Membership Privacy for Fully Dynamic Group Signatures

(Full Version)

Michael Backes[1,3], Lucjan Hanzlik[2,3], and Jonas Schneider (✉)[2,3]

[1] CISPA Helmholtz Center i.G.,
`backes@cispa.saarland`
[2] CISPA, Saarland University,
`{hanzlik, jonas.schneider}@cispa.saarland`
[3] Saarland Informatics Campus

Abstract. Group signatures present a trade-off between the traditional goals of digital signatures and the signer's desire for privacy, allowing for the creation of unforgeable signatures in the name of a group which reveal nothing about the actual signer's identity beyond their group membership. Considering the desired properties formally opens up a possibility space of different security goals under various assumptions on trust placed in the designated entities of any scheme. Many models differ in their consideration of the variability of group membership as well, yet a formal treatment of the *privacy of group membership status* is lacking in all models, thus far.
We address this issue, starting from the vantage point of the comprehensive model due to Bootle et al. (ACNS'16), who prove that any scheme secure in their model is also secure in the previous models. Their model allows for fully dynamic management of group membership by segmenting the scheme's lifetime into epochs during which group membership is static but between which users may join or leave the group.
We extend the model of Bootle et al. by introducing formal notions of membership privacy. We then propose an efficient generic construction for a fully dynamic group signature scheme with membership privacy that is based on signatures with flexible public key (SFPK) and signatures on equivalence classes ($\mathsf{SPS\text{-}EQ}$). We instantiate the construction using a SFPK scheme based on the bilinear decisional Diffie-Hellman assumption and $\mathsf{SPS\text{-}EQ}$ scheme by Fuchsbauer and Gay (PKC'18). The resulting scheme provides shorter signatures than existing schemes from standard assumption, while at the same time achieving stronger security guarantees.

1 Introduction

The concept of group signatures was first introduced by Chaum and van Heyst in [21]. It considers a group of signers, over seen by a group manager who decides group membership. Informally, the idea is that a group member may issue a signature on behalf of the entire group. The signature is publicly verifiable, yet it is infeasible to determine which member of the group has created it. Chaum and van Heyst require also that a group signature may be opened by a relevant authority, so that the identity of the actual signer becomes known in case of abuse.

The first formal security model and a construction of group signatures from general assumptions were introduced by Bellare, Micciancio and Warinschi (BMW) in [8]. In this model, group members are fixed during a one-time group setup phase and receive honestly generated signing keys from the group manager, who is also responsible for the opening of signatures.

The idea of group signatures was further extended by Bellare, Shi, Zhang (BSZ) in [9] and by Kiayias, Yung (KY) in [34, 35]. These new models allow dynamic enrollment of group members after the initial setup is finished and replace the monolithic group manager by separate issuing and opening authorities. The main differences between the BSZ and KY models are related to the interactive user enrollment and the opening soundness property. In particular, in the BSZ model after a successful enrollment of a user the issuer publishes the final state of the procedure, whereas in the KY model the entire communication transcript is published. In contrast to the BSZ model, the KY model opening authority does not have to prove that the opening has been executed honestly. Because of this, schemes in the KY model require additional trust in the opening authority since it could potentially maliciously point to a user who did not create the signature. This property is called opening soundness and has been studied by Sakai et al. in [40].

Bootle et al. [15] introduced fully-dynamic group signatures. These new results additionally address revocation, opening soundness and maliciously generated keys. The model allows group members to dynamically join and leave the group. To properly model this feature, the definition introduces an additional public epoch information that is published by the manager.

The authors show that any scheme secure in their model is also secure in all previous ones, which proves that this is the state-of-the-art model for group signatures.

Related Work. The generic constructions from [8] and [9] established a design paradigm, which is sometimes called the *sign-and-encrypt-and-prove* paradigm (SEP). It is used in a number of constructions and may be informally described as follows: a signature consists of an encryption under the opener's public key of both a signature of the message under the member's signing key and the member's identity, as well as a non-interactive zero-knowledge proof that the identity contained in the encryption is valid and is indeed that of the signer of the message. The identity of the group member is typically a signature issued by the group manager. Thus, relying on the unforgeability of this signature, such a group signature scheme achieves non-frameability and traceability. Beside this design paradigm and generic construction, which are also based on this paradigm, Abdalla and Warinschi proved in [2] that group signatures are actually equivalent to IND-CPA secure encryption schemes.

In [10], Bichsel et al. identify the SEP design paradigm as a source of inefficiency in group signatures. Then they propose a new approach based on re-randomizable signature schemes and provide an efficient construction without encryption secure in the random oracle model. In our paper we follow that idea, however we do not rely on the random oracle model to prove security of our scheme. By now many group signature schemes were designed for both the static and dynamic case in the random oracle model which utilize the RSA cryptosystem [4], [41], [19], [33], discrete logarithm setting [5], [26], and bilinear setting [11], [20].

One of the first standard model constructions was introduced by Ateniese et al. [3]. The scheme is highly efficient, it utilizes bilinear maps and the signature consists only of 8 group elements. However, the scheme does not provide full-anonymity in sense of the definition in the BMW model [8]. In particular, the adversary is not allowed to see the private keys of honest users.

Boyen and Waters [17, 18] proposed standard model schemes that use composite order bilinear groups, but in contrast to [3] allows key exposure attacks. However, the adversary cannot see any openings of signatures. This restricted version of full-anonymity is also called CPA-*anonymity*.

The introduction of the Groth-Sahai (GS) proof system [30] allowed for the design of new and efficient group signature schemes in the standard model. Groth [29] was the first to introduce a standard model group signature with a constant size public key and signatures, which preserve the full-anonymity property. The security of the scheme relies on a q-type assumption. The GS proof system was also used by Libert et al. [37, 36], who designed standard model group signatures with revocation capabilities.

At Crypto'15 Libert, Peters and Yung [38] introduced two efficient group signature schemes that rely on simple assumptions. The first scheme is secure in the static BMW model [8]. On the other hand, the second construction is less efficient, but secure in the dynamic security model from [35].

Bootle et al. [16] propose a generic construction of group signatures from accountable ring signatures. They instantiate it using a scheme based on a sigma protocol in the random oracle model. Later, Bootle et al. [15] show that this construction is a fully dynamic group signature scheme. The idea is to include the description of the ring as part of the epoch information. This way only users in the ring are member of the group in the current epoch. Security follows directly from the security of accountable ring signatures.

Derler and Slamanig proposed a generic construction for dynamic group signatures based on structure preserving signatures on equivalence classes ($\mathsf{SPS\text{-}EQ}$) [23]. $\mathsf{SPS\text{-}EQ}$ define a relation $\mathcal{R}$ that induces a partition on the message space. By signing one representative of a partition, the signer in fact signs the whole partition. Then, without knowledge of the secret key we can transform the signature to a different representative of the partition. Their group signatures make use of signatures of knowledge (as part of the group signature) and non-interactive zero-knowledge proof systems (in the issuing procedure and to ensure opening soundness). The authors present an efficient instantiation in the random oracle model. The main disadvantage of their construction is that there currently exists no standard model instantiation.

Recently, Backes et al. [6] introduced a new cryptographic primitive called signatures with flexible public key. The idea is similar to $\mathsf{SPS\text{-}EQ}$, but instead of partitioning the message space in this case the partition is on the public key space. In other words, signers can randomize their public key and secret key to a different representative of the same equivalence class and create a signature that is valid under the new public key. Whether public keys are in the same relation can only be checked using a trapdoor. This primitive also allows third parties to randomize the public key of the signer and introduces a recovery algorithm for the signer to compute the corresponding secret key. The authors also show how to combine their primitive with $\mathsf{SPS\text{-}EQ}$ to construct static group signatures, which are secure in the BMW model [8].

Group signatures can also be constructed from lattice-based assumptions [39] or symmetric primitives [12]. The former is the only scheme secure under lattice-based assumptions for which the signature size does not depend on the number of group members. Unfortunately, it is only secure in the partially dynamic model [9] and in the random oracle model. The latter scheme is also instantiated in the random oracle model.

Contribution. In this paper we revisit the fully-dynamic group signature framework by Bootle et al. [15]. We notice that the epoch information published e.g. with each change in the group (joining or leaving of a member) may leak the identities of members. In particular, this problem is visible for the scheme proposed in [16], where the epoch information contains the description of a ring of active members. It is easy to see that since this epoch information is required to verify a signature, the list of active members of the group is known. This can be an issue when group signatures are used in an access control system for resources. An adversary can distinguish the members of the group, which have access to a particular resource and perform targeted attacks e.g. phishing or DoS. We stress that group membership information is implicitly considered public in most security models.

Therefore we propose an extension to the existing model that ensures a feature which we call *membership privacy*. We model it using two new security experiments. The first one considers the privacy of users joining the group and second one of users leaving the group. In both cases the adversary is outside the system but can corrupt a subset of users. His task is to distinguish which of two users left or joined the group. These basic definitions protect against the attack described above. Note that in this case we consider both managers to be honest. It is obvious that the group manager always knows the identities of members. The tracing authority can, by definition, determine the identity of group members who have created a signature. Hence we cannot membership information for a given epoch from the tracing manager unless the user effectively does not participate in the epoch.

The second contribution of this paper is a generic construction of fully-dynamic group signatures which achieve membership privacy. We build upon ideas from [16], [23] and [6], and extend them to ensure security in this stronger model. Each epoch the group manager uses a fresh instance of SPS-EQ to certify the public keys of members. However, instead of using the original public keys in the epoch information, the group manager first randomizes the public key and encrypts the randomization using the signer's public key for an encryption scheme. Members can decrypt the randomization and use the SPS-EQ signature from the epoch information. Additionally, the signer creates a proof of knowledge of a unique representative of the equivalence class and the randomness used by the signer. This unique representative can be extracted by the tracing authority and used to identify the signer because the unique representative is also used as the signer's global public key. Membership privacy is ensured because the group manager randomizes the published public key list.

Lastly we show how to efficiently instantiate our construction under standard assumptions without relying on the random oracle model. The resulting scheme has shorter signature than state-of-the-art schemes [38, 30] that are secure in the same setting but only allow for partially-dynamic groups.

To achieve this efficient instantiation we introduce a new signatures with flexible public key scheme that is secure under the bilinear decisional Diffie-Hellman assumption and the decisional Diffie-Hellman assumption in $\mathbb{G}_1$. We also revisit the definitions introduced by Backes et al [6]. We propose a notion called *canonical representative*. Informally, we define a unique representative for every equivalence class. This new feature finds application in our group signature construction. Our scheme has shorter public keys than the schemes from [6] i.e. 2 group elements in $\mathbb{G}_1$, which allows us to more efficiently instantiate our group signature construction.

The results presented in this paper can be summarized as follows.

1. We identify a privacy issue in the fully-dynamic group signatures definition by Bootle et al. [15].
2. We extend the existing definition to ensure membership privacy (protecting the identities of group members).
3. We propose a generic construction of fully-dynamic group signatures with membership privacy that uses signatures with flexible public key and signatures on equivalence classes as building blocks.
4. We introduce an efficient signature with flexible public key scheme that is secure under standard assumptions and refine the definitions proposed by Backes et al. In particular, we identity a natural notion called canonical representative, which was not discussed in the original paper [6].
5. We use this scheme and the SPS-EQ by Fuchsbauer and Gay [25] to efficiently instantiate our group signature construction. The resulting scheme has a shorter signature size than state-of-the-art schemes with comparable assumption but which are secure in weaker models.
6. We discuss the efficiency of our instantiation and compare them with state-of-the-art schemes in a similar setting.

2 Preliminaries

Definition 1 (Bilinear map). *Let us consider cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of prime order p. Let g_1, g_2 be generators of respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. We call $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a* bilinear map *(pairing) if it is efficiently computable and the following holds:*

Bilinearity: $\forall (S,T) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\forall a,b \in \mathbb{Z}_p$, *we have* $e(S^a, T^b) = e(S,T)^{a \cdot b}$,
Non-degeneracy: $e(g_1, g_2) \neq 1$ *is a generator of group* $\mathbb{G}_T$,

Depending on the choice of groups we say that map e is of type 1 if $\mathbb{G}_1 = \mathbb{G}_2$, of type 2 if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, of type 3 if no such isomorphism ψ is known.

Definition 2 (Bilinear-group generator). *A bilinear-group generator is a deterministic polynomial-time algorithm* BGGen *that on input a security parameter* 1^λ *returns a bilinear group* $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ *such that* $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$ *and* $\mathbb{G}_T$ *are groups of order* p *and* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *is a bilinear map.*

2.1 Assumptions

Definition 3 (Decisional Diffie-Hellman Assumption in $\mathbb{G}_i$). *Given* BG *and elements* $(g_i^a, g_i^b, g_i^z) \in \mathbb{G}_i^3$ *it is hard for all PPT adversaries* $\mathcal{A}$ *to decide whether* $z = a \cdot b \mod p$ *or* $z \xleftarrow{\$} \mathbb{Z}_p^*$. *We will use* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{ddh}}(\lambda)$ *to denote the advantage of the adversary in solving this problem.*

If the problem instance were given in both groups, i.e. $(g_1^a, g_1^b, g_1^z, g_2^a, g_2^b, g_2^z)$ then the pairing would allow to efficiently check $e(g_1^a, g_2^b) = e(g_1^z, g_2)$, solving the problem. An analogous problem, which is assumed difficult even in the presence of a pairing is given by adding values g_1^c, g_2^c to the challenge and asking whether $z = a \cdot b \cdot c \mod p$. This was noted by Boneh and Franklin [13] who defined a similar problem called Weil decisional Diffie-Hellman problem in the type 1 setting. In their later work [14] it was renamed to bilinear decisional Diffie-Hellman assumption. We restate it for type 3 pairings as follows:

Definition 4 (Bilinear Decisional Diffie-Hellman Assumption). *Given* BG *and elements* $(g_1^a, g_1^b, g_1^c, g_1^z, g_2^a, g_2^b, g_2^c, g_2^z) \in \mathbb{G}_1^4 \times \mathbb{G}_2^4$ *it is hard for all PPT adversaries* $\mathcal{A}$ *to decide whether* $z = a \cdot b \cdot c \mod p$ *or* $z \xleftarrow{\$} \mathbb{Z}_p^*$. *We will use* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{bddh}}(\lambda)$ *to denote the advantage of the adversary in solving this problem.*

Definition 5 (Collision-Resistance). *We call a function* $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p^*$ *collision-resistant if it is hard for all PPT adversaries* $\mathcal{A}$ *to output two distinct message* m_1, m_2 *for which* $\mathsf{H}(m_1) = \mathsf{H}(m_2)$ *We will use* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{coll}}(\lambda)$ *to denote the advantage of the adversary in finding a collision for this hash function.*

2.2 Programmable Hash Functions

We now recall the definition of programmable hash functions introduced by Hofheinz and Kiltz [32]. In order to do so, we first define a *group hash function* for group $\mathbb{G}$ and output length $\ell = \ell(\lambda)$. A group hash function consists of two polynomial time algorithms $\mathsf{PHF.Gen}$ and $\mathsf{PHF.Eval}$. For a security parameter λ, the generation algorithm $K_{\mathsf{PHF}} \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda)$ outputs a key. This key can be used in the deterministic algorithm $\mathsf{PHF.Eval}$ to evaluate the hash function via $y \in \mathbb{G} \xleftarrow{\$} \mathsf{PHF.Eval}(K_{\mathsf{PHF}}, X)$, where $X \in \{0,1\}^\ell$.

Definition 6. *A group hash function is an* (m, n, γ, δ)*-programmable hash function if there are polynomial time algorithms* $\mathsf{PHF.TrapGen}$ *and* $\mathsf{PHF.TrapEval}$ *such that:*

– *For any* $g, h \in \mathbb{G}$ *the trapdoor algorithm* $(K'_{\mathsf{PHF}}, td) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ *outputs a key* K' *and trapdoor* td. *Moreover, for every* $X \in \{0,1\}^\ell$ *we have* $(a_X, b_X) \xleftarrow{\$} \mathsf{PHF.TrapEval}(td, X)$, *such that* $\mathsf{PHF.Eval}(K'_{\mathsf{PHF}}, X) = g^{a_X} h^{b_X}$.

– *For all $g, h \in \mathbb{G}$ and for $(K'_{\mathsf{PHF}}, td) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ and $K_{\mathsf{PHF}} \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda)$, the keys K_{PHF} and K'_{PHF} are statistically γ-close.*
– *For all $g, h \in \mathbb{G}$ and all possible keys K'_{PHF} from the range of $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$, for all $X_1, \ldots, X_m, Z_1, \ldots, Z_n \in \{0,1\}^\ell$ such that $X_i \neq Z_j$ for any i, j and for the corresponding $(a_{X_i}, b_{X_i}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(td, X_i)$ and $(a_{Z_i}, b_{Z_i}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(td, Z_i)$ we have*

$$\Pr[a_{X_1} = \cdots = a_{X_m} = 0 \quad \wedge \quad a_{Z_1} = \cdots = a_{Z_n} \neq 0] \geq \delta,$$

where the probability is over the trapdoor td that was produced along with key K'_{PHF}.

Hofheinz and Kiltz show that the function introduced by Waters [42] is a programmable hash function. For a key $K_{\mathsf{PHF}} = (h_0, \ldots, h_\ell) \in \mathbb{G}^{\ell+1}$ and message $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ the function is computed as $h_0 \cdot \prod_{i=1}^{\ell} h_i^{x_i}$. In particular, they prove that for any fixed $q = q(\lambda)$ it is a $(1, q, 0, 1/8 \cdot (\ell+1) \cdot q)$-programmable hash function.

2.3 Signatures on Equivalence Classes

We now recall the notion of signatures on equivalence classes introduced by Hanser and Slamanig [31]. The signing algorithm of the primitive $\mathsf{SPS.Sign}(\mathsf{sk}_{\mathsf{SPS}}, M)$ defines an equivalence relation $\mathcal{R}$ that induces a partition on the message space. A signer can simply sign one representative of the class to create a signature for the whole class. The signature can then be changed without the knowledge of the secret key to a different representative using the $\mathsf{SPS.ChgRep}(\mathsf{pk}_{\mathsf{SPS}}, M, \sigma_{\mathsf{SPS}}, r)$ algorithm.

Existing instantiations work in the bilinear group setting and allow to sign messages from the space $(\mathbb{G}_i^*)^\ell$, for $\ell > 1$. The partition on the message space in those schemes is induced by the relation $\mathcal{R}_{exp}$: given two messages $M = (M_1, \ldots, M_\ell)$ and $M' = (M'_1, \ldots, M'_\ell)$, we say that M and M' are from the same equivalence class (denoted by $[M]_{\mathcal{R}}$) if there exists a scalar $r \in \mathbb{Z}_p^*$, such that $\forall_{i \in [\ell]} (M_i)^r = M'_i$. The original paper defines two properties of $\mathsf{SPS\text{-}EQ}$ namely unforgeability under chosen-message attacks and class-hiding. Fuchsbauer and Gay [25] recently introduced a weaker version of unforgeability called unforgeability under chosen-open-message attacks, which restricts the adversaries' signing queries to messages where it knows all exponents.

