Membership Privacy for Fully Dynamic Group Signatures

Michael Backes
CISPA Helmholtz Center for Information Security
Saarland Informatics Campus

Lucjan Hanzlik
CISPA Helmholtz Center for Information Security
Saarland Informatics Campus
Stanford University

Jonas Schneider-Bensch
CISPA Helmholtz Center for Information Security
Saarland Informatics Campus

ABSTRACT

Group signatures present a compromise between the traditional goals of digital signatures and the need for signer 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. An important consideration that is absent in prevalent models is that group membership itself may be sensitive information, especially if group membership is dynamic, i.e. membership status may change over time.

We address this issue by introducing formal notions of membership privacy for fully dynamic group signature schemes, which can be easily integrated into the most expressive models of group signature security to date. We then propose a 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 (SPS-EQ).

Finally, we devise novel techniques for SFPK to construct a highly efficient standard model scheme (i.e. without random oracles) that provides shorter signatures than even the non-private state-of-the-art from standard assumptions. This shows that, although the strictly stronger security notions we introduce have been completely unexplored in the study of fully dynamic group signatures so far, they do not come at an additional cost in practice.

KEYWORDS

fully dynamic group signatures; membership privacy; signatures on equivalence classes

ACM Reference Format:
Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. 2019. Membership Privacy for Fully Dynamic Group Signatures. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3319535.3354257

1 INTRODUCTION

The concept of group signatures was introduced by Chaum and van Heyst in [24]. It allows a group manager to delegate signing rights to multiple signers. The group members may create publicly verifiable signatures on behalf of the entire group, such that the signature does not reveal the identity of the actual signer beyond their membership in the group. A designated opening authority has the ability to verifiably reveal the actual signer of a particular signature, in case of abuse. Ideal applications of group signatures are, for instance, business processes, where responsibility for certain actions should be shared among the members of one level of management by creating a signature, but accountability is preserved via the possibility of after-the-fact opening of a signature through a supervisory board.

In applications, which enforce *public* accountability of group members via the signature mechanism, it is essential that the public knows who is part of the group. We consider the idea that in some applications it is equally essential that group membership is not disclosed and cannot be associated with the user's global identity (e.g. represented by a public key), i.e. that the scheme provides an additional property called: *membership privacy*. In some cases leakage of membership information may have negative effects on the users. In particular, membership in controversial groups may lead to persecution, e.g. loss of job because of personal interests, beliefs or affiliation. Membership privacy is also advantageous if group signatures are used for access control to sensitive resources. Imagine, e.g., cities that provide free access to charging stations for electric cars to their inhabitants, but not to non-inhabitants. Group signatures offer a simple, privacy-preserving solution for access control in this case, since the charging station only has to verify that the user is an inhabitant of the city, represented by a group managed by the city government. However, using a non-private group signature scheme to implement this incurs a privacy leak, since cities would have to publish lists of their inhabitants and it could easily be tracked if and where people are moving between cities. In more involved scenarios, in the context of delegatable anonymous credentials [8, 23, 26], a membership private group signature scheme could be used to enforce confidentiality between members at different levels of delegation. What is more, since users can enroll the same public key in several groups—not all of which necessarily having the same privacy requirements—membership privacy reduces risk of identity disclosure by avoiding the overhead of managing a distinct pseudonymous identities for groups where privacy is required.[1]

1.1 Formal Models of Group Signatures

The first formal security model, called the *static model*, was given by Bellare, Micciancio and Warinschi (BMW) in [10], who also provided a construction from general assumptions.

CCS '19, November 11–15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6747-9/19/11…$15.00
https://doi.org/10.1145/3319535.3354257

[1] Provided users employ techniques such as anonymity networks to prevent disclosure on channels other than the group membership information.

Later models, notably the ones due to Bellare, Shi, and Zhang (BSZ) [11], as well as Kiayias, and Yung (KY) [40, 41], which will be subsumed under the term *(partially) dynamic models*, generalize the static model in terms of functionality as well as security considerations:

- The dynamic models split the group manager—previously entrusted with user key generation and opening—in separate issuing and opening authorities, allowing stronger security notions, which consider malicious behaviour on the part of either.
- The static model requires all potential group members to be known at setup time, the dynamic models allow dynamic enrollment after group creation via a join/issue protocol, where users jointly generate their signing keys in interaction with the authorities.

In the dynamic models, it has to be ensured that the opening is performed honestly. Otherwise a malicious opener or a malicious member of the group could produce a dishonest opening that identifies a wrong signer, either to claim a specific signatures for themselves, or blame a user for a signature which they did not create. Sakai et al. [47] thus define a notion called opening soundness, which, if achieved, ensures that it is infeasible to create an opening which points to any but the actual signer of a valid signature.

A further extension of the dynamic models was recently proposed by Bootle et al. [17]. Their model, which we subsequently call the *fully dynamic model*, additionally addresses revocation of group membership, incorporates opening soundness and considers security even under maliciously generated keys. To model the dynamic nature of addition and revocation of members, the scheme's lifetime is divided into a series of epochs such that changes in the group membership require advancing the scheme to the next epoch. Since the issuing authority decides who may join the group and who has to leave, the group's public information is updated by the issuing authority for each new epoch. The authors show that their model is general enough to capture previous notions, making it the most expressive model of the security of group signatures to date.

A related property to our membership privacy was conceived for the partially dynamic setting by Kiayias and Zhou in Hidden Identity-Based Signatures [42] and efficiently instantiated by Chow et al. [25]. In these works, group membership lists are avoided altogether, enabling to hide the identity of group members even from the opening authority. We stress that in the fully dynamic model some form group membership list is necessary to implement membership revocation, separating these approaches from ours.

Related Work. Generic constructions of group signatures from [10] and [11] established the widely used *sign-and-encrypt-and-prove* paradigm (SEP) design paradigm: A signature consists of an encryption under the opener's public key of both a members's signature on the message and the member's identity, as well as a non-interactive zero-knowledge proof that the encrypted identity—typically a signature issued by the group manager—is valid that of the signer of the message. Thus, relying on the unforgeability of this signature, such a group signature scheme achieves non-frameability and traceability. Abdalla and Warinschi proved in [1] that group signatures are actually equivalent to IND-CPA secure encryption schemes.

In [12], Bichsel et al. identify the SEP design paradigm as a source of inefficiency in group signatures. 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. Many group signature schemes were designed for both the static and dynamic case in the random oracle model which utilize the RSA cryptosystem [3, 21, 39, 48], discrete logarithm setting [4, 32], and bilinear setting [13, 22].

One of the first standard model constructions, i.e. without random oracles (cf. [45]), was introduced by Ateniese et al. [2] in the bilinear setting. The scheme is highly efficient, with signature consisting of only 8 group elements. However, it does not provide full-anonymity in the BMW model [10]. In particular, the adversary is not allowed to see the private keys of honest users.

Boyen and Waters [19, 20] proposed standard model schemes that use composite order bilinear groups, but in contrast to [2] allow key exposure attacks, albeit without allowing the adversary to 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 [36] allowed for the design of new and efficient group signature schemes in the standard model. Groth [35] was the first to introduce a standard model group signature with constant size public key and signatures, which preserve the full-anonymity property under a q-type assumption. The GS proof system was also used by Libert et al. [43, 44], who designed standard model group signatures with revocation capabilities.

