b'} \cdot (g^z)^{c'} \cdot (g^{z \cdot x})^{d'} \cdot \prod_i (g^{r_i})^{\alpha_i'} \cdot (g^{r_i \cdot z_i})^{\beta_i'} \cdot (g^{r_i(x + t_i \cdot y)})^{\gamma_i'}.$$

A tuple $(t^*, \sigma_1, \sigma_2)$ is valid only if $e(\sigma_1, \widetilde{X} \cdot \widetilde{Y}^{t^*}) = e(\sigma_2, \widetilde{g})$, $\sigma_1 \neq 1_{\mathbb{G}_1}$ and $t^* \neq t_i$ for all $i \in [1, q_{\mathcal{O}}]$. This gives the following relation:

$$(a + by + cz + dzx + \sum_i \alpha_i r_i + \beta_i r_i z + \gamma_i r_i (x + y t_i))(x + y t^*)$$
$$= (a' + b'y + c'z + d'zx + \sum_i \alpha_i' r_i + \beta_i' r_i z + \gamma_i' r_i (x + y t_i)).$$

The factor $(x + yt^*)$ implies that any monomial of the left member will be of degree at least 1 in x or y. We can therefore conclude that $a' = c' = \alpha'_i = \beta'_i = 0$, $\forall i \in [1, q_{\mathcal{O}}]$, leading to the following equation:

$$(a + by + cz + dzx + \sum_i \alpha_i r_i + \beta_i r_i z + \gamma_i r_i (x + yt_i))(x + yt^*)$$
$$= (b'y + d'zx + \sum_i \gamma'_i r_i (x + yt_i)).$$

The right member does not contain any monomial of degree 2 in x or y, which implies that $b = d = \gamma_i = 0$, $\forall i \in [1, q_{\mathcal{O}}]$:

$$(a + cz + \sum_i \alpha_i r_i + \beta_i r_i z)(x + yt^*) = (b'y + d'zx + \sum_i \gamma'_i r_i (x + yt_i)).$$

If we develop the left member, we get monomials such as ax, czy and $\beta_i r_i zx$. However, there are no similar terms in the right member, which implies that $a = c = \beta_i = 0$, $\forall i$:

$$(\sum_i \alpha_i r_i)(x + yt^*) = (b'y + d'zx + \sum_i \gamma'_i r_i (x + yt_i)).$$

Any term of the left member is then of degree 1 in some r_i, which implies that $b' = d' = 0$:

$$(\sum_i \alpha_i r_i)(x + yt^*) = (\sum_i \gamma'_i r_i (x + yt_i)).$$

We then have, for each $i \in [1, q_{\mathcal{O}}]$, $\alpha_i(x + yt^*) = \gamma'_i(x + ty_i)$, which implies that $\alpha_i = \gamma'_i$. However, we also know that $t^* \neq t_i$ $\forall i \in [1, q_{\mathcal{O}}]$ which means that $\alpha_i = \gamma'_i = 0$. $\mathcal{A}$ has thus returned a tuple $(t^*, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1})$, which is invalid.

Now, let us assess the probability of an accidental validity, *i.e.* when two different polynomials involved in group oracle queries evaluate to the same value. All the polynomials considered in this proof are of degree at most 2. Since there are at most $4q_{\mathcal{O}} + 7 + q_G$ polynomials, there are at most $(4q_{\mathcal{O}} + 7 + q_G)^2/2$ pairs that could evaluate to the same value. By the Schwartz-Zippel lemma, this can occur with probability at most $(4q_{\mathcal{O}} + 7 + q_G)^2/p$, which is negligible. □

B Additional Primitives

B.1 Public Key Encryption

We marginally use an encryption scheme Γ defined by the following algorithms:

- $\texttt{Keygen}(1^\lambda)$: Outputs a pair of decryption and encryption keys $(\mathsf{sk}, \mathsf{pk})$.
- $\texttt{Encrypt}(\mathsf{pk}, m)$: Outputs a ciphertext c of message m under pk.
- $\texttt{Decrypt}(\mathsf{sk}, c)$: On input the decryption key sk and a ciphertext c, this algorithm outputs a message m or $\perp$.

Several security notions exist for public key encryption but our group signature scheme will only need *indistinguishability under adaptive chosen ciphertext attacks* (IND-CCA2). Informally it requires that no adversary, even given access to a decryption oracle, is able to distinguish an encryption of m_0 from an encryption of m_1, where m_0 and m_1 are two messages chosen by the adversary.

B.2 Groth-Sahai proof system

In [24], Groth and Sahai proposed a non-interactive proof system, in the common reference string (CRS) model, which captures most of the relations for bilinear groups. There are two types of setup for the CRS that yield either perfect soundness or perfect witness indistinguishability, while being computationally indistinguishable (under the SXDH assumption, in our setting). To prove that some variables satisfy a set of relations, the prover commits to them (by using the elements from the CRS) and then computes one proof element per relation. Efficient non-interactive witness indistinguishable proofs are available for

- pairing-product equations, for variables $\{X_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{Y}_i\}_{i=1}^n \in \mathbb{G}_2$ and constants $t_T \in \mathbb{G}_T$, $\{A_i\}_{i=1}^n \in \mathbb{G}_1$, $\{\widetilde{B}_i\}_{i=1}^n \in \mathbb{G}_2$, $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}/p\mathbb{Z}$:

$$\prod_{i=1}^{n} e(A_i, \widetilde{Y}_i) \prod_{i=1}^{n} e(X_i, \widetilde{B}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(X_i, \widetilde{Y}_j)^{a_{i,j}} = t_T;$$

- multi-exponentiation equations, for variables $\{X_i\}_{i=1}^n \in \mathbb{G}_k$, $\{y_i\}_{i=1}^n \in \mathbb{Z}/p\mathbb{Z}$ and constants $T \in \mathbb{G}_k$, $\{A_i\}_{i=1}^n \in \mathbb{G}_k$, $\{b_i\}_{i=1}^n \in \mathbb{Z}/p\mathbb{Z}$, $\{a_{i,j}\}_{i,j=1}^n \in \mathbb{Z}/p\mathbb{Z}$:

$$\prod_{i=1}^{n} A_i^{y_i} \prod_{j=1}^{n} X_j^{b_j} \prod_{i=1}^{n} \prod_{j=1}^{n} X_j^{y_i \cdot a_{i,j}} = T, \qquad \text{where } k \in \{1, 2\}.$$

The Groth-Sahai framework also supports non-interactive zero-knowledge (NIZK) proofs for multi-exponentiation equations or for pairing-product equations such that $t_T = 1_{\mathbb{G}_T}$ or $t_T = \prod e(A_i, \widetilde{B}_i)$.