Definition 7 (Signing Oracles). *A* signing oracle *is an $\mathcal{O}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$ (resp. $\mathcal{O}_{\mathsf{op}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$) oracle, which accepts messages $(M_1, \ldots, M_\ell) \in (\mathbb{G}_i^*)^\ell$ (resp. vectors $(e_1, \ldots, e_\ell) \in (\mathbb{Z}_p^*)^\ell$) and returns signature under $\mathsf{sk}_{\mathsf{SPS}}$ on those messages (resp. on messages $(g_1^{e_1}, \ldots, g_1^{e_\ell}) \in (\mathbb{G}_i^*)^\ell$).*

Definition 8 (EUF-CMA (resp. EUF-CoMA)). *A $\mathsf{SPS\text{-}EQ}$ scheme $(\mathsf{SPS.BGGen}, \mathsf{SPS.KGen}, \mathsf{SPS.Sign}, \mathsf{SPS.ChgRep}, \mathsf{SPS.Verify}, \mathsf{SPS.VKey})$ on $(\mathbb{G}_i^*)^\ell$ is called existentially unforgeable under chosen message attacks (resp. adaptive chosen-open-message attacks), if for all PPT algorithms $\mathcal{A}$ having access to an open signing oracle $\mathcal{O}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$ (resp. $\mathcal{O}_{\mathsf{op}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$) the following adversary's advantage (with templates T_1, T_2 defined below) is negligible in the security parameter λ:*

$$\mathbf{Adv}^{\ell, T_1}_{\mathsf{SPS\text{-}EQ}, \mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c} \mathsf{BG} \leftarrow \mathsf{SPS.BGGen}(\lambda); \\ (\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) \xleftarrow{\$} \mathsf{SPS.KGen}(\mathsf{BG}, \ell); \\ (M^*, \sigma^*_{\mathsf{SPS}}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{T_2}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)}(\mathsf{pk}_{\mathsf{SPS}}) \end{array} : \begin{array}{c} \forall M \in Q.\ [M^*]_{\mathcal{R}} \neq [M]_{\mathcal{R}} \quad \wedge \\ \mathsf{SPS.Verify}(\mathsf{pk}_{\mathsf{SPS}}, M^*, \sigma^*_{\mathsf{SPS}}) = 1 \end{array}\right],$$

where Q is the set of messages signed by the signing oracle $\mathcal{O}_{T_2}$ and for $T_1 = \mathsf{euf\text{-}cma}$ we have $T_2 = \mathsf{SPS}$, and for $T_1 = \mathsf{euf\text{-}coma}$ we have $T_2 = \mathsf{op}$.

A stronger notion of class hiding, called perfect adaptation of signatures, was proposed by Fuchsbauer et al. in [25]. Informally, this definition states that signatures received by changing the representative of the class and new signatures for the representative are identically distributed. In our schemes we will only use this stronger notion.

Definition 9 (Perfect Adaption of Signatures). *A* $\mathsf{SPS\text{-}EQ}$ *scheme on* $(\mathbb{G}_i^*)^\ell$ *perfectly adapts signatures if for all* $(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}, M, \sigma, r)$*, where* $\mathsf{SPS.VKey}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 1$*,* $M \in (\mathbb{G}_1^*)^\ell$*,* $r \in \mathbb{Z}_p^*$ *and* $\mathsf{SPS.Verify}(\mathsf{pk}_{\mathsf{SPS}}, M, \sigma) = 1$*, the distribution of*

$$((M)^r, \mathsf{SPS.Sign}(\mathsf{sk}_{\mathsf{SPS}}, M^r)) \text{ and } \mathsf{SPS.ChgRep}(\mathsf{pk}_{\mathsf{SPS}}, M, \sigma, r)$$

are identical.

2.4 Non-Interactive Proof Systems

In this paper we make use of non-interactive proof systems. Although we define the proof system for arbitrarily languages, in our schemes we use the efficient Groth-Sahai (GS) proof system for pairing product equations [30].

Let $\mathcal{R}$ be an efficiently computable binary relation, where for $(x, w) \in \mathcal{R}$ we call x a statement and w a witness. Moreover, we will denote by $L_{\mathcal{R}}$ the language consisting of statements in $\mathcal{R}$, i.e. $L_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\}$.

Definition 10 (Non-Interactive Proof System). *A non-interactive proof system Π consists of the following three algorithms:*

$\Pi.\mathsf{Setup}(1^\lambda)$**:** *on input security parameter 1^λ, this algorithm outputs a common reference string ρ.*
$\Pi.\mathsf{Prove}(\rho, x, w)$**:** *on input common reference string ρ, statement x and witness w, this algorithm outputs a proof π.*
$\Pi.\mathsf{Verify}(\rho, x, \pi)$**:** *on input common reference string ρ, statement x and proof π, this algorithm outputs either* $\mathsf{accept}(1)$ *or* $\mathsf{reject}(0)$*.*

Some proof systems do not need a common reference string. In such a case, we omit the first argument to $\Pi.\mathsf{Prove}$ *and* $\Pi.\mathsf{Verify}$*.*

Definition 11 (Soundness). *A proof system Π is called* sound*, if for all PPT algorithms $\mathcal{A}$ the following probability, denoted by* $\mathsf{Adv}^{\mathsf{sound}}_{\Pi,\mathcal{A}}(\lambda)$*, is negligible in the security parameter 1^λ:*

$$\Pr[\rho \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda); (x, \pi) \xleftarrow{\$} \mathcal{A}(\rho) : \quad \Pi.\mathsf{Verify}(\rho, x, \pi) = \mathsf{accept} \ \wedge \ x \notin L_{\mathcal{R}}].$$

where the probability is taken over the randomness used by $\Pi.\mathsf{Setup}$ *and the adversary $\mathcal{A}$. We say that the proof system is* perfectly sound *if* $\mathsf{Adv}^{\mathsf{sound}}_{\Pi,\mathcal{A}}(\lambda) = 0$*.*

Definition 12 (Perfect Knowledge Extraction). *A proof system Π is called an argument of knowledge for $\mathcal{R}$, if there exists a knowledge extractor* $E = (\Pi.\mathsf{ExtGen}, \Pi.\mathsf{Extract})$ *such that for all algorithms $\mathcal{A}$*

$$\mathsf{Adv}^{\mathrm{e}_1}_{\Pi,\mathcal{A}}(\lambda) := |\Pr[\rho \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda) : \ \mathcal{A}(\rho) = 1] - \Pr[(\rho, \tau) \xleftarrow{\$} \Pi.\mathsf{ExtGen}(1^\lambda) : \ \mathcal{A}(\rho) = 1]|$$

is negligible in λ and

$$\mathsf{Adv}^{\mathrm{e}_2}_{\Pi,\mathcal{A}}(\lambda) := \Pr[(\rho, \tau) \xleftarrow{\$} \Pi.\mathsf{ExtGen}(1^\lambda); (x, \pi) \xleftarrow{\$} \mathcal{A}(\rho, \tau); w \xleftarrow{\$} \Pi.\mathsf{Extract}(\rho, \tau, x, \pi) : \\ \Pi.\mathsf{Verify}(\rho, x, \pi) = \mathsf{accept} \quad \wedge \quad (x, w) \notin \mathcal{R}]$$

is negligible in 1^λ. If $\mathsf{Adv}^{\mathrm{e}_2}_{\Pi,\mathcal{A}}(\lambda) = 0$*, we say that Π is a perfect proof of knowledge.*

Definition 13 (Witness Indistinguishability (WI)). *A proof system Π is* witness indistinguishable*, if for all PPT algorithms $\mathcal{A}$ we have that the advantage $\mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda)$ computed as:*

$$\begin{aligned}&|\Pr[\rho \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda); (x, w_0, w_1) \xleftarrow{\$} \mathcal{A}(1^\lambda, \rho); \pi \xleftarrow{\$} \Pi.\mathsf{Prove}(\rho, x, w_0) : \ \mathcal{A}(\pi) = 1] - \\ &\Pr[\rho \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda); (x, w_0, w_1) \xleftarrow{\$} \mathcal{A}(1^\lambda, \rho); \pi \xleftarrow{\$} \Pi.\mathsf{Prove}(\rho, x, w_1) : \ \mathcal{A}(\pi) = 1]|,\end{aligned}$$

where $(x, w_0), (x, w_1) \in \mathcal{R}$, is at most negligible in λ. We say that the proof system if perfectly witness indistinguishable if $\mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda) = 0$.

Definition 14 (Zero-Knowledge). *A proof system Π is called zero-knowledge, if there exists a PPT simulator $S = (\mathsf{SimGen}, \mathsf{Sim})$ such that for all PPT algorithms $\mathcal{A}$ the following probability, denoted by $\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}}(\lambda)$, is negligible in the security parameter 1^λ:*

$$\begin{aligned}&|\Pr[\rho \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda) : \ \mathcal{A}^{\Pi.\mathsf{Prove}(\rho,\cdot,\cdot)}(\rho) = 1] - \\ &\Pr[(\rho, \tau) \xleftarrow{\$} \mathsf{SimGen}(1^\lambda) : \ \mathcal{A}^{S(\rho,\tau,\cdot,\cdot)}(\rho) = 1]|,\end{aligned}$$

where τ is a trapdoor information, oracle call $S(\rho, \tau, x, w)$ returns the output of $\mathsf{Sim}(\rho, \tau, \mathsf{x})$ for $(x, w) \in \mathcal{R}$ and both oracles output $\perp$ if $(x, w) \notin \mathcal{R}$.

We briefly recall the framework of pairing product equations that is used for the languages of the Groth-Sahai proof system [30]. For constants $A_i \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $t_T \in \mathbb{G}_T$, $\gamma_{ij} \in \mathbb{Z}_p$ which are either publicly known or part of the statement, and witnesses $X_i \in \mathbb{G}_1$, $Y_i \in \mathbb{G}_2$ given as commitments, we give proofs that:

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

The system $(\Pi_{\mathsf{PPE}}.\mathsf{Setup}, \Pi_{\mathsf{PPE}}.\mathsf{Prove}, \Pi_{\mathsf{PPE}}.\mathsf{Verify}, \Pi_{\mathsf{PPE}}.\mathsf{ExtGen}, \Pi_{\mathsf{PPE}}.\mathsf{Extract})$ has several instantiations based on different assumptions. In this paper we only consider the instantiation based on the decisional Diffie-Hellman assumption given by Ghadafi, Smart and Warinschi [28].

2.5 Digital Signatures and Public Key Encryption

In our group signature construction we also make use of standard digital signatures and public key encryption schemes. We use $(\mathsf{DS.KeyGen}, \mathsf{DS.Sign}, \mathsf{DS.Verify})$ to denote the algorithms that make up the scheme DS and $\mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{DS}}(\lambda)$ to denote the adversaries advantage against the existential unforgeability under chosen message attacks of the signature scheme.

A public key encryption scheme PKE consists of three algorithms $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. We use the standard notion of indistinguishability of ciphertexts under chosen message attacks $(\mathsf{IND-CPA})$ as well as the notion of key privacy under chosen message attacks $(\mathsf{IK-CPA})$, which informally requires that it is infeasible for an attacker to determine which key was used to create a given ciphertext even if with access to both encryption keys. A full formal definition of this property can be found in [7]. An example of a scheme which achieves key privacy is the El Gamal encryption scheme [27].

3 Fully Dynamic Group Signatures

We recall the framework of definitions for fully dynamic group signatures established in [15].

Definition 15. *A fully dynamic group signature scheme* GS *is a defined by the following set of efficient algorithms*

$\mathsf{GS.Setup}(1^\lambda)$**:** *On input a security parameter, the setup algorithm outputs public parameters* param *and initializes the user registration table* $\boldsymbol{reg}$.

$\langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param})\rangle$**:** *Given the public parameters* param *the group manager* $\mathcal{M}$ *and tracing manager* $\mathcal{T}$ *jointly execute a key generation protocol.*
- *The private output of the group manager is a secret manager key* msk*, its public output a manager public key* mpk *and the initial group information* info.
- *The private output of the tracing manager is a secret tracing key* tsk *and a tracing manager public key* tpk.

The public outputs together are referred to as the group public key $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$.

$\mathsf{GS.UKGen}(1^\lambda)$**:** *On input the public parameters, the user key generation algorithm outputs a pair of user secret and user public key* $(\boldsymbol{usk}[\mathsf{uid}], \boldsymbol{upk}[\mathsf{uid}])$*, bound to a fresh user id* uid.

$\langle \mathsf{GS.Join}(\mathsf{info}_{\tau_{current}}, \mathsf{gpk}, \mathsf{uid}, \boldsymbol{usk}[\mathsf{uid}]), \mathsf{GS.Issue}(\mathsf{info}_{\tau_{current}}, \mathsf{msk}, \mathsf{uid}, \boldsymbol{upk}[\mathsf{uid}])\rangle$**:** *A user who has executed* $\mathsf{GS.UKGen}$*, obtaining a user id* uid *and key pair* $(\boldsymbol{usk}[\mathsf{uid}], \boldsymbol{upk}[\mathsf{uid}])$ *may, given the group public key and information regarding the current epoch* $\mathsf{info}_{\tau_{current}}$ *engage the group manager in a join-issue procedure to become a member of the group. If successful, the output of the* $\mathsf{GS.Issue}$ *algorithm is user registration information which is stored in* $\boldsymbol{reg}[\mathsf{uid}]$ *the user secret key* $\mathbf{gsk}[\mathsf{uid}]$ *is updated with the output of* $\mathsf{GS.Join}$.

$\mathsf{GS.UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{current}}, \mathcal{S}, \boldsymbol{reg})$**:** *The group manager may advance the current epoch* $\tau_{current}$ *to the next epoch* τ_{new}*, at the same time revoking membership of a subset* $\mathcal{S}$ *of the set of active group members. If any* $\mathsf{uid} \in \mathcal{S}$ *is not assigned to an active member of the group, i.e. was not assigned in a run of the join-issue procedure, the algorithm aborts. The outputs is the new group information* $\mathsf{info}_{\tau_{new}}$ *and a possibly updated registration table* $\boldsymbol{reg}$*. If the group information does not change, the algorithm outputs* $\perp$.

$\mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m)$**:** *Given their group signing key, current group information and the group public key, a user may sign a message, producing a signature* Σ*. If the user-ID* uid *is not assigned to an active group member in the current epoch* $\tau_{current}$*, the algorithm outputs* $\perp$ *instead.*

$\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma)$**:** *If the given signature* Σ *is valid for message* m *in epoch* τ*, verification outputs* accept*, otherwise* reject.

$\mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \boldsymbol{reg}, m, \Sigma)$**:** *Given a signature, message, group information for epoch* τ *and a registration table, the tracing manager may output a pair* (uid, π) *where* $\mathsf{uid} > 0$ *identifies the user-ID of the group member who produced the signature and* π *is a proof of this fact. If tracing is not successful the algorithm will output a pair* $(0, \pi)$ *indicating the failure via the special user-ID 0, which is not assigned to any regular user.*

$\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi_{\mathbf{Trace}}, \boldsymbol{upk}[\mathsf{uid}], m, \Sigma)$**:** *Given a signature for epoch* τ*, the corresponding group information and a tracing output* (uid, π)*, anyone in possession of the group public key can deterministically judge the validity of* π *w.r.t. to the statement, that* Σ *was created using* $\mathbf{gsk}[\mathsf{uid}]$*, in which case the algorithm outputs* accept*, otherwise* reject.

3.1 Security of Fully Dynamic Group Signatures

In the framework of [15], a fully dynamic group signature scheme is secure if it achieves *correctness*, *anonymity*, *non-frameability*, *traceability*, and *tracing soundness.*

Informally, these properties ensure the following guarantees:

Anonymity given a signature, it is infeasible, without a secret trapdoor information, to distinguish which signer created the signatures.

Traceability it is infeasible to produce a signature for which the opening procedure fails. In other words, even a coalition of group members and the opening authority should not be able to produce a signature which would open to an identity not generated in the setup phase or an identity that was not active in the epoch for which the signature was created.

Non-Frameability — any coalition of group members, the issuing authority and the opening authority cannot produce a signature which opens to an identity of an honest user from outside the coalition.

Tracing Soundness — this requirement ensures that even if all parties in the group collude, they cannot produce a valid signature that traces to two different members.

They are formally defined as a set of adversarial experiments relative to a number of oracles, which give the adversary an interface to the scheme:

AddU: To add an honest user to the group.
CrptU: To impersonate (corrupt) a user before joining the group.
SndToM: To communicate in the user role with an honest issuing authority.
SndToU: To communicate as an issuing authority with an honest user.
ReadReg: To read the registration table.
ModifyReg: To modify the registration table.
RevealU: To reveal an honest user's secret keys.
Sign: To obtain a group signature on behalf of an honest user.
Trace: To obtain a tracing proof on a signature.
UpdateGroup: To remove users from the group via the group update procedure.
Chall: To receive a challenge signature in the anonymity game.

Security Experiments We have informally described the security notions of [15] and will now recall their formal definitions.