At Crypto'15 Libert, Peters and Yung [45] introduced two efficient group signature schemes that rely on simple assumptions, the first scheme secure in the static BMW model [10], the second construction less efficient, but secure in the dynamic security model from [41].

Bootle et al. [18] 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. [17] 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 (SPS-EQ) [28]. SPS-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. There currently exists no standard model instantiation.

Recently, Backes et al. [5] introduced a new cryptographic primitive called signatures with flexible public key. The idea is similar to SPS-EQ, but instead of partitioning the message space, 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. The authors also show how to combine their primitive with SPS-EQ to construct static group signatures, which are secure in the BMW model [10].

Group signatures can also be constructed from lattice-based assumptions [46] or symmetric primitives [14]. 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 [11] and in the random oracle model. The latter scheme is also instantiated in the random oracle model.

1.2 Our Contribution in Detail

In this paper we revisit the fully-dynamic group signature framework by Bootle et al. [17]. We observe that the epoch information published with each modification of the group (joining or leaving of a member) may leak the identities of members. For instance in the scheme proposed in [18], where the epoch information contains a list of active members, this information is required to verify a signature. This is a major issue that limits the applications of group signatures and introduces real-world privacy risks that are not captured by the security model. In particular, let us consider the use of group signatures as part of a corporate/governmental access control system to resources. In such a scenario group signatures protect access patterns between mutually rival departments. On the other hand, leaking a list of active members of the group can be used potential adversaries to perform targetted attacks, e.g. bribery attempts, phishing attacks on private emails or denial of service attacks. An application that was impossible using previous definitions are private groups that can be used to create an electronic authentication method for private club members. Members of the club are unknown to the public and other members of the club but the group signatures allows a way to prove membership if required.

Therefore, as our first contribution, we propose a new security notion for fully dynamic group signatures, namely *membership privacy*. Informally, when a group signature scheme offers membership privacy it means that an external observer cannot tell who joined or left the group in a given epoch, even if a subset of the group's members is controlled by the observer.[2] The possibility of membership privacy changes the meaning of a group signature to the external public compared to the previous models. The public may still verify that the signature was created by a party which received signing capabilities from the issuing authority, but not only is there no indication who the signer was specifically, but even the group of potential signers is hidden. As a consequence, to an external observer, the group signature scheme is a way for the issuing authority to dynamically delegate signing capabilities to anonymous signers, who can be held *privately accountable* by the opening authority. In extending the model of Bootle et al. [17] we give formal definitions of join and leave privacy, which taken together constitute membership privacy in the most expressive model of group signature security to date.

Our second contribution is a generic construction of fully-dynamic group signatures with membership privacy. Our scheme is built upon novel techniques in the area of signatures with flexible public keys (SFPK) and their fruitful combination with signatures on equivalence classes (SPS-EQ). The former primitive allows signing keys to be re-randomized within a system of equivalence classes, while the second allows the same for messages and signatures. We build upon the idea, introduced in [5], of using the combination of SFPK and SPS-EQ schemes with *compatible* systems of equivalence classes to construct highly efficient privacy-preserving signature schemes. Each epoch the group manager uses a fresh instance of SPS-EQ to certify the public signing keys of members, which live in SFPK equivalence classes. However, instead of certifying the original keys in the epoch information, the group manager randomizes the public signing key and encrypts the randomization using the signer's public encryption key. 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 optimize our generic construction and efficiently instantiate it under standard assumptions without relying on the random oracle model. The resulting scheme has shorter signatures than state-of-the-art schemes [36, 45] that are secure in the same setting but only allow for partially-dynamic groups. To achieve this efficiency we introduce a new SFPK scheme that has an optimal public key size of 2 group elements in $\mathbb{G}_1$. The scheme is secure under the bilinear decisional Diffie-Hellman assumption and the decisional Diffie-Hellman assumption in $\mathbb{G}_1$. The technique that makes our instantiation of group signatures possible is a notion called *canonical representative*. Informally, we define a unique representative for every equivalence class, which allows us to identify the class without the use of any additional trapdoor. In the previous definition by Backes et al. [5] the only way to identify a class was by using a trapdoor created during key generation, which hinders all applications where keys have to be secret but identifying classes has to be done publicly. We summarize our results as follows:

(1) We extend the definitions of Bootle et al. [17] and show that membership privacy can be seamlessly integrated in the previous security models for fully dynamic group signatures.
(2) We devise a generic construction of fully dynamic group signatures with membership privacy that can be instantiated in the standard model.
(3) We devise a novel technique for the conjunction of SFPK and SPS-EQ, allowing us to build a highly efficient standard model group signature schemes along the lines of our generic construction, but with shorter signature size than even state-of-the-art non-private schemes with comparable assumption. This underlines that membership privacy need not come at additional cost.

[2] A similar property was recently put forward by Baldimtsi et al. [7] for the security of cryptographic accumulators, which are one of the building block of revocation systems for anonymous credentials. Although based on similar real-world concerns, their definition is specific to cryptographic accumulators and cannot be easily applied to group signatures.

2 PRELIMINARIES

Notation. We denote by $y \xleftarrow{s} \mathcal{A}(x)$ the execution of algorithm $\mathcal{A}$ on input x and with output y. By $r \xleftarrow{s} S$ we mean that r is chosen uniformly at random over the set S. We will use $1_{\mathbb{G}}$ to denote the identity element in group $\mathbb{G}$, $[n]$ to denote the set $\{1, \ldots, n\}$, and $\vec{u}$ to denote a vector. Finally, by $\mathcal{A}^{O}$ we denote an algorithm $\mathcal{A}$ that has access to oracle O. When the number of oracles is large we will also write $\mathcal{A}^{\{O_1, \ldots, O_n\}}$ to denote access to oracles $O_1, \ldots, O_n$.

We write $\mathsf{Exp}^{\phi}_{\mathcal{A},\Psi}(1^\lambda) \Rightarrow 1$ for the event that the experiment Exp returns 1, when instantiated with parameters ϕ, adversary $\mathcal{A}$ and primitive Ψ, all of which possibly omitted. We define the *adjusted advantage* of adversary $\mathcal{A}$ in this experiment as

$$\mathsf{Adv}\,[x]\ \mathsf{Exp}^{\phi}_{\mathcal{A},\Psi}(1^\lambda) := \left|\Pr\left[\mathsf{Exp}^{\phi}_{\mathcal{A},\Psi}(1^\lambda) \Rightarrow 1\right] - x\right|$$

If $x = 0$, we write instead $\mathsf{Adv}^{\mathsf{Exp}^{\phi}}_{\mathcal{A},\Psi}(1^\lambda)$ for its *advantage*.

2.1 Signatures on Equivalence Classes

We now recall the notion of signatures on equivalence classes (SPS-EQ) introduced by Hanser and Slamanig [37]. The signing algorithm $\mathsf{SPS.Sign}(\mathsf{sk}_{\mathsf{SPS}}, \vec{M})$ 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 $\vec{M}, \vec{M}' \in \mathbb{G}_i^{*\ell}$, we say that M and M' are from the same equivalence class (denoted by $[\vec{M}]_{\mathcal{R}}$) if there exists a scalar $r \in \mathbb{Z}_p^*$, such that for all $i \in [\ell]$ it holds $M_i^r = M_i'$. In terms of security, Hanser and Slamanig [37] define notions of unforgeability under chosen-message attacks and class-hiding. Fuchsbauer and Gay [31] introduce a relaxed notion, unforgeability under chosen-open-message attacks, which restricts the adversaries' signing queries to messages of which it knows all exponents.