$\mathsf{Correctness}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda)$

```
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda); \mathcal{H} := \emptyset$
$(\mathsf{msk}, \mathsf{mpk}, \mathsf{info}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{\$} \langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \rangle$
$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$(\mathsf{uid}, m, \tau) \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU}, \mathsf{ReadReg}, \mathsf{UpdateGroup}}(\mathsf{gpk}, \mathsf{info})$
$\textbf{if } \mathsf{uid} \notin \mathcal{H} \text{ or } \mathbf{gsk}[\mathsf{uid}] = \bot \text{ or } \mathsf{info}_\tau = \bot$
$\text{ or } \mathsf{GS.IsActive}(\mathsf{info}_\tau, \boldsymbol{reg}, \mathsf{uid}) = 0$
    $\textbf{then return } 0$
$\Sigma \xleftarrow{\$} \mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m)$
$\textbf{if } \mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = \mathsf{reject}$
    $\textbf{then return } 1$
$(\mathsf{uid}^*, \pi) \xleftarrow{\$} \mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \boldsymbol{reg}, m, \Sigma)$
$\textbf{if } \mathsf{uid} \neq \mathsf{uid}^* \textbf{ then return } 1$
$\textbf{if } \mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi, \boldsymbol{upk}[uid], m, \Sigma) = 0$
    $\textbf{then return } 0 \textbf{ else return } 1$
```

$\mathsf{Anonymity}^{\mathcal{A}}_{b,\mathsf{GS}}(1^\lambda)$

```
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda); \mathcal{H}, \mathcal{C}, \mathcal{B}, \mathcal{Q}, \mathcal{Q}^* := \emptyset$
$(\mathsf{state}, \mathsf{msk}, \mathsf{mpk}, \mathsf{info}) \xleftarrow{\$} \mathcal{A}^{\langle \cdot, \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \rangle}(\mathsf{init} : \mathsf{param})$
$\textbf{if } \bot \leftarrow \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \text{ or } \mathcal{A}\text{'s output invalid}$
    $\textbf{then return } 0$
$(\mathsf{tsk}, \mathsf{tpk}) \leftarrow \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param});\ \mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$d \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU}, \mathsf{CrptU}, \mathsf{SndToU}, \mathsf{RevealU}, \mathsf{Trace}, \mathsf{ModifyReg}, \mathsf{Chall}_b}(\mathsf{play} : \mathsf{state}, \mathsf{gpk})$
$\textbf{return } d$
```

```
$\mathsf{Non-Frame}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda)$
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda); \mathcal{H}, \mathcal{C}, \mathcal{B}, \mathcal{Q} := \emptyset$
$(\mathsf{state}, \mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{\$} \mathcal{A}(\mathsf{init} : \mathsf{param})$
if $\mathsf{msk} = \perp$ or $\mathsf{mpk} = \perp$
    then return $0$
$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$(m, \Sigma, \mathsf{uid}, \pi, \mathsf{info}_\tau) \xleftarrow{\$} \mathcal{A}\left\{\begin{matrix}\mathsf{CrptU}, \quad \mathsf{Sign}, \\ \mathsf{SndToU}, \mathsf{RevealU}, \\ \mathsf{ModifyReg}\end{matrix}\right\}(\mathsf{play} : \mathsf{state}, \mathsf{gpk})$
if $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$
 or $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi, \boldsymbol{upk}[\mathsf{uid}], m, \Sigma) = 0$
    then return $0$
if $\mathsf{uid} \in \mathcal{H} \setminus \mathcal{B}$ and $(\mathsf{uid}, m, \Sigma, \tau) \notin \mathcal{Q}$
    then return $1$ else return $0$
```

```
$\mathsf{Traceability}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda)$
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda); \mathcal{H}, \mathcal{C}, \mathcal{B}, \mathcal{Q} := \emptyset$
$(\mathsf{state}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{\$} \mathcal{A}^{\langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \cdot \rangle}(\mathsf{init} : \mathsf{param})$
if $\perp \leftarrow \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param})$ or $\mathcal{A}$'s output invalid
    then return $0$
$(\mathsf{msk}, \mathsf{mpk}, \mathsf{info}) \leftarrow \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param})$;  $\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$(m, \Sigma, \tau) \xleftarrow{\$} \mathcal{A}\,\}\}\ (\mathsf{play} :, \mathsf{state}, \mathsf{gpk}, \mathsf{info})$
if $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = \mathsf{reject}$
    then return $0$
$(\mathsf{uid}, \pi) \xleftarrow{\$} \mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \boldsymbol{reg}, m, \Sigma)$
if $\mathsf{GS.IsActive}(\mathsf{info}_\tau, \boldsymbol{reg}, \mathsf{uid}) = 0$ or $\mathsf{uid} = 0$
 or $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi, \boldsymbol{upk}[\mathsf{uid}], m, \Sigma) = 0$
    thenreturn $1$else return $0$
```

Oracle Definitions. We formally describe the oracles given to the adversary in the above experiments. Changes introduced for compatibility with our $\mathsf{Join-Privacy}$ and $\mathsf{Leave-Privacy}$ experiments are highlighted.

```
$\mathsf{Trace}(m, \Sigma, \mathsf{info}_\tau)$
if $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = \mathsf{reject}$ or $(m, \Sigma, \tau) \in \mathcal{Q}^*$
    then return $(\perp, \perp)$
return $\mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \boldsymbol{reg}, m, \Sigma)$
```

```
$\mathsf{UpdateGroup}(\mathcal{S})$
if $\mathcal{H}^* \cap \mathcal{S} \neq \emptyset$ then return $\perp$
return $\mathsf{GS.UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{current}}, \mathcal{S}, \boldsymbol{reg})$
```

```
$\mathsf{CrptU}(\mathsf{uid}, \mathsf{pk})$
if $\mathsf{uid} \in \mathcal{H} \cup \mathcal{C}$ $\cup \mathcal{H}^*$ then return $\perp$
$\mathcal{C} := \mathcal{C} \cup \{\mathsf{uid}\}$
$\boldsymbol{upk}[\mathsf{uid}] := \mathsf{pk}$
return accept
```

```
$\mathsf{ReadReg}(\mathsf{uid})$
return $\boldsymbol{reg}[\mathsf{uid}]$
```

```
$\mathsf{ModifyReg}(\mathsf{uid}, val)$
$\boldsymbol{reg}[\mathsf{uid}] := val$
```

```
$\mathsf{RevealU}(\mathsf{uid})$
if $\mathsf{uid} \notin \mathcal{H} \setminus (\mathcal{C} \cup \mathcal{B})$
 or $\mathsf{uid} \in \mathcal{H}^*$
    then return $\perp$
$\mathcal{B} := \mathcal{B} \cup \{\mathsf{uid}\}$
return $(\boldsymbol{usk}[\mathsf{uid}], \mathbf{gsk}[\mathsf{uid}])$
```

```
$\mathsf{Chall}_b(\mathsf{info}_\tau, \mathsf{uid}_0, \mathsf{uid}_1, m)$
if $\{\mathsf{uid}_0, \mathsf{uid}_1\} \cap \mathcal{H} \neq \{\mathsf{uid}_0, \mathsf{uid}_1\}$
 or $\exists b \in \{0, 1\}$ s.t. $\mathbf{gsk}[\mathsf{uid}_b] = \perp$
 or $\mathsf{GS.IsActive}(\mathsf{info}_\tau, \boldsymbol{reg}, \mathsf{uid}_b) = 0$
    then return $\perp$
$\Sigma \xleftarrow{\$} \mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}_b], \mathsf{info}_\tau, m)$
$\mathcal{Q}^* := \mathcal{Q}^* \cup (m, \Sigma, \tau)$
return $\Sigma$
```

```
$\mathsf{AddU}(\mathsf{uid})$
if $\mathsf{uid} \in \mathcal{H} \cup \mathcal{C}$ then return $\perp$
$(\boldsymbol{usk}[\mathsf{uid}], \boldsymbol{upk}[\mathsf{uid}]) \xleftarrow{\$} \mathsf{GS.UKGen}(1^\lambda)$
$\mathcal{H} := \mathcal{H} \cup \{\mathsf{uid}\}$
$\mathbf{gsk}[\mathsf{uid}] := \perp; \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}} := \mathsf{cont}$
$\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}} := (\tau_{current}, \mathsf{gpk}, \mathsf{uid}, \boldsymbol{usk}[\mathsf{uid}])$
$\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}} := (\tau_{current}, \mathsf{msk}, \mathsf{uid}, \boldsymbol{upk}[\mathsf{uid}])$
while $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}} = \mathsf{cont}$ and $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}} = \mathsf{cont}$ do
  $(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}}, M_{\mathsf{GS.Join}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}}) \xleftarrow{\$} \mathsf{GS.Issue}(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}}, M_{\mathsf{GS.Issue}})$
  $(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}}, M_{\mathsf{GS.Issue}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}}) \xleftarrow{\$} \mathsf{GS.Join}(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}}, M_{\mathsf{GS.Join}})$
if $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}} = \mathsf{accept}$ then $\boldsymbol{reg}[\mathsf{uid}] := \mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}}$
if $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}} = \mathsf{accept}$ then $\mathbf{gsk}[\mathsf{uid}] := \mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}}$
return $(\mathsf{info}_{\tau_{current}}, \boldsymbol{upk}[\mathsf{uid}])$
```

```
$\mathsf{SndToU}(\mathsf{uid}, M_{\mathsf{in}})$
if $\mathsf{uid} \in \mathcal{C} \cup \mathcal{B}$ then return $\perp$
if $\mathsf{uid} \notin \mathcal{H}$ then
  $\mathcal{H} := \mathcal{H} \cup \{\mathsf{uid}\}$
  $(\boldsymbol{usk}[\mathsf{uid}], \boldsymbol{upk}[\mathsf{uid}]) \xleftarrow{\$} \mathsf{GS.UKGen}(1^\lambda)$
  $\mathbf{gsk}[\mathsf{uid}] := \perp; M_{\mathsf{in}} := \perp$
if $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}} \neq \mathsf{cont}$ then return $\perp$
if $\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}} = \perp$ then $\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}} := (\tau_{current}, \mathsf{gpk}, \mathsf{uid}, \boldsymbol{usk}[\mathsf{uid}])$
$(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}}, M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}} \xleftarrow{\$} \mathsf{GS.Join}(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}}, M_{\mathsf{in}}))$
if $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}} = \mathsf{accept}$ then $\mathbf{gsk}[\mathsf{uid}] := \mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Join}}$
return $(M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Join}})$
```

```
$\mathsf{SndToM}(\mathsf{uid}, M_{\mathsf{in}})$
if $\mathsf{uid} \notin \mathcal{C}$ or $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}} \neq \mathsf{cont}$
  or $\mathsf{uid} \in \mathcal{H}^*$ then return $\perp$
$\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}} := (\tau_{current}, \mathsf{msk}, \mathsf{uid}, \boldsymbol{upk}[\mathsf{uid}])$
$(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}}, M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}}) \xleftarrow{\$} \mathsf{GS.Issue}(\mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}}, M_{\mathsf{in}})$
if $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}} = \mathsf{accept}$ then $\boldsymbol{reg}[\mathsf{uid}] := \mathsf{state}^{\mathsf{uid}}_{\mathsf{GS.Issue}}$
return $(M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{GS.Issue}})$
```

```
$\mathsf{Sign}(\mathsf{uid}, m, \tau)$
if $\mathsf{uid} \notin \mathcal{H} \setminus \mathcal{H}^*$ or $\mathbf{gsk}[\mathsf{uid}] = \perp$
 or $\mathsf{info}_\tau = \perp$ or $\mathsf{GS.IsActive}(\mathsf{info}_\tau, \boldsymbol{reg}, \mathsf{uid}) = 0$
  then return $\perp$
$\Sigma \xleftarrow{\$} \mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m)$
$\mathcal{Q} := \mathcal{Q} \cup \{(\mathsf{uid}, m, \Sigma, \tau)\}$
return $\Sigma$
```

```
$\mathsf{Trace-Sound}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda)$
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda); \mathcal{C} := \emptyset$
$(\mathsf{state}, \mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{\$} \mathcal{A}(\mathsf{init} : \mathsf{param})$
if $\mathsf{msk} = \perp$ or $\mathsf{mpk} = \perp$
  then return $0$
$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$(m, \Sigma, \{\mathsf{uid}_i, \pi_i\}_{i=1}^2, \mathsf{info}_\tau) \xleftarrow{\$} \mathcal{A}^{\mathsf{CrptU},\mathsf{ModifyReg}}(\mathsf{play} : \mathsf{state}, \mathsf{gpk})$
if $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$
  then return $0$
if $\boldsymbol{upk}[\mathsf{uid}_1] = \boldsymbol{upk}[\mathsf{uid}_2]$ or $\exists i \in \{1,2\}$ s.t. $\boldsymbol{upk}[\mathsf{uid}_i] = \perp$
 or $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}_i, \mathsf{info}_\tau, \pi_i, \boldsymbol{upk}[\mathsf{uid}_i], m, \Sigma) = 0$
  thenreturn $0$ else return $1$
```

Functional Tracing Soundness. The original definition of tracing soundness (as *opening soundness* in [40]) requires that the attacker produce two valid openings to different user IDs, as opposed to different public keys. This is motivated by the fact that in the attacker model the whole group may be corrupted and thus the attacker may give two different users the same key information. However, it implies that the GS.Vf and GS.Judge algorithms of the schemes in [40] have to check that the group is well-formed, i.e. that every user has a unique key in the registration table. These checks are costly and unavoidable. In addition it necessitates that the registration table $\boldsymbol{reg}$ must be public, since otherwise signature verification and opening judgment cannot perform these checks.

Therefore, we propose a relaxation of this property, which we call *functional tracing soundness* and which means that even in a fully corrupted group it should not be possible to create two valid openings for the same signature which indict two different user public keys. We motivate this by the observation that commonly public keys *are* the public identities of users and the link between some informal identity and the identity in the context of the signature scheme is established through the functional property that the user whose identity belongs to a given public key can create signatures which verify under that key. As discussed above, non-frameability only makes sense if there is an entity outside the signature scheme which certifies that certain public keys belong to informal identities. This entity would not assign the same public key to several different public identities. The changes to the Tracing Soundness experiment which implement this change are highlighted.

Furthermore, we observe that the fully dynamic group signature scheme based on accountable ring signatures presented in [15] implicitly uses this definition already (although it is presented differently, presumably due to a typographic mistake), since its proof of tracing soundness relies on the tracing soundness of the underlying accountable ring signature scheme. The property for accountable ring signature schemes requires that the verification keys provided in the two openings be different. Were this not a requirement assumed in the proof in [15], the given reduction wouldn't give valid outputs against tracing soundness of the accountable ring signature scheme and the proof would be invalid.

We stress that the construction of fully dynamic group signatures presented later in this work can be made to achieve the identity focused version of opening soundness, albeit at the cost of the above mentioned group integrity checks and any kind of group membership privacy.

3.2 Leave-Join Privacy for Fully Dynamic Group Signatures

Formal models of dynamic group signatures thus far implicitly assumed that the public is aware who is a member of the group. Usually, a registration table is published, such that the entries are bound to public keys of the members. This is in line with one of the main application of group signatures: authenticating messages with the authority of a known group, certifying that an indeterminate *someone* within the group has seen the signed message and taken responsibility on behalf of the group.

In their seminal work Chaum and van Heyst [21], however, did not specify this as an essential requirement. In fact, they point out that group signatures can be used for access control, where knowing members of the group is an obvious privacy leak that could for instance lead to targeted DoS attacks on the group. Therefore it seems natural that in some applications we want to hide the identities of active group members.

To address this issue we discuss for the first time *membership privacy* for group signatures. Informally, we will say that a group signature scheme has membership privacy if it protect the identity of users that join or leave the system. This means that we consider a scenario in which some kind of public identifier about users is known independently of the scheme (e.g. public key) but it is unknown to a third party who is part of the group. Moreover, we assume that some users can be corrupted or can collude to infer information about the membership status of other users.

To formally define this notion, we propose a pair of security experiments which are expressed in the fully dynamic framework put forth by [15]. However, one can easily specify similar experiments for the partially dynamic models [9, 34, 35]. The first one describes *join privacy*, since it considers the case that two non-members are known in one epoch and in the next epoch one of them joins the system and the task is to distinguish who joined the group. The second experiment describes *leave privacy* and models the case that there are two known members in one epoch and in the next epoch one of them leaves the group. Note that this assumes that the adversary knows out of band that the two users had previously joined the group. In both cases we allow an adversary to corrupt members of the group but we consider both authorities to be honest: The issuing authority always knows who is part of the group and the tracing authority can open all signatures to extract the identities of members. In particular, this implies that the registration table ***reg*** may not be public because one could easily infer current members from it. Fortunately, this seems a fairly natural assumption. This registration table is not necessary in any of the user centric algorithms and it is easier to keep it local to the authorities than publishing it online. An exception is the scheme [40] mentioned above, where the registration table is part of the verification algorithm to ensure that tracing soundness hold with respect to public user identities rather than in the functional sense we describe.

A different question is whether additionally to the identities of users, we can hide the size of the group. Unfortunately, since the fully dynamic model in [15] allows joining and leaving the group, all efficient constructions fail to hide the size of the group. Whitelisting immediately leaks the size of the group and can only be alleviated using dummy users, which incurs large overhead and fixes a constant upper bound on the group size. This is even the case for cryptographic accumulators, where it is required by members to update their witness with every epoch. Thus, some kind of information that is linear is the number of active/inactive members must be published together with the accumulator.

We formally define join and leave privacy in terms of the two experiments shown in Figure 1. Note, that we introduce a new set of challenge users $\mathcal{H}^*$. It is used simultaneously to ensure compatibility of the oracles defined in [15] (our changes are highlighted) and to manage the challenge oracle $\mathsf{PrivChall}$ without trivially revealing information about the challenge.

The oracle $\mathsf{PrivChall}$ allows the adversary to obtain signatures created by the user which *has joined* or *has not left* in the challenge epoch, respectively.

```
$\mathsf{PrivChall}_{b,\mathsf{uid}_0,\mathsf{uid}_1,\tau^*}(m,\tau)$
if $\tau < \tau^*$ then return $\bot$
if $\mathsf{dec}_{\mathsf{inv}} = \mathbf{true}$
    $\Sigma \xleftarrow{\$} \mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}_{(1-b)}], \mathsf{info}_\tau, m)$
else
    $\Sigma \xleftarrow{\$} \mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}_b], \mathsf{info}_\tau, m)$
$\mathcal{Q}^* := \mathcal{Q}^* \cup (m, \Sigma, \tau)$
return $\Sigma$
```

Definition 16. *For a group signature scheme* GS *and a two stage adversary* $\mathcal{A}$*, we define the adversary's advantage in the* $\mathsf{Join}-\mathsf{Privacy}$ *experiment as*

$$\mathsf{Adv}^{\mathsf{join-privacy}}_{\mathcal{A},\mathsf{GS}}(\lambda) := \Pr\left[\mathsf{Join}-\mathsf{Privacy}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda) \Rightarrow 1\right].$$

n A group signature scheme GS *has join privacy if for all ppt adversaries* $\mathcal{A}$*, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{join-privacy}}_{\mathcal{A},\mathsf{GS}}(\lambda) \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Definition 17. *For a group signature scheme* GS *and a two stage adversary* $\mathcal{A}$*, we define the adversary's advantage in the* $\mathsf{Leave}-\mathsf{Privacy}$ *experiment as*

$$\mathsf{Adv}^{\mathsf{leave-privacy}}_{\mathcal{A},\mathsf{GS}}(\lambda) := \Pr\left[\mathsf{Leave}-\mathsf{Privacy}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda) \Rightarrow 1\right].$$

```
$\mathsf{Join} - \mathsf{Privacy}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda)$
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda)$
$(\mathsf{msk}, \mathsf{mpk}, \mathsf{info}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{\$} \langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \rangle$
$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$(\mathsf{state}, \mathsf{uid}_0, \mathsf{uid}_1) \xleftarrow{\$} \mathcal{A}_0 \left\{ \begin{smallmatrix} \mathsf{AddU}, \mathsf{RevealU}, \mathsf{CrptU}, \mathsf{SndToM}, \\ \mathsf{Sign}, \mathsf{Trace}, \mathsf{UpdateGroup} \end{smallmatrix} \right\} (\mathsf{gpk}, \mathsf{info})$
$\textbf{if } \{\mathsf{uid}_0, \mathsf{uid}_1\} \cap \mathcal{C} \neq \emptyset \textbf{ then return } 0$
$b \xleftarrow{\$} \{0,1\};\ (\mathsf{info}^*, \boldsymbol{upk}[\mathsf{uid}_b]) \xleftarrow{\$} \mathsf{AddU}(\mathsf{uid}_b);$
$(\boldsymbol{usk}[\mathsf{uid}_{1-b}], \boldsymbol{upk}[\mathsf{uid}_{1-b}]) \xleftarrow{\$} \mathsf{GS.UKGen}(1^\lambda)$
$\tau^* := \tau_{current};\ \mathcal{H}^* := \{\mathsf{uid}_0, \mathsf{uid}_1\}$
$d \xleftarrow{\$} \mathcal{A}_1 \left\{ \begin{smallmatrix} \mathsf{AddU}, \mathsf{RevealU}, \mathsf{Sign}, \\ \mathsf{CrptU}, \mathsf{SndToM}, \mathsf{UpdateGroup}, \\ \mathsf{Trace}, \mathsf{PrivChall}_{b, \mathsf{uid}_0, \mathsf{uid}_1, \tau^*} \end{smallmatrix} \right\} (\mathsf{state}, \mathsf{info}^*, \boldsymbol{upk}[\mathsf{uid}_0], \boldsymbol{upk}[\mathsf{uid}_1])$
$\textbf{return } b = d$
```

```
$\mathsf{Leave} - \mathsf{Privacy}^{\mathcal{A}}_{\mathsf{GS}}(1^\lambda)$
$\mathsf{param} \xleftarrow{\$} \mathsf{GS.Setup}(1^\lambda)$
$(\mathsf{msk}, \mathsf{mpk}, \mathsf{info}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{\$} \langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \rangle$
$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$
$(\mathsf{state}, \mathsf{uid}_0, \mathsf{uid}_1) \xleftarrow{\$} \mathcal{A}_0 \left\{ \begin{smallmatrix} \mathsf{AddU}, \mathsf{RevealU}, \\ \mathsf{UpdateGroup}, \mathsf{Sign}, \mathsf{Trace} \end{smallmatrix} \right\} (\mathsf{gpk}, \mathsf{info})$
$\textbf{if } \{\mathsf{uid}_0, \mathsf{uid}_1\} \cap \mathcal{H} \setminus (\mathcal{C} \cup \mathcal{B}) \neq \{\mathsf{uid}_0, \mathsf{uid}_1\} \textbf{ then return } 0$
$b \xleftarrow{\$} \{0,1\}; \mathcal{H}^* := \{\mathsf{uid}_0, \mathsf{uid}_1\}; \mathsf{dec}_{\mathsf{inv}} := \textbf{true};\ \tau^* := \tau_{current}$
$\mathsf{info}^* \xleftarrow{\$} \mathsf{GS.UpdateGroup}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau^*}, \mathsf{uid}_b, \boldsymbol{reg})$
$d \xleftarrow{\$} \mathcal{A}_1 \left\{ \begin{smallmatrix} \mathsf{AddU}, \mathsf{RevealU}, \mathsf{Sign}, \mathsf{Trace}, \\ \mathsf{UpdateGroup}, \mathsf{PrivChall}_{b, \mathsf{uid}_0, \mathsf{uid}_1, \tau^*} \end{smallmatrix} \right\} (\mathsf{state}, \mathsf{info}^*)$
$\textbf{return } b = d$
```

Fig. 1. Join- and Leave-Privacy experiments.