Definition 2.1 (EUF-{CMA, CoMA}). For SPS-EQ scheme SPS = (BGGen, KGen, Sign, ChgRep, Verify, VKey) on $\mathbb{G}_i^{*\ell}$ we define the following experiments, parameterized in the given signing oracle:

$\mathsf{EUF}\text{-}t^{\ell}_{\mathcal{A},\mathsf{SPS}}$

$\mathsf{BG} \leftarrow \mathsf{SPS.BGGen}(\lambda);$
$(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) \xleftarrow{s} \mathsf{SPS.KGen}(\mathsf{BG}, \ell);$
$(\vec{M}^*, \sigma^*_{\mathsf{SPS}}) \xleftarrow{s} \mathcal{A}^{O_t(\mathsf{sk}_{\mathsf{SPS}}, \cdot)}(\mathsf{pk}_{\mathsf{SPS}})$
return $\forall M \in Q.\ [\vec{M}^*]_{\mathcal{R}} \neq [\vec{M}]_{\mathcal{R}} \quad \wedge$
$\mathsf{SPS.Verify}(\mathsf{pk}_{\mathsf{SPS}}, \vec{M}^*, \sigma^*_{\mathsf{SPS}}) = 1$

$O_{\mathsf{CMA}}(\mathsf{sk}_{\mathsf{SPS}}, \vec{M} \in \mathbb{G}_i^{*\ell})$

$\sigma_{\mathsf{SPS}} \leftarrow \mathsf{SPS.Sign}(\mathsf{sk}_{\mathsf{SPS}}, \vec{M})$
$Q := Q \cup \vec{M}$
return σ_{SPS}

$O_{\mathsf{CoMA}}(\mathsf{sk}_{\mathsf{SPS}}, \vec{e} \in \mathbb{Z}_p^{*\ell})$

$\vec{M} := (g^{e_1}, \ldots, g^{e_\ell})$
$\sigma_{\mathsf{SPS}} \leftarrow \mathsf{SPS.Sign}(\mathsf{sk}_{\mathsf{SPS}}, \vec{M})$
$Q := Q \cup \vec{M}$
return σ_{SPS}

A SPS-EQ is existentially unforgeable under chosen-message attacks / chosen-open-message attacks, if for all PPT algorithms $\mathcal{A}$, their advantage $\mathsf{Adv}^{\mathsf{EUF}-t^{\ell}}_{\mathcal{A},\mathsf{SPS}}(1^\lambda)$ is negligible where $t \in \{\mathsf{CMA}, \mathsf{CoMA}\}$.

Fuchsbauer and Gay also propose a strengthened class hiding notion, called perfect adaptation of signatures. Informally, this notion states that signatures received by changing the representative of the class and new signatures for the representative are identically distributed.

Definition 2.2 (Perfect Adaption of Signatures). A SPS-EQ scheme on $\mathbb{G}_i^{*\ell}$ perfectly adapts signatures if for all $(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}, \vec{M}, \sigma, r)$, where $\vec{M} \in \mathbb{G}_1^{*\ell}, r \in \mathbb{Z}_p^*$,

$$\mathsf{SPS.VKey}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 1, \text{ and } \mathsf{SPS.Verify}(\mathsf{pk}_{\mathsf{SPS}}, \vec{M}, \sigma) = 1,$$

the two distributions

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

are identical.

2.2 Signatures with Flexible Public Key

In our group signature construction we use a primitive called signatures with flexible public keys (SFPK), that was recently introduced by Backes et al. [5] and which we extend by the notion of *canonical representatives* (or *canonical form*).

In SFPK the public key space is partitioned into equivalence classes induced by a relation $\mathcal{R}$. A signer can efficiently generate $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{s} \mathsf{SFPK.KGen}(1^\lambda)$ and change her key pair to a different representative of the same class via two algorithms $\mathsf{pk}' \xleftarrow{s} \mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$ for the public key and $\mathsf{sk}' \xleftarrow{s} \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$ for the secret key, which take the same randomness $r \xleftarrow{s}$ coin. The randomized secret key can be used to sign a message $\mathsf{Sig} \xleftarrow{s} \mathsf{SFPK.Sign}(\mathsf{sk}', m)$, such that the signature can be verified by running $\mathsf{SFPK.Verify}(\mathsf{pk}', m, \mathsf{Sig})$. *Class-hiding* ensures that, without a trapdoor, it is hard to distinguish if two public keys are related, i.e. in the same equivalence class. If a key pair has been generated along a trapdoor δ using $(\mathsf{sk}, \mathsf{pk}, \delta) \xleftarrow{s} \mathsf{SFPK.TKGen}(1^\lambda)$, then given the trapdoor one can run $0/1 \leftarrow \mathsf{SFPK.ChkRep}(\delta, \mathsf{pk}')$ to check if pk' is in relation to pk. The original definition by Backes et al. uses a strong corruption model, where the adversary is given the random coins used to generate the challenged keys. We will show that in our construction a slightly weaker model, where the adversary gets only the secret keys, is sufficient. 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 CRS can be used to forge signatures. This kind of trapdoor δ_ρ was not specified in [5], 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 2.3 (Signature with Flexible Public Keys). A signature scheme with flexible public keys SFPK is a set of *PPT* algorithms such that:

- $\mathsf{SFPK.CRSGen}(1^\lambda)$: on input a security parameter 1^λ, outputs a trapdoor δ_ρ and a common reference string ρ, which is an implicit input for all the algorithms.
- $\mathsf{SFPK.KGen}(1^\lambda, \omega)$: on input a security parameter 1^λ and random coins $\omega \in \mathsf{coin}$, outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of secret and public keys.
- $\mathsf{SFPK.TKGen}(1^\lambda, \omega)$: on input a security parameter 1^λ and random coins $\omega \in \mathsf{coin}$, outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of secret and public keys, and a trapdoor δ.
- $\mathsf{SFPK.Sign}(\mathsf{sk}, m)$: on input a message $m \in \{0,1\}^*$ and a signing key sk, outputs a signature Sig.
- $\mathsf{SFPK.ChkRep}(\delta, \mathsf{pk}')$: on input a trapdoor δ for some equivalence class $[\mathsf{pk}]_\mathcal{R}$ and public key pk', outputs 1 if $\mathsf{pk}' \in [\mathsf{pk}]_\mathcal{R}$ and 0 otherwise.
- $\mathsf{SFPK.ChgPK}(\mathsf{pk}, r)$: on input a representative pk of equivalence class $[\mathsf{pk}]_\mathcal{R}$ and random coins r, outputs 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, outputs an updated secret key sk'.
- $\mathsf{SFPK.Verify}(\mathsf{pk}, m, \mathsf{Sig})$: on input a message m, signature Sig and public verification key pk, outputs 1 if the signature is valid and 0 otherwise.

To simplify notation we write $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{SFPK.ChgKeys}(\mathsf{sk}, \mathsf{pk}, r)$ as shorthand for the joint randomization of secret and public keys using the same r.

Definition 2.4 (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 key pairs output by SFPK.KGen and SFPK.TKGen are identically distributed.
(2) For all key pairs $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{SFPK.KGen}(1^\lambda, \omega)$ 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 $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{SFPK.ChgKeys}(\mathsf{sk}, \mathsf{pk}, r)$.
(3) For all $(\mathsf{sk}, \mathsf{pk}, \delta) \xleftarrow{\$} \mathsf{SFPK.TKGen}(1^\lambda, \omega)$ and all pk' we have $\mathsf{SFPK.ChkRep}(\delta, \mathsf{pk}') = 1$ if and only if $\mathsf{pk}' \in [\mathsf{pk}]_\mathcal{R}$.

Definition 2.5 (Class-hiding with Key Corruption). For scheme SFPK with relation $\mathcal{R}$ and adversary $\mathcal{A}$ we define the following experiment:

$\underline{\mathsf{C\text{-}H}^\mathcal{R}_{\mathcal{A},\mathsf{SFPK}}(\lambda)}$

$\omega_0, \omega_1 \xleftarrow{\$} \mathsf{coin}$

$(\mathsf{sk}_i, \mathsf{pk}_i) \xleftarrow{\$} \mathsf{SFPK.KGen}(1^\lambda, \omega_i)$ for $i \in \{0,1\}$

$b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \mathsf{coin}$

$(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{SFPK.ChgKeys}(\mathsf{sk}_b, \mathsf{pk}_b, r)$

$\hat{b} \xleftarrow{\$} \mathcal{A}^{\mathsf{SFPK.Sign}(\mathsf{sk}', \cdot)}((\mathsf{sk}_0, \mathsf{pk}_0), (\mathsf{sk}_1, \mathsf{pk}_1), \mathsf{pk}')$

return $b = \hat{b}$

A SFPK is *class-hiding with key corruption* if for all *PPT* adversaries $\mathcal{A}$, their adjusted advantage $\mathsf{Adv}\left[\frac{1}{2}\right]^{\mathsf{C\text{-}H}}_{\mathcal{A},\mathsf{SFPK}}(1^\lambda)$ is negligible.

Definition 2.6 (Strong Existential Unforgeability under Flexible Public Key). For scheme SFPK with relation $\mathcal{R}$ and adversary $\mathcal{A}$ we define the following experiment:

$\underline{\mathsf{sEUF-CMA}^\mathcal{R}_{\mathcal{A},\mathsf{SFPK}}(\lambda)}$

$\omega \xleftarrow{\$} \mathsf{coin}$

$(\mathsf{sk}, \mathsf{pk}, \delta) \xleftarrow{\$} \mathsf{SFPK.TKGen}(1^\lambda, \omega); Q := \emptyset$

$(\mathsf{pk}', m^*, \mathsf{Sig}^*) \xleftarrow{\$} \mathcal{A}^{O_1(\mathsf{sk}, \cdot), O_2(\mathsf{sk}, \cdot, \cdot)}(\mathsf{pk}, \delta)$

return $(m^*, \mathsf{Sig}^*) \notin Q \wedge$
$\mathsf{SFPK.ChkRep}(\delta, \mathsf{pk}') = 1 \wedge$
$\mathsf{SFPK.Verify}(\mathsf{pk}', m^*, \mathsf{Sig}^*) = 1$

$\underline{O_1(\mathsf{sk}, m)}$

$\mathsf{Sig} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk}, m)$

$Q := Q \cup \{(m, \mathsf{Sig})\}$

return Sig

$\underline{O_2(\mathsf{sk}, m, r)}$

$\mathsf{sk}' \xleftarrow{\$} \mathsf{SFPK.ChgSK}(\mathsf{sk}, r)$

$\mathsf{Sig} \xleftarrow{\$} \mathsf{SFPK.Sign}(\mathsf{sk}', m)$

$Q := Q \cup \{(m, \mathsf{Sig})\}$

return Sig

A SFPK is *existentially unforgeable with flexible public key under chosen message attacks* if for all *PPT* adversaries $\mathcal{A}$, their advantage $\mathsf{Adv}^{\mathsf{sEUF-CMA}^\mathcal{R}}_{\mathcal{A},\mathsf{SFPK}}(1^\lambda)$ is negligible.

Canonical Representatives. It might be the case that every public key equivalence class has a unique representative which can act as a description of the class. We will call such objects the *canonical representatives* of the given classes and further assume that if a scheme has canonical representatives, there is an efficient predicate IsCanonical which on input a public key will return 1 if and only if the public key is canonical. We will use this type of public key and representative later in the optimized variant of our scheme.

*Definition 2.7 (*SPS-EQ*/*SFPK *Compatibility).* An SPS-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.

2.3 Additional Preliminaries

In the following sections, we make use of a number of well-known cryptographic primitives, including programmable hash functions, digital signature schemes, key-private public key encryption, as well as efficient non-interactive proof systems. The instantiations of our constructions are in the bilinear setting, i.e. in the presence of a bilinear group generation algorithm BG and we prove security under the standard decisional Diffie-Hellman and and bilinear decisional Diffie-Hellman assumptions, relative to BG respectively.

Definition 2.8 (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 2.9 (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.

2.4 Assumptions

Definition 2.10 (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 $\mathsf{Adv}^{\mathrm{ddh}}_{\mathcal{A}}(\lambda)$ to denote the advantage of the adversary in solving this problem.

If the 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 $e(g_1^a, g_2^b) = e(g_1^z, g_2)$. 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 [15] who defined a similar problem called Weil decisional Diffie-Hellman problem in the type 1 setting. In their later work [16] it was renamed to bilinear decisional Diffie-Hellman assumption. We restate it for type 3 pairings as follows:

Definition 2.11 (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 $\mathsf{Adv}^{\mathrm{bddh}}_{\mathcal{A}}(\lambda)$ to denote the advantage of the adversary in solving this problem.

Definition 2.12 (Collision-Resistance). We call a hash 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 $\mathsf{Adv}^{\mathrm{coll}}_{\mathcal{A}}(\lambda)$ to denote the advantage of the adversary in finding a collision for this hash function.

2.5 Programmable Hash Functions

We now recall the definition of programmable hash functions introduced by Hofheinz and Kiltz [38]. We first define a *group hash function* for group $\mathbb{G}$ and output length $\ell = \ell(\lambda)$ as consisting of two polynomial time algorithms PHF.Gen and 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 to deterministically evaluate the hash function via $y \in \mathbb{G} \leftarrow \mathsf{PHF.Eval}(K_{\mathsf{PHF}}, X)$, where $X \in \{0,1\}^\ell$.

Definition 2.13. A group hash function (PHF.Gen, PHF.Eval) is an (m, n, γ, δ)-programmable hash function if there are polynomial time algorithms PHF.TrapGen and PHF.TrapEval such that:

- For any $g, h \in \mathbb{G}$ the $(K'_{\mathsf{PHF}}, td) \xleftarrow{\$} \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ outputs a key K' and trapdoor td. Then, 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

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

  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 [49] 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.

2.6 Non-Interactive Proof Systems

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. $L_{\mathcal{R}} = \{x \mid \exists w : (x, w) \in \mathcal{R}\}$.