A group signature scheme GS *has join privacy if for all ppt adversaries $\mathcal{A}$, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{leave-privacy}}_{\mathcal{A},\mathsf{GS}}(\lambda) \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Remark 1. Note that leave privacy as stated above only seems to ensure privacy, when a single user leaves the group, however, the GS.UpdateGroup algorithm allows simultaneous membership revocation for a whole set of users $\mathcal{S}$. However, a simple hybrid argument should suffice to extend the join privacy property from one revocation to many revocations.

4 Signatures with Flexible Public Key

In this section we first recall the notion of signatures with flexible public key (SFPK) introduced by Backes et al. [6]. We then extend this primitive by a natural notion called *canonical representatives* (or *canonical form*) and show a weaker anonymity definition that allows us to construct a more efficient scheme than ones presented in [6]. We use this scheme to efficiently instantiate our group signature construction presented in Section 5

The basic idea behind signatures with flexible public key is to divide the key space into equivalence classes induced by a relation $\mathcal{R}$. A signer can efficiently generate $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{SFPK.KeyGen}(1^\lambda)$ and change her key pair to a different representative of the same class. This can be done using a random coin $r \xleftarrow{\$} \mathsf{coin}$ and two algorithms: $\mathsf{pk}' \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ for the public key and $\mathsf{sk}' \xleftarrow{\$} \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$ for the secret key. The randomized secret key can be used to sign a message $\mathsf{Sig} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk}, m)$, such that the signature can be verified by running $\mathsf{SFPK.Verify}(\mathsf{pk}', m, \mathsf{Sig})$. The main feature is class-hiding, which ensures that without a trapdoor it is hard to distinguish if two public keys are related, i.e. in the same equivalence class. However, given the trapdoor one can run the $\mathsf{SFPK.ChkRep}(\delta, \mathsf{pk}')$ algorithm to check if pk' is in relation to the public key for which the trapdoor was generated using trapdoor key generation algorithm $(\mathsf{sk}, \mathsf{pk}, \delta) \xleftarrow{\$} \mathsf{SFPK.TKeyGen}(1^\lambda)$. We will define this scheme in the multi-user setting, i.e. with a setup algorithm $\mathsf{SFPK.CRSGen}(1^\lambda)$ that outputs a common reference string ρ. We only consider a scenario in which this setup has to be executed by a trusted party in order for the scheme to be unforgeable. Note that this means that the secrets used to generate the ρ can be used to forge signatures. This kind of trapdoor δ_ρ was not specified in [6], but we will use it in the security proof of our group signature scheme. In other words, this means there is an alternative signing algorithm $\mathsf{SFPK.Sign}(\delta_\rho, \mathsf{pk}, m)$, which outputs valid signatures for the relation class $[\mathsf{pk}]_{\mathcal{R}}$, without knowledge of the corresponding secret key sk.

Definition 18 (Signature with Flexible Public Key). *A signature scheme with flexible public key* SFPK *is a set of PPT algorithms such that:*

$\mathsf{SFPK.CRSGen}(1^\lambda)$ *takes as input a security parameters 1^λ and outputs a trapdoor δ_ρ and a common reference string ρ, which is an implicit input for all the algorithms.*

$\mathsf{SFPK.KeyGen}(1^\lambda, \omega)$**:** *takes as input a security parameter 1^λ, random coins $\omega \in \mathsf{coin}$ and outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of secret and public keys,*

$\mathsf{SFPK.TKeyGen}(1^\lambda, \omega)$**:** *a trapdoor key generation that takes as input a security parameter 1^λ, random coins $\omega \in \mathsf{coin}$ and outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of secret and public keys, and a trapdoor δ.*

$\mathsf{SFPK.Sign}(\mathsf{sk}, m)$**:** *takes as input a message $m \in \{0,1\}^*$ and a signing key sk, and outputs a signature Sig,*

$\mathsf{SFPK.ChkRep}(\delta, \mathsf{pk})$**:** *takes as input a trapdoor δ for some equivalence class $[\mathsf{pk}']_{\mathcal{R}}$ and public key pk, the algorithm outputs 1 if $\mathsf{pk} \in [\mathsf{pk}']_{\mathcal{R}}$ and 0 otherwise,*

$\mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$**:** *on input a representative public key pk of an equivalence class $[\mathsf{pk}]_{\mathcal{R}}$ and random coins r, this algorithm returns a different representative pk', where $\mathsf{pk}' \in [\mathsf{pk}]_{\mathcal{R}}$.*

$\mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$**:** *on input a secret key sk and random coins r, this algorithm returns an updated secret key sk'.*

$\mathsf{SFPK.Verify}(\mathsf{pk}, m, \mathsf{Sig})$**:** *takes as input a message m, signature Sig, public verification key pk and outputs 1 if the signature is valid and 0 otherwise.*

Definition 19 (Correctness). *We say that a* SFPK *scheme is* correct *if for all $1^\lambda \in \mathbb{N}$, all random coins $\omega, r \in \mathsf{coin}$ the following conditions hold:*

1. *The output distribution of* SFPK.KeyGen *and* SFPK.TKeyGen *is identical.*
2. *For all key pairs $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{SFPK.KeyGen}(1^\lambda, \omega)$ and all messages m we have $\mathsf{SFPK.Verify}(\mathsf{pk}, m, \mathsf{SFPK.Sign}(\mathsf{sk}, m)) = 1$ and $\mathsf{SFPK.Verify}(\mathsf{pk}', m, \mathsf{SFPK.Sign}(\mathsf{sk}', m)) = 1$, where $\mathsf{SFPK.ChgPK}(\mathsf{pk}, r) = \mathsf{pk}'$ and $\mathsf{SFPK.ChgSK}(\mathsf{sk}, r) = \mathsf{sk}'$.*
3. *For all $(\mathsf{sk}, \mathsf{pk}, \delta) \xleftarrow{\$} \mathsf{SFPK.TKeyGen}(1^\lambda, \omega)$ and all pk' we have $\mathsf{SFPK.ChkRep}(\delta, \mathsf{pk}') = 1$ if and only if $\mathsf{pk}' \in [\mathsf{pk}]_{\mathcal{R}}$.*

Definition 20 (Class-hiding). *For scheme* SFPK *with relation $\mathcal{R}$ and adversary $\mathcal{A}$ we define the following experiment:*

```
$\mathsf{C\text{-}H}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}}(\lambda)$
$\omega_0, \omega_1 \xleftarrow{\$} \mathsf{coin}$
$(\mathsf{sk}_i, \mathsf{pk}_i) \xleftarrow{\$} \mathsf{SFPK.KeyGen}(1^\lambda, \omega_i)$ for $i \in \{0,1\}$
$m \xleftarrow{\$} \mathcal{A}(\omega_0, \omega_1); b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \mathsf{coin}$
$\mathsf{sk}' \xleftarrow{\$} \mathsf{SFPK.ChgSK}(\mathsf{sk}_b, r); \mathsf{pk}' \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk}_b, r)$
$\mathsf{Sig} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk}', m)$
$\hat{b} \xleftarrow{\$} \mathcal{A}(\omega_0, \omega_1, m, \mathsf{Sig}, \mathsf{pk}')$
return $b = \hat{b}$
```

An SFPK *is* class-hiding *if for all PPT adversaries $\mathcal{A}$, its advantage in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \left|\Pr\left[\mathsf{C\text{-}H}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}}(\lambda) = 1\right] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)\,.$$

Definition 21 (Strong Existential Unforgeability under Flexible Public Key). *For scheme* SFPK *with relation $\mathcal{R}$ and adversary $\mathcal{A}$ we define the following experiment:*

```
$\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}}(\lambda)$
$\omega \xleftarrow{\$} \mathsf{coin}$
$(\mathsf{sk},\mathsf{pk},\delta) \xleftarrow{\$} \mathsf{SFPK.TKeyGen}(1^\lambda,\omega); Q := \emptyset$
$(\mathsf{pk}',m^*,\mathsf{Sig}^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^1(\mathsf{sk},\cdot),\mathcal{O}^2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk},\delta)$
return $(m^*,\mathsf{Sig}^*) \notin Q \wedge$
          $\mathsf{SFPK.ChkRep}(\delta,\mathsf{pk}') = 1 \wedge$
          $\mathsf{SFPK.Verify}(\mathsf{pk}',m^*,\mathsf{Sig}^*) = 1$
```

```
$\mathcal{O}^1(\mathsf{sk},m)$
$\mathsf{Sig} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk},m)$
$Q := Q \cup \{(m,\mathsf{Sig})\}$
return $\mathsf{Sig}$
```

```
$\mathcal{O}^2(\mathsf{sk},m,r)$
$\mathsf{sk}' \xleftarrow{\$} \mathsf{SFPK.ChgSK}(\mathsf{sk},r)$
$\mathsf{Sig} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk}',m)$
$Q := Q \cup \{(m,\mathsf{Sig})\}$
return $\mathsf{Sig}$
```

A SFPK *is* existentially unforgeable with flexible public key under chosen message attacks *if for all PPT adversaries $\mathcal{A}$ the advantage in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathsf{seuf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \Pr\left[\mathsf{EUF-CMA}^{\mathcal{A}}_{\mathsf{SFPK}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,.$$

Backes et al. introduced also a optional property called key recovery, which allows a user to compute a secret key that corresponds to a different representative of the equivalence of the users public key. A straightforward application of this property is that the $\mathsf{SFPK.ChgPK}$ algorithm can be executed by a third party on the users secret key and the user can compute the corresponding secret key without the knowledge of the randomization r. More formally.

Definition 22 (Key Recovery Property). *A* SFPK *has* recoverable signing keys *if there exists an efficient algorithm* $\mathsf{SFPK.Recover}$ *such that for all security parameters $1^\lambda \in \mathbb{N}$, random coins ω, r and all* $(\mathsf{sk},\mathsf{pk},\delta) \xleftarrow{\$} \mathsf{SFPK.TKeyGen}(1^\lambda,\omega)$ *and* $\mathsf{pk}' \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk},r)$ *we have* $\mathsf{SFPK.ChgSK}(\mathsf{sk},r) = \mathsf{SFPK.Recover}(\mathsf{sk},\delta,\mathsf{pk}')$.

New Definitions. The class-hiding definition introduced by Backes et al. implements the strongest corruption model, i.e. the adversary is given the random coins used by the signer to generate her public key and is still not able to distinguish whether a randomized public key is from the same equivalence class as the signer's public key. As shown in [6], this strong definition allows to construct ring signatures but requires a technique called invertible sampling introduced by Damgård and Nielsen [22]. In many cases we do not require such a strong corruption model and assume that those random coins are destroyed after the user/signer generates her secret key. This slightly weaker model is for example used in the security of group signatures.

On the other hand, the original definition only allows the adversary to see the randomized public key pk' after it specifies the message m in the first phase. This does not model scenarios, where the signer uses the same representative for multiple messages. An easy approach to allow the adversary for adaptive signature queries is to give it a signing oracle. We address these issues by introducing *adaptive class-hiding with key corruption.*

Definition 23 (Adaptive Class-hiding with Key Corruption). *For scheme* SFPK *with relation* $\mathcal{R}$ *and adversary* $\mathcal{A}$ *we define the following experiment:*

```
$\mathsf{adaptC\text{-}H}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}}(\lambda)$
$\omega_0, \omega_1 \xleftarrow{\$} \mathsf{coin}$
$(\mathsf{sk}_i, \mathsf{pk}_i) \xleftarrow{\$} \mathsf{SFPK.KeyGen}(1^\lambda, \omega_i)$ for $i \in \{0,1\}$
$b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \mathsf{coin}$
$\mathsf{sk}' \xleftarrow{\$} \mathsf{SFPK.ChgSK}(\mathsf{sk}_b, r); \mathsf{pk}' \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk}_b, r)$
$\hat{b} \xleftarrow{\$} \mathcal{A}^{\mathsf{SFPK.Sign}(\mathsf{sk}',\cdot)}((\mathsf{sk}_0, \mathsf{pk}_0), (\mathsf{sk}_1, \mathsf{pk}_1), \mathsf{pk}')$
return $b = \hat{b}$
```

A SFPK *is* adaptively class-hiding with key corruption *if for all PPT adversaries* $\mathcal{A}$*, its advantage in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \left| \Pr\left[\mathsf{adaptC\text{-}H}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}}(\lambda) = 1 \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

In some applications it might be required that every equivalence class has a unique representative that can act as a description of the class. For example let the public keys be of the form $\mathsf{pk} = (g_1^a, g_1^x)$ and let the classes be induced by the relation $\mathcal{R}_{exp}$. Since g_1 is publicly known, we can define $\mathsf{pk}' = (g_1, g_1^{x \cdot a^{-1}})$ as this unique representative. We will call pk' a *canonical representative.* The canonical representative can be scheme specific but using a publicly known element (in this case g_1) to fix the first element of the public key allows to fix the structure of the representatives over all equivalence classes. Later, we will assume that if a scheme has canonical representatives, there is an efficient algorithm $\mathsf{IsCanonical}$ which on input a public key will return 1 if and only if the public key is canonical.

We further define the following useful property between signatures on equivalence classes and signatures with flexible public keys.

Definition 24 ($\mathsf{SPS\text{-}EQ}$/SFPK Compatibility). *An* $\mathsf{SPS\text{-}EQ}$ *scheme and an* SFPK *scheme are* compatible *if the message space of the former is the same as the key space of the latter and they share the same equivalence relation.*

4.1 Our Signatures with Flexible Public Key

In this section we propose our signatures with flexible public key. We propose a scheme that is closely related to the ones proposed in [6]. However, security relies on the bilinear decisional Diffie-Hellman assumption instead of the decisional linear assumption. This allows us to decrease the size of the public key by 1 group element in $\mathbb{G}_1$, i.e. from 3 to 2. Unlike the schemes in [6], this scheme only has adaptive class-hiding with key corruption but this is still sufficient for group signatures construction.

In our scheme we assume that both the $\mathsf{SFPK.KeyGen}$ and $\mathsf{SFPK.TKeyGen}$ output a public key that is the canonical representative of its equivalence class. Further we assume that every user has access to a collision resistant hash function H, which we express by including it in the output of $\mathsf{SFPK.CRSGen}$. The $\mathsf{SFPK.ChgPK}$ and $\mathsf{SFPK.ChgSK}$ algorithms work by drawing uniformly at random an exponent $r \in \mathbb{Z}_p$ and raising every component of the public key, or respectively the secret key to the power of r. More details can be found in Scheme 2.

```
$\mathsf{SFPK.CRSGen}(1^\lambda)$
    $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(\lambda);\ y, z \xleftarrow{\$} \mathbb{Z}_p^*$
    $K_{\mathsf{PHF}} \xleftarrow{\$} \mathsf{PHF.Gen}(1^\lambda)$
    $Y_1 \leftarrow g_1^y;\ Y_2 \leftarrow g_2^y;\ \hat{g} \leftarrow g_1^z$
    return $(\rho := (\mathsf{BG}, Y_1, Y_2, K_{\mathsf{PHF}}, \hat{g}, \mathsf{H}),$
            $\delta_\rho := (y, z))$
```

```
$\mathsf{SFPK.ChkRep}(\delta_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{SFPK}})$
    $\mathsf{pk}_{\mathsf{SFPK}} = (\mathsf{pk}_1, \mathsf{pk}_2);\ \tau_{\mathsf{SFPK}} = (\tau)$
    if $e(\mathsf{pk}_1, \tau) = e(\mathsf{pk}_2, g_2)$
        return 1 else 0
```

```
$\mathsf{SFPK.KeyGen}(1^\lambda)$
    $x \xleftarrow{\$} \mathbb{Z}_p^*$
    return $(\mathsf{pk}_{\mathsf{SFPK}} := (g_1, g_1^x),$
            $\mathsf{sk}_{\mathsf{SFPK}} := (Y_1^x, \mathsf{pk}_{\mathsf{SFPK}}))$
```

```
$\mathsf{SFPK.TKeyGen}(1^\lambda)$
    $x \xleftarrow{\$} \mathbb{Z}_p^*$
    return $(\mathsf{pk}_{\mathsf{SFPK}} := (g_1, g_1^x),$
            $\mathsf{sk}_{\mathsf{SFPK}} := (Y_1^x, \mathsf{pk}_{\mathsf{SFPK}}),$
            $\tau_{\mathsf{SFPK}} := (g_2^x))$
```

```
$\mathsf{SFPK.Sign}(\mathsf{sk}, m)$
    $\mathsf{sk}_{\mathsf{SFPK}} = (Z, \mathsf{pk}_{\mathsf{SFPK}});$
    $r, s \xleftarrow{\$} \mathbb{Z}_p^*;\ \mathsf{Sig}^2_{\mathsf{SFPK}} \leftarrow g_1^r;\ \mathsf{Sig}^3_{\mathsf{SFPK}} \leftarrow g_2^r$
    $h \leftarrow \mathsf{H}(m||\mathsf{Sig}^2_{\mathsf{SFPK}}||\mathsf{Sig}^3_{\mathsf{SFPK}}||\mathsf{pk}_{\mathsf{SFPK}})$
    $M \leftarrow g_1^h \cdot \hat{g}^s$
    return $(Z \cdot (\mathsf{PHF.Eval}(K_{\mathsf{PHF}}, M))^r, g_1^r, g_2^r, s)$
```

```
$\mathsf{SFPK.Verify}(\mathsf{pk}_{\mathsf{SFPK}}, m, \mathsf{Sig}_{\mathsf{SFPK}})$
    $\mathsf{pk}_{\mathsf{SFPK}} = (\cdot, X);\ \mathsf{Sig}_{\mathsf{SFPK}} = (\mathsf{Sig}^1_{\mathsf{SFPK}}, \mathsf{Sig}^2_{\mathsf{SFPK}}, \mathsf{Sig}^3_{\mathsf{SFPK}}, s)$
    $h \leftarrow \mathsf{H}(m||\mathsf{Sig}^2_{\mathsf{SFPK}}||\mathsf{Sig}^3_{\mathsf{SFPK}}||\mathsf{pk}_{\mathsf{SFPK}});\ M \leftarrow g_1^h \cdot \hat{g}^s$
    if $e(\mathsf{Sig}^2_{\mathsf{SFPK}}, g_2) = e(g_1, \mathsf{Sig}^3_{\mathsf{SFPK}})$ and
        $e(\mathsf{Sig}^1_{\mathsf{SFPK}}, g_2) = e(X, Y_2) \cdot e(\mathsf{PHF.Eval}(K_{\mathsf{PHF}}, M), \mathsf{Sig}^3_{\mathsf{SFPK}})$
        return 1 else 0
```

Fig. 2. Our Flexible Signatures.

Theorem 1 (Unforgeability). *Scheme 2 is strongly existential unforgeable under flexible public key in the common reference string model, assuming the bilinear decisional Diffie-Hellman assumption holds (which implies that the discrete logarithm assumption holds) and that* PHF *is a* $(1, \mathsf{poly}(\lambda))$*-programmable hash function and* H *is collision-resistant.*

Proof. Let $(\mathsf{Sig}^*_{\mathsf{SFPK}}, m^*, \mathsf{pk}^*_{\mathsf{SFPK}})$ be the forgery returned by an adversary $\mathcal{A}$, where $\mathsf{Sig}^*_{\mathsf{SFPK}} = (\mathsf{Sig}^*_1, \mathsf{Sig}^*_2, \mathsf{Sig}^*_3, s^*)$. We distinguish three types of strategies and will use $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ to denote the corresponding adversaries:

Type 1 We call the adversary a type 1 adversary if there exists a public key $\mathsf{pk}_{\mathsf{SFPK}}$ and signature $\mathsf{Sig}_{\mathsf{SFPK}} = (\mathsf{Sig}_1, \mathsf{Sig}_2, \mathsf{Sig}_3, s)$ on message m generated by oracle $\mathcal{O}^1$ or $\mathcal{O}^2$, where $\mathsf{H}(m^*||\mathsf{Sig}^*_2||\mathsf{Sig}^*_3||\mathsf{pk}^*_{\mathsf{SFPK}}) = \mathsf{H}(m||\mathsf{Sig}_2||\mathsf{Sig}_3||\mathsf{pk}_{\mathsf{SFPK}})$.

Type 2 We call the adversary a type 2 adversary if there exists a public key $\mathsf{pk}_{\mathsf{SFPK}}$ and signature $\mathsf{Sig}_{\mathsf{SFPK}} = (\mathsf{Sig}_1, \mathsf{Sig}_2, \mathsf{Sig}_3, s)$ on message m generated by oracle $\mathcal{O}^1$ or $\mathcal{O}^2$, where

$$e^* = \mathsf{H}(m^*||\mathsf{Sig}^*_2||\mathsf{Sig}^*_3||\mathsf{pk}^*_{\mathsf{SFPK}}) \neq \mathsf{H}(m||\mathsf{Sig}_2||\mathsf{Sig}_3||\mathsf{pk}_{\mathsf{SFPK}}) = e$$

but $M^* = g_1^{e^*} \cdot \hat{g}^{s^*} = g_1^e \cdot \hat{g}^s = M$.

Type 3 We call the adversary a type 3 adversary in all other cases. In particular, we ensure that M^* is distinct from all M's used in the oracles $\mathcal{O}^1$ and $\mathcal{O}^2$.

Type 1 It is easy to see that the adversary broke the collision-resistance of function H and we can build a reduction $\mathcal{R}$ that uses $\mathcal{A}_1$ to break collision-resistance of function H by simulating the system and returning

$$(m^*||\mathsf{Sig}^*_2||\mathsf{Sig}^*_3||\mathsf{pk}^*_{\mathsf{SFPK}}, m||\mathsf{Sig}_2||\mathsf{Sig}_3||\mathsf{pk}_{\mathsf{SFPK}})$$

as a valid collision.

Type 2 In this case we show that a type 2 can be used to break the discrete logarithm assumption. We can apply the same reasoning as for Pedersen commitments, i.e. the reduction can set $\hat{g}$ as the element for which we want to compute the discrete logarithm in respect to g_1. The reduction can then simply simulate the whole system for $\mathcal{A}_2$ and output $(e - e^*)/(s^* - s)$.

Type 3 Let $(\mathsf{BG}, g_1^a, g_2^a, g_1^b, g_2^b, g_1^c, g_2^c, g_1^d, g_2^d)$ be an instance of the bilinear decisional Diffie-Hellman problem. We will show that we can use any efficient adversary $\mathcal{A}_3$ can be used to break the above problem instance. To do so, we will build a reduction algorithm $\mathcal{R}$ that uses $\mathcal{A}_3$ in a black box manner, i.e. it plays the role of the challenger in the unforgeability experiment.

First $\mathcal{R}$ prepares the common reference string ρ by setting $Y_1 = g_1^a$, $Y_2 = g_2^a$, $\hat{g} = g_1^z$, for some $z \xleftarrow{\$} \mathbb{Z}_p^*$ and executes the trapdoor generation algorithm $(K_{\mathsf{PHF}}, \tau_{\mathsf{PHF}}) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g_1^a, g_1)$. Note that δ_ρ is not publicly known, so $\mathcal{R}$ does not have to know the exponent a but still knows z.

Next $\mathcal{R}$ prepares the public key $\mathsf{pk}_{\mathsf{SFPK}}$ and the trapdoor τ_{SFPK}. For this it uses the values g_1^b and g_2^b from the problem instance. It sets $\mathsf{pk}_{\mathsf{SFPK}} = (g_1, g_1^b)$ and $\tau_{\mathsf{SFPK}} = (g_2^b)$.

To answer $\mathcal{A}$'s signing queries for message m and randomness t_1 (which is equal to 1 for oracle $\mathcal{O}^1$), the reduction $\mathcal{R}$ follows the following steps:

- it chooses random values $t_2 \xleftarrow{\$} \mathbb{Z}_p^*$,
- it computes $M = g_1^{e'} \cdot \hat{g}^{s'}$ for some $e', s' \xleftarrow{\$} \mathbb{Z}_p^*$.
- it computes $(a_m, b_m) \xleftarrow{\$} \mathsf{PHF.TrapEval}(\tau_{\mathsf{PHF}}, M)$ and aborts if $a_m = 0$,
- it computes $\mathsf{pk}'_{\mathsf{SFPK}} \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, t_1)$,
- it computes:

$$\begin{aligned}
\mathsf{Sig}^2_{\mathsf{SFPK}} &= (g_1^b)^{-a_m^{-1} \cdot t_1} \cdot g_1^{t_2},\\
\mathsf{Sig}^3_{\mathsf{SFPK}} &= (g_2^b)^{-a_m^{-1} \cdot t_1} \cdot g_2^{t_2},\\
\mathsf{Sig}^1_{\mathsf{SFPK}} &= (g_1^a)^{t_2} \cdot ((g_1^b)^{(-a_m^{-1} \cdot t_1)} \cdot g_1^{t_2}))^{b_m},\\
e &= \mathsf{H}(m || \mathsf{Sig}^2_{\mathsf{SFPK}}, \mathsf{Sig}^3_{\mathsf{SFPK}}, \mathsf{pk}'_{\mathsf{SFPK}}),\\
s &= ((e' - e) + s' \cdot z)/z,
\end{aligned}$$

- set the signature $\mathsf{Sig}_{\mathsf{SFPK}} = (\mathsf{Sig}^1_{\mathsf{SFPK}}, \mathsf{Sig}^2_{\mathsf{SFPK}}, \mathsf{Sig}^3_{\mathsf{SFPK}}, s)$.

It is easy to see that this is a valid signature. Note that the a valid signature is of the form $(g_1^{a \cdot b \cdot t_1} \cdot ((g_1^a)^{a_m} \cdot g_1^{b_m})^r, g_1^r, g_2^r, s)$. In this case, the reduction has set $r = -a_m^{-1} \cdot b \cdot t_1 + t_2$ and this means that the $g_1^{a \cdot b \cdot t_1}$ cancels out and the reduction does not need to compute $g_1^{a \cdot b}$. Note that this only works because $a_m \neq 0$. Otherwise, this would not work.

It follows that for the forgery $(\mathsf{pk}^*_{\mathsf{SFPK}}, m^*, \mathsf{Sig}^*_{\mathsf{SFPK}}, s^*)$ of $\mathcal{A}$ we require that $(a_{m^*}, b_{m^*}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(\tau_{\mathsf{PHF}}, M^*)$ and $a_{M^*} = 0$, where $M^* = g_1^{e^*} \hat{g}^{s^*}$ and $e^* = \mathsf{H}(m^* || \mathsf{Sig}^2_{\mathsf{SFPK}} || \mathsf{Sig}^3_{\mathsf{SFPK}} || \mathsf{pk}^*_{\mathsf{SFPK}})$. In such a case, the reduction works as follows:

- parses $\mathsf{Sig}^*_{\mathsf{SFPK}}$ as $(\mathsf{Sig}^1_{\mathsf{SFPK}}, \mathsf{Sig}^2_{\mathsf{SFPK}}, \mathsf{Sig}^3_{\mathsf{SFPK}}, s^*)$,
- computes

$$g_1^{a \cdot b \cdot t^*} = \mathsf{Sig}^1_{\mathsf{SFPK}} \cdot (\mathsf{Sig}^2_{\mathsf{SFPK}})^{-b_{m^*}} = \left(g_1^{a \cdot b \cdot t^*} \cdot ((g_1^a)^{a_{m^*}} \cdot g_1^{b_{m^*}})^{r^*}\right) \cdot (g_1^{r^*})^{-b_{m^*}}$$

- parses $\mathsf{pk}^*_{\mathsf{SFPK}}$ and since for a valid forgery we have $\mathsf{pk}^*_{\mathsf{SFPK}} \in [\mathsf{pk}_{\mathsf{SFPK}}]_{\mathcal{R}}$, we have $\mathsf{pk}^*_{\mathsf{SFPK}} = (g_1^{t^*}, (g_1^b)^{t^*})$ and $\mathcal{R}$ can use $g_1^{t^*}$,

– outputs 1 iff
$$e(g_1^{a\cdot b\cdot t^*}, g_2^c) = e(g_1^{t^*}, g_2^d).$$

It is easy to see that the probability that $\mathcal{R}$ successfully solves the bilinear decisional Diffie-Hellman problem depends on the advantage of $\mathcal{A}$ and the probability that $\mathcal{R}$'s simulation succeeds. Since the programmable hash function PHF is $(1, \mathsf{poly}(\lambda))$-programmable and because this is a type 3 adversary, we conclude that this probability is non-negligible. Note that since in this case we use $\mathcal{A}_3$, M^* is distinct from all M's used in $\mathcal{O}^1$ and $\mathcal{O}^2$, which is not the case for type 1 and type 2 adversaries.

Theorem 2 (Adaptive Class-Hiding with Key Corruption). *Scheme 2 is adaptively class-hiding with key corruption in the common reference string model, assuming the decisional Diffie-Hellman assumption holds.*

Proof. In this proof we will use the game based approach. We start with $\mathbf{GAME}_0$ which is the original class-hiding experiment and let S_0 be an event that the experiment evaluates to 1, i.e. the adversary wins. We will use S_i to denote the event that the adversary wins the class-hiding experiment in $\mathbf{GAME}_i$.

Let $\mathsf{pk}_{\mathsf{SFPK}} = (A, B)$ be the public key given to the adversary, $\mathsf{pk}_0 = (A_0, B_0) = (g_1, g_1^{x_0})$ and $\mathsf{pk}_1 = (A_0, B_1) = (g_1, g_1^{x_1})$ be the public keys that are returned by $\mathsf{SFPK.KeyGen}$, $\mathsf{sk}_0 = (Y_1^{x_0}, \mathsf{pk}_0)$ and $\mathsf{sk}_1 = (Y_1^{x_1}, \mathsf{pk}_1)$ the corresponding secret keys given to the adversary and $\hat{b}$ be the bit chosen by the challenger.

$\mathbf{GAME}_0$: The original class-hiding game.

$\mathbf{GAME}_1$: In this game we do not use the $\mathsf{SFPK.ChgSK}$ algorithm to compute $\mathsf{sk}_{\mathsf{SFPK}}$ and $\mathsf{pk}_{\mathsf{SFPK}}$ but compute them as $\mathsf{pk}_{\mathsf{SFPK}} = (Q, Q^{x_{\hat{b}}})$, and $\mathsf{sk}_{\mathsf{SFPK}} = ((Q^{x_{\hat{b}}})^y, \mathsf{pk}_{\mathsf{SFPK}})$, where $Y_1 = g_1^y$ is part of the common reference string ρ generated by the challenger. In other words, instead of using the exponent r to randomize the public key and secret key, we use a group element Q to do it.

Since the distribution of the keys does not change, it follows that $\Pr[S_1] = \Pr[S_0]$. Note that the oracle can still use $\mathsf{sk}_{\mathsf{SFPK}}$ to compute valid signatures.

$\mathbf{GAME}_1$: In this game instead of computing $\mathsf{pk}_{\mathsf{SFPK}} = (Q, Q^{x_{\hat{b}}})$ as in $\mathbf{GAME}_1$, we sample $B' \xleftarrow{\$} \mathbb{G}_1$ and set $\mathsf{pk}_{\mathsf{SFPK}} = (Q, B')$.

We will show that this transition only lowers the adversaries advantage by a negligible fraction. This can be show by construction using a reduction $\mathcal{R}$ that uses an adversary $\mathcal{A}$ that can distinguish between those two games to break the decisional Diffie-Hellman assumption in $\mathbb{G}_1$.

Let $(g_1^\alpha, g_1^\beta, g_1^\gamma)$ be an instance of this problem in $\mathbb{G}_1$. $\mathcal{R}$ samples $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $B_0 = (g_1^\alpha)^{r_0}$, $B_1 = (g_1^\alpha)^{r_1}$. Note that in such a case, we also have to set $\mathsf{sk}_0 = ((B_0)^y, \mathsf{pk}_0)$ and $\mathsf{sk}_1 = ((B_1)^y, \mathsf{pk}_1)$. Additionally, the reduction uses $Q = g_1^\beta$ and the public key $\mathsf{pk}_{\mathsf{SFPK}} = (Q, (g_1^\gamma)^{r_{\hat{b}}})$. Note that the reduction can use the secret key $\mathsf{sk}_{\mathsf{SFPK}} = (((g_1^\gamma)^{r_{\hat{b}}})^y, \mathsf{pk}_{\mathsf{SFPK}})$ to generate signatures and answer signing queries.

Now $\gamma = \alpha \cdot \beta$ then $\mathsf{pk}_{\mathsf{SFPK}}$ has the same distribution as in $\mathbf{GAME}_1$ and otherwise as in $\mathbf{GAME}_2$. Thus, it follows that $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$.

We will now show that we have $\Pr[S_2] = \frac{1}{2}$. This follow from the fact that we have $\mathsf{pk}_{\mathsf{SFPK}} = (Q, B')$ and signatures of the form $\mathsf{Sig}_{\mathsf{SFPK}} = ((B')^y \cdot (\mathsf{PHF.Eval}(K_{\mathsf{PHF}}, m))^r, g_1^r, g_2^r, s)$ for some $r \in \mathbb{Z}_p^*$ and Q, B', which are independent from the bit $\hat{b}$. Thus, we have $\mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \Pr[S_0] \le \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

5 Our Group Signatures

In this section we formalize the group signature proposed in the introduction. We present the algorithms related to key generation and group management in Figure 3 and the algorithms related to signature creation, verification, tracing and judgment in Figure 4.