Definition 2.14 (Non-Interactive Proof System). A non-interactive proof system Π consists of the following three algorithms:

- $\mathsf{Setup}(1^\lambda)$**:** on input security parameter 1^λ, outputs a common reference string ρ.
- $\mathsf{Prove}(\rho, x, w)$**:** on input common reference string ρ, statement x and witness w, outputs a proof π.
- $\mathsf{Verify}(\rho, x, \pi)$**:** on input common reference string ρ, statement x and proof π, outputs either accept(1) or 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 2.15 (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:

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

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 2.16 (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:

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

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 2.17 (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:

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

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 $(x, w) \notin \mathcal{R}$.

Definition 2.18 (Simulation Sound Extractability). A proof system Π is called simulation sound, if there exists a knowledge extractor

E = (ExtGen, Extract) and simulator S = Sim, such that for all algorithms $\mathcal{A}$

$$\mathsf{Adv}^{\mathsf{sse}}_{\Pi,\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{l}(\rho,\tau,\psi)\xleftarrow{\$}\mathsf{ExtGen}(1^\lambda);\\ (x,\pi)\xleftarrow{\$}\mathcal{A}^{\mathsf{Sim}(\rho,\tau,\cdot)}(\rho,\psi);\\ w\xleftarrow{\$}\mathsf{Extract}(\rho,\psi,x,\pi)\end{array} : \begin{array}{c}(x,\pi)\notin Q \ \wedge \ (x,w)\notin R\\ \wedge \ \mathsf{Verify}(\rho,x,\pi)=1\end{array}\right]$$

is negligible in λ, where Q is a list of simulation queries and responses (x_i,π_i) of Sim and (ρ,τ) is identical to the output of SimGen from the definition of zero-knowledge.

2.7 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 (DS.KGen, DS.Sign, 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 existential unforgeability under chosen message attacks of the signature scheme.

A public key encryption scheme PKE consists of three algorithms (PKE.KeyGen, PKE.Enc, PKE.Dec). We use the standard notion of indistinguishability of ciphertexts under chosen message attacks (IND − CPA) as well as the notion of key privacy under chosen message attacks (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 [9]. An example of a scheme which achieves key privacy is the El Gamal encryption scheme [33]. Finally, we will use $\rightleftharpoons$ to denote the relation between the secret key and the corresponding public key. Note that for many schemes, like e.g. El Gamal, this relation can be easily checked and proven.

3 FULLY DYNAMIC GROUP SIGNATURES

We recall the framework of definitions for fully dynamic group signatures established in [17].

Definition 3.1. A fully dynamic group signature (FDGS) scheme GS is 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 $\vec{reg}$.
- $\langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param})\rangle$**:** 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 gpk := (param, mpk, tpk).
- $\mathsf{GS.KGen}_{\mathcal{U}}(\mathsf{param})$**:** On input the public parameters, the user key generation algorithm outputs a pair of user secret and user public key $\left(\vec{usk}[\mathsf{uid}], \vec{upk}[\mathsf{uid}]\right)$, bound to a fresh user id uid.
- $\langle \mathsf{GS.Join}(\mathsf{info}_\tau, \mathsf{gpk}, \mathsf{uid}, \vec{usk}[\mathsf{uid}]),$ $\mathsf{GS.Issue}(\mathsf{info}_\tau, \mathsf{msk}, \mathsf{uid}, \vec{upk}[\mathsf{uid}])\rangle$**:** A user who has executed $\mathsf{GS.KGen}_{\mathcal{U}}$, obtaining a user id uid and key pair $\left(\vec{usk}[\mathsf{uid}], \vec{upk}[\mathsf{uid}]\right)$ may, given the group public key and information regarding the current epoch info_τ engage the group manager in a join-issue procedure to become a member of the group. If successful, the output of the GS.Issue algorithm is user registration information which is stored in $\vec{reg}[\mathsf{uid}]$; the user group signing key $\vec{gsk}[\mathsf{uid}]$ is updated with the output of GS.Join.
- $\mathsf{GS.RevokeMember}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{current}}, \mathcal{S}, \vec{reg})$**:** 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 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 output is the new group information $\mathsf{info}_{\tau_{new}}$ and a possibly updated registration table $\vec{reg}$. If the group information does not change, the algorithm outputs $\perp$.
- $\mathsf{GS.Sig}(\mathsf{gpk}, \vec{gsk}[\mathsf{uid}], \mathsf{info}_\tau, m)$**:** Given their group signing key, current group information and the group public key, a user may sign a message, producing a signature Σ. If 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 output accept, otherwise reject.
- $\mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \vec{reg}, m, \Sigma)$**:** Given a signature, message, group information for epoch τ and a registration table, the tracing manager may output a pair (uid, π) where 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.
- $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \pi_{\mathbf{Trace}}, \vec{upk}[\mathsf{uid}], m, \Sigma)$**:** 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 $\vec{gsk}[\mathsf{uid}]$, in which case the algorithm outputs accept, otherwise reject.

3.1 Security Definitions

We recall from [17] the correctness and security definitions for FDGS (cf. Figure 1).

Correctness. A malicious user may not undermine the correctness of the scheme. This includes unforgeability of the scheme.

Definition 3.2. A FDGS scheme GS is *correct* if for all *PPT* adversaries $\mathcal{A}$, their advantage $\mathsf{Adv}^{\mathsf{Correctness}}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$ is negligible.

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

Definition 3.3. A FDGS scheme GS achieves *anonymity* if for all *PPT* adversaries $\mathcal{A}$, their adjusted advantage $\mathsf{Adv}\left[\frac{1}{2}\right]^{\mathsf{Anonymity}^b}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$ over the additional uniform choice of $b \in \{0,1\}$ is negligible.

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

param $\xleftarrow{s}$ GS.Setup(1^λ); $\mathcal{H} := \emptyset$

(msk, mpk, info, tsk, tpk) $\xleftarrow{s}$ ⟨GS.KGen$_{\mathcal{M}}$(param), GS.KGen$_{\mathcal{T}}$(param)⟩

gpk := (param, mpk, tpk)

(uid, m, τ) $\xleftarrow{s}$ $\mathcal{A}^{\text{AddU, ReadReg, Revoke}}$(gpk, info)

if uid $\notin \mathcal{H}$ or $\vec{gsk}$[uid] = $\perp$ or info$_\tau$ = $\perp$

or GS.IsActive(info$_\tau$, $\vec{reg}$, uid) = 0

then return 0

$\Sigma \xleftarrow{s}$ GS.Sig(gpk, $\vec{gsk}$[uid], info$_\tau$, m)

if GS.Vf(gpk, info$_\tau$, m, Σ) = reject

then return 1

(uid*, π) $\xleftarrow{s}$ GS.Trace(gpk, tsk, info$_\tau$, $\vec{reg}$, m, Σ)

if uid $\neq$ uid* **then return** 1

if GS.Judge(gpk, uid, info$_\tau$, π, $\vec{upk}[uid]$, m, Σ) = 0

then return 0 **else return** 1

$\mathsf{Non-Frame}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$

param $\xleftarrow{s}$ GS.Setup(1^λ); $\mathcal{H}$, $\mathcal{C}$, $\mathcal{B}$, $\mathcal{Q} := \emptyset$

(st, info, msk, mpk, tsk, tpk) $\xleftarrow{s}$ $\mathcal{A}$(init : param)

if msk = $\perp$ or mpk = $\perp$

then return 0

gpk := (param, mpk, tpk)

$(m, \Sigma, \text{uid}, \pi, \text{info}_\tau) \xleftarrow{s} \mathcal{A}\left\{\begin{matrix}\text{CrptU,}\\ \text{SndToU,}\\ \text{ModifyReg}\end{matrix}\right.\left.\begin{matrix}\\ \text{RevealU,}\\ \end{matrix}\right.\left.\begin{matrix}\text{Sign,}\\ \\ \end{matrix}\right\}$(play : st, gpk)

if GS.Vf(gpk, info$_\tau$, m, Σ) = 0

or GS.Judge(gpk, uid, info$_\tau$, π, $\vec{upk}$[uid], m, Σ) = 0

then return 0

if uid $\in \mathcal{H} \setminus \mathcal{B}$ and (uid, m, Σ, τ) $\notin \mathcal{Q}$

then return 1 **else return** 0

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

param $\xleftarrow{s}$ GS.Setup(1^λ); $\mathcal{H}$, $\mathcal{C}$, $\mathcal{B}$, $\mathcal{Q} := \emptyset$

(st, tsk, tpk) $\xleftarrow{s}$ $\mathcal{A}^{\langle \text{GS.KGen}_{\mathcal{M}}(\text{param}), \cdot \rangle}$(init : param)

if $\perp \leftarrow$ GS.KGen$_{\mathcal{M}}$(param) or $\mathcal{A}$'s output invalid

then return 0

(msk, mpk, info) $\leftarrow$ GS.KGen$_{\mathcal{M}}$(param); gpk := (param, mpk, tpk)

$(m, \Sigma, \tau) \xleftarrow{s} \mathcal{A}\left\{\begin{matrix}\text{AddU,}\\ \text{SndToM,}\\ \text{Sign,}\\ \text{Revoke}\end{matrix}\right.\left.\begin{matrix}\\ \\ \\ \end{matrix}\right.\left.\begin{matrix}\text{CrptU,}\\ \text{RevealU,}\\ \text{ModifyReg,}\\ \end{matrix}\right\}$(play :, st, gpk, info)

if GS.Vf(gpk, info$_\tau$, m, Σ) = reject

then return 0

(uid, π) $\xleftarrow{s}$ GS.Trace(gpk, tsk, info$_\tau$, $\vec{reg}$, m, Σ)

if GS.IsActive(info$_\tau$, $\vec{reg}$, uid) = 0 or uid = 0

or GS.Judge(gpk, uid, info$_\tau$, π, $\vec{upk}$[uid], m, Σ) = 0

then return 1 **else return** 0

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

param $\xleftarrow{s}$ GS.Setup(1^λ); $\mathcal{H}$, $\mathcal{C}$, $\mathcal{B}$, $\mathcal{Q}$, $\mathcal{Q}^* := \emptyset$

(st, msk, mpk, info) $\xleftarrow{s}$ $\mathcal{A}^{\langle \cdot, \text{GS.KGen}_{\mathcal{T}}(\text{param}) \rangle}$(init : param)

if $\perp \leftarrow$ GS.KGen$_{\mathcal{T}}$(param) or $\mathcal{A}$'s output invalid

then return 0

(tsk, tpk) $\leftarrow$ GS.KGen$_{\mathcal{T}}$(param); gpk := (param, mpk, tpk)

$d \xleftarrow{s} \mathcal{A}\left\{\begin{matrix}\text{AddU,}\\ \text{SndToU,}\\ \text{Trace,}\\ \text{Chall}_b\end{matrix}\right.\left.\begin{matrix}\\ \\ \\ \end{matrix}\right.\left.\begin{matrix}\text{CrptU,}\\ \text{RevealU,}\\ \text{ModifyReg,}\\ \end{matrix}\right\}$(play : st, gpk)

return $d == b$

Figure 1: Security experiments for fully dynamic group signatures, excluding Tracing Soundness.

Traceability. No coalition of group members and the opening authority can produce a signature which opens to an invalid identity or an identity that was not active in the signing epoch.

Definition 3.4. A FDGS scheme GS achieves *traceability* if for all *PPT* adversaries $\mathcal{A}$, their advantage $\mathsf{Adv}^{\mathsf{Traceability}}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$ is negligible.

Non-Frameability. No coalition of malicious group members and the issuing and opening authorities can produce a signature which opens to an honest user identity.

Definition 3.5. A FDGS scheme GS achieves *non-frameability* if for all *PPT* adversaries $\mathcal{A}$, their advantage $\mathsf{Adv}^{\mathsf{Non-Frame}}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$ is negligible.

Functional Tracing Soundness. A further property defined in [17] is *tracing* or *opening soundness*: Even if all parties in the group collude, they cannot produce a valid signature that traces to two different members.

A subtle point arises in the definition of tracing soundness, namely how is the uniqueness of group members established? If the adversary controls several users, they may share the same public key, hence their signatures cannot be distinguished by an opening which reveals the public key of the signer. Because of this, the opening instead leads to a specific user identity, i.e. an entry in the public registration table. This has two-fold consequences: 1) The user registration table has to be public, otherwise the opening is meaningless. 2) To verify an opening or a signature, it has to be verified as well that the group at the time of the creation of the signature was well-formed, i.e. every member occupies exactly one slot in the registration table.

We propose a relaxation of this notion, which allows us to avoid these implications. Our notion, *functional tracing soundness* distinguishes members by their public keys, i.e. it should not be possible, even in a fully corrupted group to create a valid signature and two openings for it which indicate conflicting public keys. The modifications to the tracing soundness experiment which implement this change are highlighted.

We observe that the FDGS scheme based on accountable ring signatures presented in [17] adheres to this definition already, 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.

Note that the construction of FDGS presented later in this work can be made to achieve the original version of tracing soundness,

albeit at the cost of the above mentioned group integrity checks and any kind of group membership privacy.

Definition 3.6 (Functional Tracing Soundness). For a FDGS scheme GS we define the following experiment:

$\mathsf{Functional-Trace-Sound}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$

$\mathsf{param} \xleftarrow{s} \mathsf{GS.Setup}(1^\lambda); C := \emptyset$

$(\mathsf{st}, \mathsf{info}, \mathsf{msk}, \mathsf{mpk}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{s} \mathcal{A}(\mathsf{init} : \mathsf{param})$

if $\mathsf{msk} = \bot$ or $\mathsf{mpk} = \bot$

  then return 0

$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$

$(m, \Sigma, \{\mathsf{uid}_i, \pi_i\}_{i=1}^2, \mathsf{info}_\tau) \xleftarrow{s} \mathcal{A}^{\mathsf{CrptU},\mathsf{ModifyReg}}(\mathsf{play} : \mathsf{st}, \mathsf{gpk})$

if $\mathsf{GS.Vf}(\mathsf{gpk}, \mathsf{info}_\tau, m, \Sigma) = 0$

  then return 0

if $\vec{upk}[\mathsf{uid}_1] = \vec{upk}[\mathsf{uid}_2]$ or $\exists i \in \{1,2\}$ s.t. $\vec{upk}[\mathsf{uid}_i] = \bot$

or $\mathsf{GS.Judge}(\mathsf{gpk}, \mathsf{uid}_i, \mathsf{info}_\tau, \pi_i, \vec{upk}[\mathsf{uid}_i], m, \Sigma) = 0$

  then return 0 **else return** 1

A FDGS scheme GS achieves *functional tracing soundness* if for all *PPT* adversaries $\mathcal{A}$, their advantage

$$\mathsf{Adv}^{\mathsf{Functional-Trace-Sound}}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$$

is negligible.