To securely instantiate our scheme we require the following components:

- a digital signature scheme DS which is existentially unforgeable under chosen message attacks,
- a structure-preserving signature scheme on equivalence classes SPS-EQ with message space of dimension $\ell = 2$ which is existentially unforgeable under chosen message attacks and perfectly adapts signatures,
- a compatible signature scheme with flexible public keys SFPK which has canonical representatives, is strongly existentially unforgeable under chosen message attacks and has adaptive class-hiding with key corruption,
- a public key encryption scheme PKE which is $\mathsf{IND-CPA}$ secure and has key-privacy under chosen message attacks,
- a proof system $\mathit{\Pi}_{\mathsf{PPE}}$ for pairing product equations which is a witness-indistinguishable proof of knowledge,
- a second proof system $\mathit{\Pi}_{\mathsf{GS.Judge}}$ which is zero-knowledge.

```
GS.Setup($1^\lambda$)
$(\rho_{\mathsf{SFPK}}, \cdot) \xleftarrow{\$} \mathsf{SFPK.CRSGen}(1^\lambda)$
$\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \xleftarrow{\$} \mathsf{SPS.BGGen}(1^\lambda)$
$\rho_{\mathcal{J}} \xleftarrow{\$} \mathit{\Pi}_{\mathsf{GS.Judge}}.\mathsf{Setup}(1^\lambda);\ K_2 \xleftarrow{\$} \mathbb{G}_2;\ \tau := 0$
return $\mathsf{param} := (1^\lambda, \mathsf{BG}, \rho_{\mathsf{SFPK}}, \rho_{\mathcal{J}}, K_2)$
```

```
GS.KGen$_{\mathcal{M}}$(param)
$(\mathsf{sk}_{\mathsf{DS}}, \mathsf{pk}_{\mathsf{DS}}) \xleftarrow{\$} \mathsf{DS.KeyGen}(1^\lambda)$
$(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) \xleftarrow{\$} \mathsf{SPS.KGen}(\mathsf{BG}, 2)$
$\mathsf{info} := (\mathsf{pk}_{\mathsf{SPS}}, \mathsf{DS.Sign}(\mathsf{sk}_{\mathsf{DS}}, \mathsf{pk}_{\mathsf{SPS}}), \emptyset)$
return $(\mathsf{msk} := (\mathsf{sk}_{\mathsf{DS}}, \mathsf{sk}_{\mathsf{SPS}}),$
       $\mathsf{mpk} := \mathsf{pk}_{\mathsf{DS}}, \mathsf{info})$
```

```
GS.KGen$_{\mathcal{T}}$(param)
$\omega \xleftarrow{\$} \mathsf{coin};$
$(\rho_{\mathit{\Pi}}, \tau_{\mathit{\Pi}}) \leftarrow \mathit{\Pi}_{\mathsf{PPE}}.\mathsf{ExtGen}(1^\lambda; \omega)$
return $(\mathsf{tsk} := (\tau_{\mathit{\Pi}}, \omega),$
        $\mathsf{tpk} := \rho_{\mathit{\Pi}})$
```

```
GS.Issue($\mathsf{info}_{\tau_{current}}, \mathsf{msk}, \mathsf{uid}, \boldsymbol{upk}[\mathsf{uid}]$)
$\mathsf{msk} = (\mathsf{sk}_{\mathsf{DS}}, \mathsf{sk}_{\mathsf{SPS}}), \boldsymbol{upk}[\mathsf{uid}] = (\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{Enc}})$
$\mathsf{info}_{\tau_{current}}[\mathsf{uid}] = (\mathsf{pk}_{\mathsf{SPS}}, \sigma_{\mathsf{DS}}, \mathsf{Active})$
if $\neg\mathsf{IsCanonical}(\mathsf{pk}_{\mathsf{SFPK}})$ then return $\perp$
$k \xleftarrow{\$} \mathsf{coin}; c \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{Enc}}, k);$
$\sigma_{\mathsf{SPS}} \xleftarrow{\$} \mathsf{SPS.Sign}(\mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, k), \mathsf{sk}_{\mathsf{SPS}})$
$\mathsf{Active}' := \mathsf{Active} \cup \{(c, \sigma_{\mathsf{SPS}})\}$
$\mathsf{info}_{\tau_{current}}[\mathsf{uid}] := (\mathsf{pk}_{\mathsf{SPS}}, \sigma_{\mathsf{DS}}, \mathsf{Active}')$
$\boldsymbol{reg}[\mathsf{uid}] := \boldsymbol{upk}[\mathsf{uid}]$
```

```
GS.UpdateGroup($\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{current}}, \mathcal{S}, \boldsymbol{reg}$)
$\mathsf{msk} = (\mathsf{sk}_{\mathsf{DS}}, \mathsf{sk}_{\mathsf{SPS}})$
$(\mathsf{sk}'_{\mathsf{SPS}}, \mathsf{pk}'_{\mathsf{SPS}}) \xleftarrow{\$} \mathsf{SPS.KGen}(\mathsf{BG}, 2)$
$\mathsf{msk} := (\mathsf{sk}_{\mathsf{DS}}, \mathsf{sk}'_{\mathsf{SPS}})$
$\mathsf{info}_{\tau_{current}} = (\cdot, \cdot, \mathsf{Active});\ A := \{i |\ \text{user } i \text{ is active}\}$
foreach $i \in A \setminus \mathcal{S}$
  $\boldsymbol{reg}[\mathsf{uid}] = (\mathsf{pk}^i_{\mathsf{SFPK}}, \mathsf{pk}^i_{\mathsf{Enc}})$
  $k \xleftarrow{\$} \mathsf{coin}; c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}^i_{\mathsf{Enc}}, k);$
  $\sigma_{\mathsf{SPS}} \xleftarrow{\$} \mathsf{SPS.Sign}(\mathsf{SFPK.ChgPK}(\mathsf{pk}^i_{\mathsf{SFPK}}, k), \mathsf{sk}_{\mathsf{SPS}})$
  $\mathsf{Active}' := \mathsf{Active}' \cup (c, \sigma_{\mathsf{SPS}})$
return $\mathsf{info}_{\tau_{new}} = (\mathsf{pk}'_{\mathsf{SPS}}, \mathsf{DS.Sign}(\mathsf{sk}_{\mathsf{DS}}, \mathsf{pk}'_{\mathsf{SPS}}), \mathsf{Active}')$
```

```
GS.UKGen($1^\lambda$)
$(\mathsf{sk}_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{SFPK}}) \xleftarrow{\$} \mathsf{SFPK.KeyGen}(1^\lambda)$
$(\mathsf{sk}_{\mathsf{Enc}}, \mathsf{pk}_{\mathsf{Enc}}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda)$
return $(\boldsymbol{usk}[\mathsf{uid}] := (\mathsf{sk}_{\mathsf{SFPK}}, \mathsf{sk}_{\mathsf{Enc}}),$
        $\boldsymbol{upk}[\mathsf{uid}] := (\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{Enc}}))$
```

Fig. 3. Algorithms related to key generation and group management.

```
$\mathsf{GS.Sig}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m)$
$\mathsf{info}_{\tau_{current}} = (\mathsf{pk}_{\mathsf{SPS}}, \cdot, \mathsf{Active})$
$\mathbf{gsk}[\mathsf{uid}] = (\mathsf{sk}_{\mathsf{SFPK}}, \mathsf{sk}_{\mathsf{Enc}})$; $\mathsf{gpk}[\mathsf{uid}] = (\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{Enc}})$
if $\neg\exists\ (c, \sigma_{\mathsf{SPS}}) \in \mathsf{Active}$ s. t.
  $k \leftarrow \mathsf{PKE.Dec}(c, \mathsf{sk}_{\mathsf{Enc}})$ and
  $\mathsf{SPS.Verify}(\mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, k), \sigma_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 1$
   then return $\perp$
$r \xleftarrow{\$} \mathsf{coin}$; $\mathsf{pk}'_{\mathsf{SFPK}} \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r)$; $\mathsf{sk}'_{\mathsf{SFPK}} \xleftarrow{\$} \mathsf{SFPK.ChgSK}(\mathsf{sk}_{\mathsf{SFPK}}, r)$
$\sigma'_{\mathsf{SPS}} \xleftarrow{\$} \mathsf{SPS.ChgRep}(\mathsf{pk}_{\mathsf{SFPK}}, \sigma_{\mathsf{SPS}}, r \cdot k^{-1}, \mathsf{pk}_{\mathsf{SPS}})$
$\Pi_{\mathsf{SFPK}} \xleftarrow{\$}$ $\Pi_{\mathsf{PPE}}.\mathsf{Prove}(\rho_\Pi, x_{\mathsf{Sign}}, w = (\mathsf{pk}_{\mathsf{SFPK}}, r, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2}))$ for $x_{\mathsf{Sign}}$:
      $\exists\ (\mathsf{pk}_{\mathsf{SFPK}}, r, w_1, w_2)$ s. t.
        $\mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r) = \mathsf{pk}'_{\mathsf{SFPK}} \ \wedge\ \mathsf{IsCanonical}(\mathsf{pk}_{\mathsf{SFPK}})$
        $\vee\ \ e(w_1, K_2) = e(w_2, g_2)$
$\mathsf{Sig}_{\mathsf{SFPK}} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk}'_{\mathsf{SFPK}}, m || \tau_{current} || \mathsf{pk}'_{\mathsf{SFPK}} || \sigma'_{\mathsf{SPS}} || \Pi_{\mathsf{SFPK}})$
return $\Sigma := (\mathsf{pk}'_{\mathsf{SFPK}}, \sigma'_{\mathsf{SPS}}, \Pi_{\mathsf{SFPK}}, \mathsf{Sig}_{\mathsf{SFPK}})$
```

```
$\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma)$
$\mathsf{info}_\tau = (\mathsf{pk}_{\mathsf{SPS}}, \sigma_{\mathsf{DS}}, \cdot)$; $\mathsf{mpk} = \mathsf{pk}_{\mathsf{DS}}$
$\Sigma = (\mathsf{pk}_{\mathsf{SFPK}}, \sigma_{\mathsf{SPS}}, \Pi_{\mathsf{SFPK}}, \mathsf{Sig}_{\mathsf{SFPK}})$
// $x_{\mathsf{Sign}}$ is the same statement as in GS.Sig
if $\mathsf{DS.Verify}(\mathsf{pk}_{\mathsf{DS}}, \mathsf{pk}_{\mathsf{SPS}}, \sigma_{\mathsf{DS}}) = 0$ or
  $\Pi_{\mathsf{PPE}}.\mathsf{Verify}(\rho_\Pi, x_{\mathsf{Sign}}, \Pi_{\mathsf{SFPK}}) = 0$ or
  $\mathsf{SPS.Verify}(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SFPK}}, \sigma_{\mathsf{SPS}}) = 0$
 then return $0$
return $\mathsf{SFPK.Verify}(\mathsf{pk}_{\mathsf{SFPK}}, m || \tau || \mathsf{pk}_{\mathsf{SFPK}} || \sigma_{\mathsf{SPS}} || \Pi_{\mathsf{SFPK}}, \mathsf{Sig}_{\mathsf{SFPK}})$
```

```
$\mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \boldsymbol{reg}, m, \Sigma)$
$\Sigma = (\mathsf{pk}_{\mathsf{SFPK}}, \sigma_{\mathsf{SPS}}, \Pi_{\mathsf{SFPK}}, \mathsf{Sig}_{\mathsf{SFPK}})$; $\mathsf{tsk} = (\tau_\Pi, \omega)$
$(\mathsf{pk}_{\mathsf{SFPK}}, r, \cdot, \cdot) \xleftarrow{\$} \Pi_{\mathsf{PPE}}.\mathsf{Extract}(\tau_\Pi, \Pi_{\mathsf{SFPK}})$
if $\neg\exists \mathsf{uid}$ s. t. $\boldsymbol{reg}[\mathsf{uid}] = (\mathsf{pk}_{\mathsf{SFPK}}, \cdot)$
 then return $\perp$
$\pi \xleftarrow{\$}$ $\Pi_{\mathsf{GS.Judge}}.\mathsf{Prove}(\rho_{\mathcal{J}}, x_{\mathsf{Trace}}, w = (\tau_\Pi, \omega))$ for $x_{\mathsf{Trace}}$:
    $\exists\ (\tau_\Pi, \omega)$ s. t.
      $(\mathsf{pk}_{\mathsf{SFPK}}, \cdot, \cdot, \cdot) \xleftarrow{\$} \Pi_{\mathsf{PPE}}.\mathsf{Extract}(\tau_\Pi, \Pi_{\mathsf{SFPK}})\ \wedge$
      $(\rho_\Pi, \tau_\Pi) \leftarrow \Pi_{\mathsf{PPE}}.\mathsf{ExtGen}(1^\lambda; \omega)$
return $(\mathsf{uid}, \pi)$
```

```
$\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi_{\mathrm{Trace}}, \boldsymbol{upk}[\mathsf{uid}], m, \Sigma)$
if $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$
 then return $0$
$\Sigma = (\cdot, \cdot, \Pi_{\mathsf{SFPK}}, \cdot)$; $\boldsymbol{upk}[\mathsf{uid}] = (\mathsf{pk}_{\mathsf{SFPK}}, \cdot)$
// Statement $x_{\mathsf{Trace}}$ as in GS.Trace
return $\Pi_{\mathsf{GS.Judge}}.\mathsf{Verify}(\rho_{\mathcal{J}}, x, \pi)$
```

Fig. 4. Algorithms related to creating and processing signatures.

Theorem 3 (Traceability). *Our group signature construction is traceable if the decisional Diffie-Hellman assumption holds in $\mathbb{G}_2$, the* SPS-EQ *signature scheme is existential unforgeable under chosen-message attacks, the* SFPK *scheme is existential unforgeable and the digital signature scheme used by the Issuer is existential unforgeable under chosen-message attacks.*

Proof. We will use the game base approach. Let us denote by S_i the event that the adversary wins the traceability experiment in $\mathbf{GAME}_i$. Let $(m^*, \Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}}), \mathsf{info}^*_\tau = (\mathsf{pk}^*_{\mathsf{SPS}}, \sigma^*_{\mathsf{DS}}, \mathsf{Active}^*))$ be the forgery outputted by the adversary. Moreover, let u be the maximum number of oracle queries to UpdateGroup made by the adversary and n the number of queries to the AddU oracle.

GAME$_0$: The original experiment.

GAME$_1$: We abort in the case that $\mathsf{Verify}_{\mathsf{DS}}(\mathsf{pk}_{\mathsf{DS}}, \mathsf{pk}^*_{\mathsf{SPS}}, \sigma^*_{\mathsf{DS}}) = \mathsf{accept}$ but the signature σ^*_{DS} was not created by the $\mathsf{UpdateGroup}$ oracle. Informally, we exclude the case that the adversary creates a custom $\mathsf{SPS\text{-}EQ}$ public key and uses it to create his own epoch information.

It is easy to see that this change only decreases the adversary's advantage by a negligible fraction. In particular, we can simply use any adversary $\mathcal{A}$ to break the existential unforgeability of the digital signature scheme used by the Issuer. Thus, it follows that $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{DS}}(\lambda)$.

GAME$_2$: We abort in case the witness extracted by the tracing authority from Π^*_{SFPK} is $(\cdot, \cdot, w_1, w_2)$ such that $w_1 \neq 1_{\mathbb{G}_1}$ and $w_2 \neq 1_{\mathbb{G}_1}$.

Again, this change only decreases the adversary's advantage by a negligible fraction. We will show that we can use $\mathcal{A}$ to break the decisional Diffie-Hellman assumption in $\mathbb{G}_2$. Let $(g_2^\alpha, g_2^\beta, g_2^\gamma)$ be an instance of this problem. We set $K_2 = g_2^\alpha$. Since we always have that $e(w_1, K_2) = e(w_2, g_2)$, which follows from the fact that the tracing authority can extract such a witness and this implies that $w_2 = (w_1)^\alpha$. Thus, we can return $e(w_2, g_2^\beta) = e(w_1, g_2^\gamma)$.

We have shown that $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

GAME$_3$: Choose $u^* \xleftarrow{\$} \{1, \ldots, u\}$ and abort if info^*_τ is not the output of the u^*-th call to the $\mathsf{UpdateGroup}$ oracle.

Because of the changes made by the previous game we know that the adversary can only use epoch information outputted by this oracle. Thus, we have $\Pr[S_2] = u \cdot \Pr[S_3]$.

GAME$_4$: We abort in case $\mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}^*_\tau, \boldsymbol{reg}, m^*, \Sigma^*) = \perp$ but $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}^*_\tau, m^*, \Sigma^*) = \mathsf{accept}$. Informally, we exclude the case that the adversary creates a new user from outside the group, i.e. a new $\mathsf{SPS\text{-}EQ}$ signature.

We will show that any adversary $\mathcal{A}$ returns a forgery for which we abort, can be used to break the existential unforgeability of the $\mathsf{SPS\text{-}EQ}$ signature scheme. The reduction $\mathcal{R}$ algorithm on input of the public key $\mathsf{pk}_{\mathsf{SPS}}$ performs as follows. It first sets $\mathsf{info}_{u^*} = (\mathsf{pk}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{DS}}(\mathsf{sk}_{\mathsf{DS}}, \mathsf{pk}_{\mathsf{SPS}}), \mathsf{Active})$. For every active user i is this epoch, Active contains a tuple $(\mathsf{Enc}(\mathsf{pk}^i_{\mathsf{Enc}}, k_i), \sigma^i_{\mathsf{SPS}})$, where σ^i_{SPS} is a signature generated for $\mathcal{R}$ by the signing oracle on input $\mathsf{SFPK.ChgPK}(\mathsf{pk}^i_{\mathsf{SFPK}}, k_i)$. It then runs the system for $\mathcal{A}$ according to description.

After some interactions, the adversary returns $(m^*, \Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}}), \mathsf{info}^*_\tau = (\mathsf{pk}^*_{\mathsf{SPS}}, \sigma^*_{\mathsf{DS}}, \mathsf{Active}^*))$. Note that because of the changes in the previous games, we know that $\mathsf{pk}^*_{\mathsf{SPS}} = \mathsf{pk}_{\mathsf{SPS}}$, i.e. the forgery is created for a epoch that uses our challenged $\mathsf{SPS\text{-}EQ}$ public key to certify members. Finally, the reduction $\mathcal{R}$ returns $(\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}})$ as a valid forgery. It is easy to see that this is a valid solution. Note that since opening failed, this means that the trusted authority extracted witness $w = (\mathsf{pk}_{\mathsf{SFPK}}, r^*, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2})$ and $\mathsf{pk}_{\mathsf{SFPK}}$ is not a public key of any honest user. Moreover, since $\mathsf{pk}_{\mathsf{SFPK}}$ is canonical representative of $[\mathsf{pk}^*_{\mathsf{SFPK}}]_{\mathcal{R}}$ it follows that this is in fact a valid forgery.
We conclude that $|\Pr[S_4] - \Pr[S_3]| \leq \mathsf{Adv}^{\ell,\mathsf{euf\text{-}cma}}_{\mathsf{SPS\text{-}EQ},\mathcal{A}}(\lambda)$.

Finally, we will show hat any adversary $\mathcal{A}$ that has non-negligible advantage in winning traceability experiment in $\mathbf{GAME}_4$ can be used by a reduction algorithm $\mathcal{R}$ to break the existential unforgeability of the SFPK scheme for a public key $\mathsf{pk}_{\mathsf{SFPK}}$.

The reduction simulator works as follows. It generates all values according to description but for $i \xleftarrow{\$} [n]$ the reduction answers the i-th queries of the adversary to AddU by setting $\boldsymbol{upk}[] = (\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{Enc}})$ for some $(\mathsf{sk}_{\mathsf{Enc}}, \mathsf{pk}_{\mathsf{Enc}}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{Enc}}(1^\lambda)$. The reduction aborts if at some point the adversary asks for the group secret key of this member.

To answer signing queries $\mathsf{Sign}(i, m, \tau)$ for this member, the reduction parses $\mathsf{info}_\tau = (\mathsf{pk}_{\mathsf{SPS}}, \cdot, \mathsf{Active})$. Then it returns $\perp$ if for all tuples $(E, \sigma_{\mathsf{SPS}})$ in Active the decryption $k \xleftarrow{\$} \mathsf{Dec}(E, \mathsf{sk}_{\mathsf{Enc}})$ fails. $\mathcal{R}$ chooses random coins $r \xleftarrow{\$} \mathsf{coin}$, randomizes the flexible public key $\mathsf{pk}'_{\mathsf{SFPK}} \xleftarrow{\$} \mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r)$ and the signature $\sigma'_{\mathsf{SPS}} \xleftarrow{\$} \mathsf{SPS.ChgRep}(\mathsf{pk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SFPK}}, \sigma_{\mathsf{SPS}}, r \cdot k^{-1})$. It then creates a proof $\mathit{\Pi}_{\mathsf{SFPK}}$ for the statement:

$$\begin{aligned} x = \{ \exists_{\mathsf{pk}_{\mathsf{SFPK}}, r, w_1, w_2} \ & \mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r) = \mathsf{pk}'_{\mathsf{SFPK}} \ \wedge \\ & \mathsf{pk}_{\mathsf{SFPK}} \text{ is a canonical representative } \ \vee \\ & e(w_1, K_2) = e(w_2, g_2) \ \} \end{aligned}$$

using witness $w = (\mathsf{pk}_{\mathsf{SFPK}}, r, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2})$. It then uses its own signing oracle $\mathcal{O}^2((m||\tau||\mathsf{pk}'_{\mathsf{SFPK}}||\sigma'_{\mathsf{SPS}}||\mathit{\Pi}_{\mathsf{SFPK}}), r)$, receiving signature $\mathsf{Sig}_{\mathsf{SFPK}}$. Finally, it outputs $\mathit{\Sigma} = (\mathsf{pk}'_{\mathsf{SFPK}}, \sigma'_{\mathsf{SPS}}, \mathit{\Pi}_{\mathsf{SFPK}}, \mathsf{Sig}_{\mathsf{SFPK}})$. Note that since values required to perform the above computations are known to $\mathcal{R}$, it can efficiently compute valid group signatures for this member.