3.1.1 Experiment State and Oracle Intuition. The experiments may be stateful and keep lists of the attackers' actions to subsequently determine whether the attacker was successful or not. These lists are the following:

- $\mathcal{H}$: Honest users added via AddU.
- C: Users with maliciously generated keys, added via CrptU.
- $\mathcal{B}$: Users whose secret keys were revealed to the adversary via RevealU.
- Q: Signature Queries, populated by Sign.
- Q^*: Signatures created by the challenge users, populated by Chall.

The formal description of the given oracles can be found in the full version of this paper. Informally, they serve the following functions:

- AddU(uid): If uid is new to the system, run $\mathsf{GS.KGen}_{\mathcal{U}}(1^\lambda)$ to honestly generate the user's keys $(\vec{usk}[\mathsf{uid}], \vec{upk}[\mathsf{uid}])$ and add uid to $\mathcal{H}$. Afterwards the honest key generation is run using GS.Join and GS.Issue. This determines the user group secret key $\vec{gsk}[\mathsf{uid}]$ and the contents of the registration table $\vec{reg}[\mathsf{uid}]$. Return new epoch information info_τ and the user's public key $\vec{upk}[\mathsf{uid}]$.
- CrptU(uid, pk): If uid is new to the system, set $\vec{upk}[\mathsf{uid}]$ to the supplied key pk and add uid to C. Initiates a join/issue session for uid.
- SndToM(uid, M_{in}): Advance a currently running join/issue session for corrupted user uid by running the group manager side of the session with adversary provided input M_{in}. If the session concludes successfully, the challenger updates $\vec{reg}[\mathsf{uid}]$ with the final group manager session state. Return the group manager response M_{out}.
- SndToU(uid, M_{in}): Advance or initiate a join/issue session for user uid by running the user side of the session with the adversary provided input M_{in}. If the session concludes successfully, the challenger updates $\vec{gsk}[\mathsf{uid}] := \mathsf{st}^{\mathsf{uid}}_{\mathsf{GS.Join}}$ accordingly with the final user session state. Return the user response M_{out}.
- ReadReg(uid): Return registration table entry $\vec{reg}[\mathsf{uid}]$.
- ModifyReg(uid, *val*): Set entry $\vec{reg}[\mathsf{uid}] := val$.
- RevealU(uid): Return the user secret keys $(\vec{usk}[\mathsf{uid}], \vec{gsk}[uid])$ add uid to the set of bad users $\mathcal{B}$.
- Sign(uid, m, τ): If τ is a valid epoch, where uid is active, create a signature $\Sigma \xleftarrow{s} \mathsf{GS.Sig}(\mathsf{gpk}, \vec{gsk}[uid], \mathsf{info}_\tau, m)$ and add $(\mathsf{uid}, m, \Sigma, \tau)$ to the set of queried signatures Q. Return Σ.
- Trace($m, \Sigma, \mathsf{info}_\tau$): If Σ is valid in epoch τ and is not part of the challenge set Q^*, return $\mathsf{GS.Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \vec{reg}, m, \Sigma)$.
- Revoke($\mathcal{S}$): Return $\mathsf{GS.RevokeMember}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_\tau, \mathcal{S}, \vec{reg})$.
- $\mathsf{Chall}_b(\mathsf{info}_\tau, \mathsf{uid}_0, \mathsf{uid}_1, m)$: If uid_0 and uid_1 are both active and honest in τ, run $\mathsf{GS.Sig}(\mathsf{gpk}, \vec{gsk}[\mathsf{uid}_b], \mathsf{info}_\tau, m)$ to obtain signature Σ, adding (m, Σ, τ) to the challenge signature set Q^* and returning the signature.

In addition, the challenger keeps track of the active members of the group. We assume it has access to an algorithm GS.IsActive as follows.

- $\mathsf{GS.IsActive}(\mathsf{info}_\tau, \vec{reg}, \mathsf{uid})$: Given a group information for epoch τ, a registration table $\vec{reg}$ and a user-ID uid, outputs 1 if *uid* is a non-revoked member of the group in that epoch, 0 otherwise.

3.2 Leave-Join Privacy for FDGS

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 [24], 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 fully dynamic group signatures. Informally, we will say that a group signature scheme has membership privacy if it protects 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 [17]. However, one can easily specify similar experiments

$\mathsf{Join-Privacy}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$

$\mathsf{param} \xleftarrow{s} \mathsf{GS.Setup}(1^\lambda)$

$(\mathsf{msk}, \mathsf{mpk}, \mathsf{info}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{s} \langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \rangle$

$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$

$(\mathsf{st}, \mathsf{uid}_0, \mathsf{uid}_1) \xleftarrow{s} \mathcal{A}_0 \left\{ \begin{matrix} \mathsf{AddU}, \mathsf{RevealU}, \mathsf{CrptU}, \mathsf{SndToM}, \\ \mathsf{Sign}, \mathsf{Trace}, \mathsf{Revoke} \end{matrix} \right\} (\mathsf{gpk}, \mathsf{info})$

if $\{\mathsf{uid}_0, \mathsf{uid}_1\} \cap C \neq \emptyset$ **then return** 0

$b \xleftarrow{s} \{0,1\}$; $(\mathsf{info}^*, \vec{upk}[\mathsf{uid}_b]) \xleftarrow{s} \mathsf{AddU}(\mathsf{uid}_b)$;

$(\vec{usk}[\mathsf{uid}_{1-b}], \vec{upk}[\mathsf{uid}_{1-b}]) \xleftarrow{s} \mathsf{GS.KGen}_{\mathcal{U}}(1^\lambda)$

$\tau^* := \tau_{current}$; $\mathcal{H}^* := \{\mathsf{uid}_0, \mathsf{uid}_1\}$

$d \xleftarrow{s} \mathcal{A}_1 \left\{ \begin{matrix} \mathsf{AddU}, \mathsf{RevealU}, \mathsf{CrptU}, \mathsf{SndToM}, \\ \mathsf{Sign}, \mathsf{Trace}, \mathsf{Revoke} \end{matrix} \right\} (\mathsf{st}, \mathsf{info}^*, \vec{upk}[\mathsf{uid}_0], \vec{upk}[\mathsf{uid}_1])$

return $b = d$

$\mathsf{Leave-Privacy}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$

$\mathsf{param} \xleftarrow{s} \mathsf{GS.Setup}(1^\lambda)$

$(\mathsf{msk}, \mathsf{mpk}, \mathsf{info}, \mathsf{tsk}, \mathsf{tpk}) \xleftarrow{s} \langle \mathsf{GS.KGen}_{\mathcal{M}}(\mathsf{param}), \mathsf{GS.KGen}_{\mathcal{T}}(\mathsf{param}) \rangle$