Finally, $\mathcal{A}$ outputs a valid group signature $(m^*, \mathit{\Sigma}^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \mathit{\Pi}^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}}), \mathsf{info}^*_\tau = (\mathsf{pk}^*_{\mathsf{SPS}}, \sigma^*_{\mathsf{DS}}, \mathsf{Active}^*))$ and the reduction algorithm outputs $((m^*||\tau^*||\mathsf{pk}^*_{\mathsf{SFPK}}||\sigma^*_{\mathsf{SPS}}||\mathit{\Pi}^*_{\mathsf{SFPK}}), \mathsf{Sig}^*_{\mathsf{SFPK}})$ as a valid SFPK forgery. Note that this is only true if $\mathsf{pk}^*_{\mathsf{SFPK}}$ and $\mathsf{pk}_{\mathsf{SFPK}}$ are in the same equivalence class. By the changes made in the previous games we know that $\mathsf{pk}^*_{\mathsf{SFPK}}$ is in a relation with a public key of an honest user and with probability $1/n$ we guessed the correct member for which $\mathsf{Trace}(m^*, \mathit{\Sigma}^*, \mathsf{info}^*_\tau) = i$ and we have set his public key to $\mathsf{pk}_{\mathsf{SFPK}}$. Note that also in such a case we do not have to worry about a corruption query for this member, since the forgery must be for non-corrupted users. We conclude that since m^* was never queried previously, the reduction also never used the prefix m^* in its oracle queries.

In the end we have:

$$\Pr[S_0] \le u \cdot \left(n \cdot \mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\ell,\mathsf{euf\text{-}cma}}_{\mathsf{SPS\text{-}EQ},\mathcal{A}}(\lambda)\right) + \mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{DS}}(\lambda) + \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda).$$

Theorem 4 (Anonymity). *Our group signature construction is anonymous if the decisional Diffie-Hellman assumption holds in $\mathbb{G}_2$, the* $\mathsf{SPS\text{-}EQ}$ *signature scheme perfectly adapts signatures, the* SFPK *scheme is adaptively class-hiding with key corruption and strongly existential unforgeable, the proof system used by signers is witness-indistinguishable and the proof system used by the tracing authority is zero-knowledge.*

Proof. We will use the game base approach. Let us denote by S_i the event that the adversary wins the anonymity experiment in $\mathbf{GAME}_i$. Moreover, let n be the number of queries to the AddU oracle made by the adversary and let $(\mathsf{info}^*_\tau, \mathsf{uid}^*_1, \mathsf{uid}^*_2, m^*)$ be the query made to the Chall_b oracle, which outputs $\mathit{\Sigma}^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \mathit{\Pi}^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$.

$\mathbf{GAME}_0$: The original experiment.

GAME$_1$: We abort in case the witness extracted inside the Trace oracle (by the $\mathsf{GS.Trace}$ algorithm) is $(\cdot, \cdot, w_1, w_2)$ such that $w_1 \neq 1_{\mathbb{G}_1}$ and $w_2 \neq 1_{\mathbb{G}_1}$.

This change only decreases the adversary's advantage by a negligible fraction. We will show that we can use $\mathcal{A}$ to break the decisional Diffie-Hellman assumption in $\mathbb{G}_2$. Let $(g_2^\alpha, g_2^\beta, g_2^\gamma)$ be an instance of this problem. We set $K_2 = g_2^\alpha$. Since we always have that $e(w_1, K_2) = e(w_2, g_2)$, which follows from the fact that the tracing authority can extract such a witness and implies that $w_2 = (w_1)^\alpha$. Thus, we can return $e(w_2, g_2^\beta) = e(w_1, g_2^\gamma)$.

We have shown that $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

GAME$_2$: We now simulate the proof generated in $\mathsf{GS.Trace}$ by the tracing authority.

Obviously, we only lower the advantage of the adversary by a negligible fraction because of the zero-knowledge property of this proof. Thus, we have $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{A},\Pi_{\mathsf{GS.Judge}}}(\lambda)$.

GAME$_3$: We now replace the way the proof Π^*_{SFPK} is computed. Instead of using witness $w = (\mathsf{pk}_{\mathsf{SFPK}}, r, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2})$, where $\mathsf{pk}^*_{\mathsf{SFPK}} = \mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{pk})$ we use the witness $w = (\cdot, \cdot, w_1, w_1^k)$ for some $w_1 \xleftarrow{\$} \mathbb{G}_1$, k such that $K_2 = g_2^k$ and K_2 is part of param.

Note that until now we did not exclude the case that the adversary somehow randomizes the challenged signature Σ^* and queries it to the Trace oracle. This could be e.g. possible if the used SFPK signatures would not be strongly existential unforgeable. Thus, we have to show that we can still execute the Trace oracle as in an original execution. Fortunately, Π^*_{SFPK} is the only proof in the system with this trapdoor witness (all other cases were excluded in **GAME**$_1$) and if we extract this value in the Trace oracle, we can return the correct identity. Note that in **GAME**$_2$ we simulate the proof created by the tracing authority, so we can create a valid proof without the correct values. It follow that by witness-indistinguishability we have that $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\mathsf{wi}}_{\mathcal{A},\Pi_{\mathsf{PPE}}}(\lambda)$.

GAME$_4$: We now change the way we compute σ^*_{SPS} . Instead of using the $\mathsf{SPS.ChgRep}$ algorithm to change representation of an old signature, we compute the $\mathsf{SPS\text{-}EQ}$ signature directly on $\mathsf{pk}^*_{\mathsf{SFPK}}$.

Since the $\mathsf{SPS\text{-}EQ}$ signature scheme perfectly adapts signatures, we have $\Pr[S_4] = \Pr[S_3]$

GAME$_5$: Given the experiments bit b, we choose index $i \xleftarrow{\$} [n]$ and abort if uid_b does not correspond to the user created in the i-th query of the adversary to AddU.

We have $\Pr[S_4] = n \cdot \Pr[S_5]$.

GAME$_6$: Let $\mathsf{pk}_{\mathsf{SFPK}}$ be the SFPK public key of the user chosen in the previous game. We now instead of using $\mathsf{pk}_{\mathsf{SFPK}}$ to create $\mathsf{pk}^*_{\mathsf{SFPK}}$, we use a fresh key generated using $\mathsf{KeyGen}_{\mathsf{SFPK}}$.

We will now show that any adversary $\mathcal{A}$ that can distinguish those games, can be used to brake the weak class-hiding of the SFPK scheme. We will show how to build a reduction $\mathcal{R}$ that does this. Let $(\mathsf{sk}^0_{\mathsf{SFPK}}, \mathsf{pk}^0_{\mathsf{SFPK}})$, $(\mathsf{sk}^1_{\mathsf{SFPK}}, \mathsf{pk}^1_{\mathsf{SFPK}})$ and $\mathsf{pk}'_{\mathsf{SFPK}}$ be the inputs given to $\mathcal{R}$ by the challenger in the adaptive class-hiding experiment. The reduction then sets $\mathsf{pk}^0_{\mathsf{SFPK}}$ as the i-th honest user SFPK public key. All other key material for those users is constructed as described in the scheme. Now in order to answer the query $(\mathsf{info}^*_\tau, \mathsf{uid}^*_1, \mathsf{uid}^*_2, m^*)$ to the Chall_b oracle, the reduction:

- sets $\mathsf{pk}^*_{\mathsf{SFPK}} = \mathsf{pk}'_{\mathsf{SFPK}}$,
- computes σ^*_{SPS} as in $\mathbf{GAME}_4$,
- computes Π^*_{SFPK} as in $\mathbf{GAME}_3$,
- asks its signing oracle for $\mathsf{Sig}^*_{\mathsf{SFPK}}$ under message $m^*||\tau^*||\mathsf{pk}^*_{\mathsf{SFPK}}||\sigma^*_{\mathsf{SPS}}||\Pi^*_{\mathsf{SFPK}}$,

and returns $\Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$. Note that since it knows $\mathsf{sk}^0_{\mathsf{SFPK}}$ and $\mathsf{sk}^1_{\mathsf{SFPK}}$ it can easily answer all corruption queries made by $\mathcal{A}$. In the end $\mathcal{A}$ outputs a bit b, which is also returned by $\mathcal{R}$.

It follow that we have $|\Pr[S_6] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$.

We now argue that the only way the adversary $\mathcal{A}$ can break anonymity is by somehow creating a randomization $\Sigma' = (\mathsf{pk}'_{\mathsf{SFPK}}, \sigma'_{\mathsf{SPS}}, \Pi'_{\mathsf{SFPK}}, \mathsf{Sig}'_{\mathsf{SFPK}})$ of the signature $\Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$ and use Σ' in a query to the Trace oracle. Since in $\mathbf{GAME}_6$ we changed the public key $\mathsf{pk}^*_{\mathsf{SFPK}}$ to a random one, this is the only part of the simulation, where the adversary can notice something. Thus, for this to work the adversary must use a valid signature $\mathsf{Sig}'_{\mathsf{SFPK}}$ for $\mathsf{pk}'_{\mathsf{SFPK}} \in [\mathsf{pk}^*_{\mathsf{SFPK}}]_{\mathcal{R}}$. We distinguish two cases: $\mathsf{Sig}'_{\mathsf{SFPK}} = \mathsf{Sig}^*_{\mathsf{SFPK}}$ and $\mathsf{Sig}'_{\mathsf{SFPK}} \neq \mathsf{Sig}^*_{\mathsf{SFPK}}$. If $\mathsf{Sig}'_{\mathsf{SFPK}} = \mathsf{Sig}^*_{\mathsf{SFPK}}$ this means that $\mathsf{pk}'_{\mathsf{SFPK}} = \mathsf{pk}^*_{\mathsf{SFPK}}$ and either $\sigma'_{\mathsf{SPS}} \neq \sigma^*_{\mathsf{SPS}}$ or $\Pi'_{\mathsf{SFPK}} \neq \Pi^*_{\mathsf{SFPK}}$. Since $\mathsf{pk}^*_{\mathsf{SFPK}}$ is set to random public key in $\mathbf{GAME}_6$ we can use an adversary that creates such a signature Σ' to break strong existential unforgeability of the SFPK scheme. In case $\mathsf{Sig}'_{\mathsf{SFPK}} \neq \mathsf{Sig}^*_{\mathsf{SFPK}}$, we notice that in order for the adversary to see that this is a simulation the public key $\mathsf{pk}'_{\mathsf{SFPK}}$ must be in relation to $\mathsf{pk}^*_{\mathsf{SFPK}}$. Thus, we can again use the adversary to break the strong existential unforgeability of the SFPK scheme, even if $\sigma'_{\mathsf{SPS}} = \sigma^*_{\mathsf{SPS}}$, $\Pi'_{\mathsf{SFPK}} = \Pi^*_{\mathsf{SFPK}}$ and $\mathsf{pk}'_{\mathsf{SFPK}} = \mathsf{pk}^*_{\mathsf{SFPK}}$.

In other words, the only way the adversary can randomize the challenged signature is by randomizing the SFPK signature because the other values are signed. However, since the scheme is strongly unforgeable the adversary has negligible chances to do so. It follows that $\Pr[S_6] = \mathsf{Adv}^{\mathsf{seuf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$.

In the end we have:

$$\begin{aligned}\Pr[S_0] \leq & n \cdot \left(\mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\mathsf{seuf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)\right) + \mathsf{Adv}^{\mathsf{wi}}_{\mathcal{A},\Pi_{\mathsf{PPE}}}(\lambda) + \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda) + \\ & \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{A},\Pi_{\mathsf{GS.Judge}}}(\lambda).\end{aligned}$$

Theorem 5 (Non-frameability). *Our group signature construction is non-frameable if the* SFPK *scheme is existential unforgeable and the proof system used by the tracing authority is sound.*

Proof. We again use the game base approach. Let us denote by S_i the event that the adversary wins the anonymity experiment in $\mathbf{GAME}_i$. Moreover, let n be the number of queries to the CrptU oracle made by the adversary and let $(m^*, \Sigma^*, \mathsf{uid}^*, \pi^*_{\mathrm{Trace}}, \mathsf{info}^*_\tau)$ be the output of the adversary $\mathcal{A}$.
$\mathbf{GAME}_0$: The original experiment.

$\mathbf{GAME}_1$: Let $\Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$. We extract the witness $w' = (\mathsf{pk}'_{\mathsf{SFPK}}, r', 1_{\mathbb{G}_1}, 1_{\mathbb{G}_2})$ from Π^*_{SFPK} using the tracing authorities secret key $\mathsf{tsk} = (\tau_\Pi)$. We abort if $\mathsf{pk}'_{\mathsf{SFPK}} \neq \boldsymbol{upk}[\mathsf{uid}^*]$ but the $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}^*, \mathsf{info}^*_\tau, \pi^*_{\mathrm{Trace}}, \boldsymbol{upk}[\mathsf{uid}^*], m^*, \Sigma^*)$ outputs accept.

We will show that this lowers the adversaries advantage only by a negligible fraction. In particular, this means that π^*_{Trace} is a valid proof for the statement:

$$\begin{aligned} x = \{ \ \exists_{\tau_\Pi,\omega} \ \ & (\mathsf{pk}^*_{\mathsf{SFPK}}, \cdot, \cdot, \cdot) \xleftarrow{\$} \mathsf{Extract}(\tau_\Pi, \Pi^*_{\mathsf{SFPK}}) \\ & (\rho_\Pi, \tau_\Pi) \leftarrow \mathsf{ExtGen}_{\mathsf{PPE}}(1^\lambda; \omega) \ \}.\end{aligned}$$

However, since we know that $\mathsf{pk}'_{\mathsf{SFPK}} \neq \boldsymbol{upk}[\mathsf{uid}^*]$ it follows that π^*_{Trace} is a proof that breaks the soundness property of the proof used by the tracing authority. We have shown that $|\Pr[S_1]-\Pr[S_0]| \leq \mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A},\Pi_{\mathsf{GS.Judge}}}(\lambda)$.

It is easy to see that $\Pr[S_1] = n \cdot \Pr[S_2]$.

We will now show that any adversary $\mathcal{A}$ that breaks the non-frameability of the scheme can be used to break the existential unforgeability of the SFPK scheme. To do so, we construct a reduction $\mathcal{R}$ that plays the role of the adversary in the existential unforgeability experiment. Let $\mathsf{pk}_{\mathsf{SFPK}}$ be the public key given to $\mathcal{R}$. The reduction sets $\boldsymbol{upk}[j] = \mathsf{pk}_{\mathsf{SFPK}}$, where j is the identifier from $\mathbf{GAME}_2$. To answer the queries to the Sign oracle for $\mathsf{uid} = j$, the reduction reduction outputs group signature $\Sigma' = (\mathsf{pk}'_{\mathsf{SFPK}}, \sigma'_{\mathsf{SPS}}, \Pi'_{\mathsf{SFPK}}, \mathsf{Sig}'_{\mathsf{SFPK}})$. To do so, the reduction can choose the randomization r freely and randomize the public key $\mathsf{pk}_{\mathsf{SFPK}}$ by running $\mathsf{pk}'_{\mathsf{SFPK}} \overset{\$}{\leftarrow} \mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r)$. It can also randomize the SPS-EQ signature to receive σ'_{SPS} and compute the proof Π'_{SFPK}. Finally, it uses its own signing oracle $\mathcal{O}^2$ to compute the SFPK signature $\mathsf{Sig}'_{\mathsf{SFPK}}$.

In the end, the adversary returns a group signature $\Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$ under message m^* and for epoch info^*_τ, for which we know (by $\mathbf{GAME_2}$) that $\mathsf{pk}^*_{\mathsf{SFPK}}$ is from the same relation as the public key $\mathsf{pk}_{\mathsf{SFPK}}$ from the existential unforgeability experiment. Since this is a valid forgery, it follows that $(\mathsf{uid}^*, m^*, \Sigma^*, \tau^*) \notin \mathcal{Q}$ and that $((m^*||\tau^*||\mathsf{pk}^*_{\mathsf{SFPK}}||\sigma^*_{\mathsf{SPS}}||\Pi^*_{\mathsf{SFPK}}), \mathsf{Sig}^*_{\mathsf{SFPK}})$ is a valid forgery against the SFPK scheme.

We conclude that $\Pr[S_0] = n \cdot \mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A},\Pi_{\mathsf{GS.Judge}}}(\lambda)$.

Theorem 6 (Functional Tracing Soundness). *Our group signature scheme has functional tracing soundness if the underlying* SFPK *scheme has canonical representatives, the proof system used by the Judge is sound and the proof system used by the signers is a proof of knowledge.*

Proof. Let $\mathcal{A}$ be an adversary against the tracing soundness of our scheme. We show how to construct a reduction $\mathcal{B}$ against the soundness of $\Pi_{\mathsf{GS.Judge}}$.

Given the CRS $\rho_{\mathsf{GS.Judge}}$, the reduction generates the remaining parameters according to $\mathsf{GS.Setup}$ and forwards them to the adversary. At some point the adversary will output the group and tracing manager's key material and the initial group information $(\mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk} = (\tau_{\mathsf{PPE}}, \omega), \mathsf{tpk} = \rho_{\mathsf{PPE}})$. Let us further denote by $(m, \Sigma, \{\mathsf{uid}_i, \pi_i\}_{i=1}^2, \mathsf{info}_\tau)$ the adversary's final output. We will assume that $\boldsymbol{upk}[\mathsf{uid}_1]$ and $\boldsymbol{upk}[\mathsf{uid}_2]$ are both defined and not equal.

Assume that

$$\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}_i, \mathsf{info}_\tau, \pi_i, \boldsymbol{upk}[\mathsf{uid}_i], m, \Sigma) = 1$$

for both $i = 1$ and $i = 2$, i.e. we have in particular

$$\mathsf{Verify}(\rho_{\mathsf{GS.Judge}}, x_{\mathsf{Trace}}, \pi_i) = 1$$

for both i.

We now consider the following cases:

Case I: The tracing manager's secret key is not properly generated, i.e. we have $(\tau', \rho') \leftarrow \mathsf{ExtGen}(1^\lambda; \omega)$ for $(\tau', \rho') \neq (\tau_{\mathsf{PPE}}, \rho_{\mathsf{PPE}})$. The reduction can check this, since the adversary provides the ω as part of the tracing manager's secret key. In this case, either of the two proofs π_i breaks the soundness of $\Pi_{\mathsf{GS.Judge}}$.

Case II: The tracing manager's secret key is properly generated. In this case, the reduction uses the extraction trapdoor to obtain the witness used for the proof Π_{SFPK} contained in the signature. There are two possibilities:

1. The extraction does not produce a valid witness. We bound this case by the advantage of $\mathcal{A}$ against the extractor.
2. The extraction is successful, yielding a valid witness $(\mathsf{pk}_{\mathsf{SFPK}}, r, w_1, w_2)$. Since the witness is valid, $\mathsf{pk}_{\mathsf{SFPK}}$ is the unique canonical representative of the key that created the signature. Since the keys $\boldsymbol{upk}[\mathsf{uid}_1]$ and $\boldsymbol{upk}[\mathsf{uid}_2]$ are different, at most one of them can be equal to the extracted $\mathsf{pk}_{\mathsf{SFPK}}$. The reduction thus returns $(x_{\mathsf{Trace}}, \pi_i)$ such that $\boldsymbol{upk}[uid_i] \neq \mathsf{pk}_{\mathsf{SFPK}}$ again breaking the soundness of $\Pi_{\mathsf{GS.Judge}}$.

Theorem 7 (Join Privacy). *Our group signature scheme has private joins if the encryption scheme used by the signers is* $\mathsf{IND-CPA}$ *secure and has* $\mathsf{IK-CPA}$ *key privacy and the* SFPK *scheme is adaptively class hiding with key corruption.*

Proof. We consider a series of games. In the following let uid_b be the challenge user who is inserted into the group and let $\mathsf{gpk}[\mathsf{uid}_b] = (\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{pk}_{\mathsf{Enc}})$ be their public key and $\mathbf{gsk}[\mathsf{uid}_b] = (\mathsf{sk}_{\mathsf{SFPK}}, \mathsf{sk}_{\mathsf{Enc}})$ be their secret key. Let S_i denote the event that the adversary wins in $\mathbf{GAME}_i$.

$\mathbf{GAME}_0$ Is the original join privacy game, hence $\Pr[S_0] = \mathsf{Adv}^{\mathsf{join-privacy}}_{\mathsf{GS},\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_1$ We modify how the challenge group information is created. For this we generate a fresh public key encryption key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda)$. After the challenge user uid_b is added using AddU, we replace his entry $(c = \mathsf{PKE.Enc}(\mathsf{pk}_b, k), \sigma_{\mathsf{SPS}})$ in the epoch information with $(\mathsf{PKE.Enc}(\mathsf{pk}, k), \sigma_{\mathsf{SPS}})$, i.e. we replace the encryption key of the randomness to a fresh key. It is easy to see that, since the encryption scheme has key privacy we have $\Pr[S_1] \leq \Pr[S_0] + \mathsf{Adv}^{\mathsf{ik-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_2$ In this game we further modify the ciphertext in the challenge user's part of info^* by encrypting the value 0 instead of the randomness used to change the SFPK key signed in σ_{SPS}. Because the encryption scheme is $\mathsf{IND-CPA}$ secure it holds that $\Pr[S_2] \leq \Pr[S_1] + \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_3$ Instead of changing the representative of user uid_b's SFPK public key, we generate a fresh public key and change its representative. The signature in info^* will now be on this fresh represenatative. We will also use this fresh key to sign in the queries made to $\mathsf{PrivChall}$. We observe that $\Pr[S_3] \leq \Pr[S_2] + \mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathsf{SFPK},\mathcal{A}}(\lambda)$. Further, we have $\Pr[S_3] = \frac{1}{2}$, since the updated epoch information and the signatures received from the challenge signing oracle are completely independent of the challenge users.

Putting it all together we thus have

$$\mathsf{Adv}^{\mathsf{join-privacy}}_{\mathsf{GS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ik-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathsf{SFPK},\mathcal{A}}(\lambda).$$

Theorem 8 (Leave Privacy). *Our group signature scheme has leave privacy if the encryption scheme used by the signers is* $\mathsf{IND-CPA}$ *secure and has* $\mathsf{IK-CPA}$ *key privacy and the* SFPK *scheme is adaptively class hiding with key corruption.*

Proof. This proof follows similar steps as the proof for join privacy. We consider a series of games, where in the first game b is fixed to 0 and in the last game, b is fixed to 1. Let S_i denote the event that $\mathcal{A}$'s final output in $\mathbf{GAME}_i$ is 0.

$\mathbf{GAME}_0$ The $\mathsf{Leave-Privacy}$ game, where bit b is fixed to 0.

GAME$_1$ We change the public key used to encrypt the epoch data for user uid_0 using the public key of user uid_1. We have
$$|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{ik-cpa}}_{\mathcal{A},\mathsf{PKE}}(\lambda).$$

GAME$_2$ We now change the randomness encrypted in this ciphertext to the randomness for user uid_1. Because of $\mathsf{IND-CPA}$ security of the encryption scheme we have
$$|\Pr[S_1] - \Pr[S_2]| \leq \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathcal{A},\mathsf{PKE}}(\lambda).$$

GAME$_3$ We change the SFPK public key to the public key of uid_1, also changing the signatures in $\mathsf{PrivChall}$ to this secret key. The game is now the same as the $\mathsf{Leave-Privacy}$ game with the bit fixed to 1. Because of adaptive class hiding we have
$$|\Pr[S_2] - \Pr[S_3]| \leq \mathsf{Adv}^{\mathsf{adaptc\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda).$$

5.1 Efficient Instantiation and Discussion

In this subsection we show how to efficiently instantiate our group signature construction. Our main objective is to minimize the signature size of our scheme while using only building blocks that are secure under standard assumptions and without random oracles.

We first take a look at a signature itself, which is composed of an SFPK public key $\mathsf{pk}'_{\mathsf{SFPK}}$, an SFPK signature $\mathsf{Sig}_{\mathsf{SFPK}}$, an $\mathsf{SPS\text{-}EQ}$ signature σ'_{SPS} and proof Π_{SFPK}. To instantiate SFPK signatures we use scheme 2, which means that $\mathsf{pk}'_{\mathsf{SFPK}}$ is 2 elements in $\mathbb{G}_1$ and $\mathsf{Sig}_{\mathsf{SFPK}}$ is 2 elements in $\mathbb{G}_1$, 1 in $\mathbb{G}_2$ and 1 in $\mathbb{Z}_p^*$. As mentioned in the introduction and similar to the static group signature in [6], we will instantiate the $\mathsf{SPS\text{-}EQ}$ with the scheme from [25]. This means that the $\mathsf{SPS\text{-}EQ}$ signature takes 10 elements in $\mathbb{G}_1$ and 4 elements in $\mathbb{G}_2$. Note that both building blocks are secure under standard assumptions.

To properly calculate the signature size, we have to instantiate the proof system Π_{PPE}. Recall, that this is a proof for the statement:

$$\begin{aligned}&\exists\ (\mathsf{pk}_{\mathsf{SFPK}}, r, w_1, w_2) \text{ s. t.}\\ &\mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r) = \mathsf{pk}'_{\mathsf{SFPK}} \ \wedge\ \mathsf{IsCanonical}(\mathsf{pk}_{\mathsf{SFPK}}) \vee\ e(w_1, K_2) = e(w_2, g_2).\end{aligned}$$

Taking into account that we will use Scheme 2, this statement can be simplified in a way that the scheme and the security proofs still work. Let $\mathsf{pk}'_{\mathsf{SFPK}} = (\mathsf{pk}'_1, \mathsf{pk}'_2)$ and $\mathsf{pk}_{\mathsf{SFPK}} = (\mathsf{pk}_1, \mathsf{pk}_2)$, we can then express this proof by the pairing product equations: $e(w_1, K_2) = e(w_2, g_2)$ and $e(\mathsf{pk}'_1, g_2^{r^{-1}}) = e(g_1, g_2) \cdot e(w_1, g_2)$. It is easy to see that the witness $(r, w_1, w_2) = (0, (g_1)^{-1}, (K_1)^{-1})$ is a trapdoor witness that can be used in the security proof to create a valid proof for an arbitrary $\mathsf{pk}'_{\mathsf{SFPK}}$. The canonical representative $\mathsf{pk}_{\mathsf{SFPK}}$ is only used by the tracing authority to open signatures. However, by extracting the witness $R = g_2^{r^{-1}}$ it can still do this because if $\mathsf{pk}'_2 = g_1^{x \cdot r}$, then $e(\mathsf{pk}'_2, R) = e(g_1^x, g_2)$ is a static value that is common for all public keys in relation with $\mathsf{pk}'_{\mathsf{SFPK}}$. Since the tracing authority has access to the registration table that contains public keys in canonical form of active members, we conclude that it can still correctly open signatures. This also does not influence any of the proofs.

Instantiating those equations using the fine-tuned Groth-Sahai proofs presented in [24] (assuming decisional Diffie-Hellman), the proof size is 10 elements in $\mathbb{G}_1$ and 8 elements in $\mathbb{G}_2$. This is constituted by: 1) two group elements in $\mathbb{G}_2$ for the first equation, which is linear, 2) four elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ for the second equation, 3) six elements in $\mathbb{G}_1$ for the three witnesses in $\mathbb{G}_1$, 4) two elements in $\mathbb{G}_2$ for the witness r. Overall the group signature is composed of 28 elements in $\mathbb{G}_1$, 15 in $\mathbb{G}_2$ and 1 in $\mathbb{Z}_p^*$. We provide a comparison with existing group signature schemes in Figure 5. Note that we do not include schemes from lattices in our comparison, because the only constant-size scheme was proposed by Ling et al. [39] and as argued by the authors the size is impractical.

It remains to show how to instantiate the digital signature scheme DS, the public key encryption scheme PKE and the proof system $\Pi_{\mathsf{GS.Judge}}$. The first two building blocks are standard and can easily instantiated in the standard model from simple assumptions. In case of PKE we can use the El Gamal encryption scheme, which is key-private. Finally, the system $\Pi_{\mathsf{GS.Judge}}$ can also be instantiated using Groth-Sahai proofs for pairing product equations [24]. Note that this basically means that the tracing authority has to prove correct decryption of an El-Gamal ciphertext and that its public key was generated using a DDH tuple, which can easily be expressed as pairing product equations.

Scheme	Signature size* [*bits*]	Membership	Assumptions
Libert-Peters-Yung [38]	8 448	static	standard
Boyen-Waters [18]‡	6 656	static	q-type
Boneh-Boyen-Shacham [11]	2 304	static	q-type
Bichsel et al. [10]	1 280	partially dynamic†	interactive
Groth [29]	13 056	partially dynamic	q-type
Libert-Peters-Yung [38]	14 848	partially dynamic	standard
Bootle et al. [15]	$O(\log N)$	fully dynamic	standard
Ours with [25]	13 056	fully dynamic + membership hiding	standard

* At a 256-bit (resp. 512-bit) representation of $\mathbb{Z}_q, \mathbb{G}_1$ (resp. $\mathbb{G}_2$) for Type 3 pairings and at a 3072-bit factoring and DL modulus with 256-bit key

† The scheme defines additionally a join↔issue procedure

‡ Adapted from type 1 to type 3 pairings as in [38]

Fig. 5. Comparison of Group Signature Schemes for N Active Members

References

[1] Michel Abdalla and Ricardo Dahab, eds. *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*. Vol. 10770. Lecture Notes in Computer Science. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-76581-5.

[2] Michel Abdalla and Bogdan Warinschi. "On the Minimal Assumptions of Group Signature Schemes". In: *Information and Communications Security, 6th International Conference, ICICS 2004, Malaga, Spain, October 27-29, 2004, Proceedings*. 2004. URL: https://doi.org/10.1007/978-3-540-30191-2_1.

[3] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. *Practical Group Signatures without Random Oracles*. Cryptology ePrint Archive, Report 2005/385. 2005.

[4] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. "A Practical and Provably Secure Coalition-Resistant Group Signature Scheme". In: *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*. 2000. URL: https://doi.org/10.1007/3-540-44598-6_16.

[5] Giuseppe Ateniese and Breno de Medeiros. "Efficient Group Signatures without Trapdoors". In: *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*. 2003. URL: https://doi.org/10.1007/978-3-540-40061-5_15.

[6] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. *Signatures with Flexible Public Key: A Unified Approach to Privacy-Preserving Signatures (Full Version)*. Cryptology ePrint Archive, Report 2018/191. https://eprint.iacr.org/2018/191. 2018.

[7] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. "Key-Privacy in Public-Key Encryption". In: *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*. 2001. URL: https://doi.org/10.1007/3-540-45682-1_33.

[8] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. "Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions". In: *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*. 2003. URL: https://doi.org/10.1007/3-540-39200-9_38.

[9] Mihir Bellare, Haixia Shi, and Chong Zhang. "Foundations of Group Signatures: The Case of Dynamic Groups". In: *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*. 2005. URL: https://doi.org/10.1007/978-3-540-30574-3_11.

[10] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. "Get Shorty via Group Signatures without Encryption". In: *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*. 2010. URL: https://doi.org/10.1007/978-3-642-15317-4_24.

[11] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*. 2004. URL: https://doi.org/10.1007/978-3-540-28628-8_3.

[12] Dan Boneh, Saba Eskandarian, and Ben Fisch. *Post-Quantum EPID Group Signatures from Symmetric Primitives*. Cryptology ePrint Archive, Report 2018/261. https://eprint.iacr.org/2018/261. 2018.

[13] Dan Boneh and Matthew K. Franklin. “Identity-Based Encryption from the Weil Pairing”. In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. 2001. URL: https://doi.org/10.1007/3-540-44647-8_13.

[14] Dan Boneh and Matthew K. Franklin. “Identity-Based Encryption from the Weil Pairing”. In: *SIAM J. Comput.* 32.3 (2003). URL: https://doi.org/10.1137/S0097539701398521.

[15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. “Foundations of Fully Dynamic Group Signatures”. In: *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*. 2016. URL: https://doi.org/10.1007/978-3-319-39555-5_7.

[16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. “Short Accountable Ring Signatures Based on DDH”. In: *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*. 2015. URL: https://doi.org/10.1007/978-3-319-24174-6_13.

[17] Xavier Boyen and Brent Waters. “Compact Group Signatures Without Random Oracles”. In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*. 2006. URL: https://doi.org/10.1007/11761679_26.

[18] Xavier Boyen and Brent Waters. “Full-Domain Subgroup Hiding and Constant-Size Group Signatures”. In: *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*. 2007. URL: https://doi.org/10.1007/978-3-540-71677-8_1.

[19] Jan Camenisch and Jens Groth. “Group Signatures: Better Efficiency and New Theoretical Aspects”. In: *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*. 2004. URL: https://doi.org/10.1007/978-3-540-30598-9_9.

[20] Jan Camenisch and Anna Lysyanskaya. “Signature Schemes and Anonymous Credentials from Bilinear Maps”. In: *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*. 2004. URL: https://doi.org/10.1007/978-3-540-28628-8_4.

[21] David Chaum and Eugène Van Heyst. “Group Signatures”. In: *EUROCRYPT’91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Springer, Heidelberg, 1991.

[22] Ivan Damgård and Jesper Buus Nielsen. “Improved Non-committing Encryption Schemes Based on a General Complexity Assumption”. In: *CRYPTO 2000*. 2000. URL: https://doi.org/10.1007/3-540-44598-6_27.

[23] David Derler and Daniel Slamanig. *Fully-Anonymous Short Dynamic Group Signatures Without Encryption*. Cryptology ePrint Archive, Report 2016/154. 2016.

[24] Alex Escala and Jens Groth. “Fine-Tuning Groth-Sahai Proofs”. In: *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Springer, 2014. URL: https://doi.org/10.1007/978-3-642-54631-0_36.

[25] Georg Fuchsbauer and Romain Gay. “Weakly Secure Equivalence-Class Signatures from Standard Assumptions”. In: *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770.

Lecture Notes in Computer Science. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-76581-5_6.

[26] Jun Furukawa and Shoko Yonezawa. "Group Signatures with Separate and Distributed Authorities". In: *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*. 2004. URL: https://doi.org/10.1007/978-3-540-30598-9_6.

[27] Taher El Gamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*. 1984. URL: https://doi.org/10.1007/3-540-39568-7_2.

[28] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. "Groth-Sahai Proofs Revisited". In: *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*. 2010. URL: https://doi.org/10.1007/978-3-642-13013-7_11.

[29] Jens Groth. "Fully Anonymous Group Signatures Without Random Oracles". In: *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*. 2007. URL: https://doi.org/10.1007/978-3-540-76900-2_10.

[30] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *EUROCRYPT 2008*. 2008. URL: https://doi.org/10.1007/978-3-540-78967-3_24.

[31] Christian Hanser and Daniel Slamanig. "Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials". In: *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*. 2014. URL: https://doi.org/10.1007/978-3-662-45611-8_26.

[32] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications". In: *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*. 2008. URL: https://doi.org/10.1007/978-3-540-85174-5_2.

[33] Aggelos Kiayias and Moti Yung. "Efficient Secure Group Signatures with Dynamic Joins and Keeping Anonymity Against Group Managers". In: *Progress in Cryptology - Mycrypt 2005, First International Conference on Cryptology in Malaysia, Kuala Lumpur, Malaysia, September 28-30, 2005, Proceedings*. 2005. URL: https://doi.org/10.1007/11554868_11.

[34] Aggelos Kiayias and Moti Yung. "Group Signatures with Efficient Concurrent Join". In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. 2005. URL: https://doi.org/10.1007/11426639_12.

[35] Aggelos Kiayias and Moti Yung. "Secure scalable group signature with dynamic joins and separable authorities". In: *IJSN* 1.1/2 (2006). URL: https://doi.org/10.1504/IJSN.2006.010821.

[36] Benoît Libert, Thomas Peters, and Moti Yung. "Group Signatures with Almost-for-Free Revocation". In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. 2012. URL: https://doi.org/10.1007/978-3-642-32009-5_34.

[37] Benoît Libert, Thomas Peters, and Moti Yung. "Scalable Group Signatures with Revocation". In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. 2012. URL: https://doi.org/10.1007/978-3-642-29011-4_36.

[38] Benoît Libert, Thomas Peters, and Moti Yung. "Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions". In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*. 2015. URL: https://doi.org/10.1007/978-3-662-48000-7_15.

[39] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. "Constant-Size Group Signatures from Lattices". In: *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. Lecture Notes in Computer Science. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-76581-5_3.

[40] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. "On the Security of Dynamic Group Signatures: Preventing Signature Hijacking". In: *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*. 2012. URL: https://doi.org/10.1007/978-3-642-30057-8_42.

[41] Gene Tsudik and Shouhuai Xu. "Accumulating Composites and Improved Group Signing". In: *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*. 2003. URL: https://doi.org/10.1007/978-3-540-40061-5_16.

[42] Brent Waters. "Efficient Identity-Based Encryption Without Random Oracles". In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. 2005. URL: https://doi.org/10.1007/11426639_7.