$\mathsf{gpk} := (\mathsf{param}, \mathsf{mpk}, \mathsf{tpk})$

$(\mathsf{st}, \mathsf{uid}_0, \mathsf{uid}_1) \xleftarrow{s} \mathcal{A}_0 \left\{ \begin{matrix} \mathsf{AddU}, \mathsf{RevealU}, \\ \mathsf{Revoke}, \mathsf{Sign}, \mathsf{Trace} \end{matrix} \right\} (\mathsf{gpk}, \mathsf{info})$

if $\{\mathsf{uid}_0, \mathsf{uid}_1\} \cap \mathcal{H} \setminus (C \cup \mathcal{B}) \neq \{\mathsf{uid}_0, \mathsf{uid}_1\}$ **then return** 0

$b \xleftarrow{s} \{0,1\}; \mathcal{H}^* := \{\mathsf{uid}_0, \mathsf{uid}_1\}; \mathsf{dec}_{\mathsf{inv}} := \mathbf{true}; \tau^* := \tau_{current}$

$\mathsf{info}^* \xleftarrow{s} \mathsf{GS.RevokeMember}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau^*}, \{\mathsf{uid}_b\}, \vec{reg})$

$d \xleftarrow{s} \mathcal{A}_1 \left\{ \begin{matrix} \mathsf{AddU}, \mathsf{RevealU}, \\ \mathsf{Revoke}, \mathsf{Sign}, \mathsf{Trace} \end{matrix} \right\} (\mathsf{st}, \mathsf{info}^*)$

return $b = d$

Figure 2: Security experiments for Join- and Leave-Privacy.

for the partially dynamic models [11, 40, 41]. 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.[3] 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 $\vec{reg}$ 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-run algorithms and it is easier to keep it local to the authorities than publishing it online. An exception is the scheme [47] mentioned above, where the registration table is part of the verification algorithm to ensure that tracing soundness holds with respect to public user identities rather than in the functional sense we describe.

We formally define join and leave privacy in terms of the two experiments shown in Figure 2. Note, that we introduce a new set of privacy challenge users $\mathcal{H}^*$. In the two experiments, $\mathcal{H}^*$ is used to restrict the function of oracles which would allow trivial success for the adversary:

- The privacy challenge users may not be removed from the group, i.e. Revoke returns $\perp$ if $\mathcal{S} \cap \mathcal{H}^* \neq \emptyset$. This is because GS.RevokeMember is defined to return $\perp$ if the group information does not change as result of the revocation, which would be the case if the user was already removed from the group.
- The privacy challenge users may not be corrupted or have their keys revealed. Note, that this also prevents an adversary from re-enrolling a challenge user by initiating a join-issue session for them.
- The signing oracle treats signature requests for user IDs in the privacy challenge set differently. In the case of join privacy, a signature request for any privacy challenge user, i.e. uid_0 or uid_1 will be treated like a signature request for user uid_b who joined the system. In the case of leave privacy, it will be treated like a signature request for user $\mathsf{uid}_{(1-b)}$ who did not leave the group. Additionally, the queries will be added to the set of challenge queries Q^*, which prevents the adversary from using the Trace oracle to produce an opening for them.

Definition 3.7. A FDGS scheme GS has *join privacy* if for all *PPT* adversaries $\mathcal{A}$, their adjusted advantage

$$\mathsf{Adv}\left[\frac{1}{2}\right]^{\mathsf{Join-Privacy}}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$$

is negligible.

Definition 3.8. A FDGS scheme GS has *leave privacy* if for all *PPT* adversaries $\mathcal{A}$, their adjusted advantage

$$\mathsf{Adv}\left[\frac{1}{2}\right]^{\mathsf{Leave-Privacy}}_{\mathcal{A},\mathsf{GS}}(1^\lambda)$$

is negligible.

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

4 OUR CONSTRUCTION

In this section we formalize the group signature proposed in the introduction. We present the full algorithms in Figure 3. The idea of our construction is as follows. The issuer uses signatures on equivalence classes to certify group members' SFPK public keys[4]. As already noted by Backes et al. [5] this forms self-blindable certificates, i.e. each member can randomize the certificate and their

[3] Note that two users cannot join in the same epoch by the definition of AddU. Schemes, where batch additions in the same epoch are possible may, however, still achieve membership hiding.

[4] The definition by Hanser and Slamanig uses bilinear groups BG but this primitive is not limited to the bilinear setting and BG can be seen as parameters.

public key which is computationally indistinguishable from the original public key used during the issuing procedure. To add and revoke members, each epoch the issuer generates a new SPS-EQ keypair and puts the public key in the epoch information. To prevent malicious epoch information, the issuer signs the SPS-EQ public key using a standard digital signature scheme. To protect the identities of members, the issuer does not directly publish the new certificates but uses a randomization, i.e. certificates for public keys that are in relation to keys of members. To allow the members to restore the right certificate, the issuer encrypts the random coins that can be used to restore the original certificate. The encryption is done under the members encryption key. What is more, key-privacy ensures that the ciphertexts do not leak the identities. Note, that the join/issue session in our construction is non-interactive in the sense that the group manager can add members to the group given only their user public key, hence the GS.Join algorithm is trivial.

To enable tracing we use the canonical representative of SFPK, i.e. a signer encrypts their canonical representative under the tracing authority public key and uses proof system $\Pi_{\mathsf{GS.Trace}}$ to prove in statement x_{Sign} that the randomized SFPK public key is in relation to this encrypted key.

Statement x_{Sign}:

$\exists\, (\mathsf{pk}_{\mathsf{SFPK}}, r)$ s. t.

$\mathsf{SFPK.ChgPK}(\mathsf{pk}_{\mathsf{SFPK}}, r) = \mathsf{pk}'_{\mathsf{SFPK}}$

$\wedge\ \mathsf{IsCanonical}(\mathsf{pk}_{\mathsf{SFPK}})$

$\vee\ c_{\mathsf{SFPK}} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{SFPK}}, \mathsf{tpk})$

The complete group signature is composed of a randomized SPS-EQ certificate from the issuer on the randomized SFPK public key of the member, a ciphertext of the canonical representative, a proof that this ciphertext is sound and a SFPK signature on all those values and the message.

Finally, the proof system $\Pi_{\mathsf{GS.Judge}}$ is used by the tracing authority to prove in statement x_{Trace} that the decrypted public key corresponds to public keys used during the issuing procedure.

Statement x_{Trace}:

$\exists\, (\mathsf{tsk})$ s. t.

$(\vec{upk}[\mathsf{uid}]) \xleftarrow{s} \mathsf{PKE.Dec}(\mathsf{tsk}, c_{\mathsf{SFPK}})$

$\wedge\ \mathsf{tsk} \rightleftharpoons \mathsf{tpk}$

Theorem 1 (Join Privacy). *Our construction has private joins if the encryption scheme used by the signers is* IND − CPA *secure and has* 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 $\vec{gsk}[\mathsf{uid}_b] = (\mathsf{sk}_{\mathsf{SFPK}}, \mathsf{sk}_{\mathsf{Enc}})$ 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, so $\Pr[S_0] = \mathsf{Adv}^{\mathsf{join-privacy}}_{\mathsf{GS},\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_1$ We modify how the challenge group information is created. For this we generate a fresh public key encryption key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{s} \mathsf{PKE.KeyGen}(1^\lambda)$. 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 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 representative. We will also use this fresh key to sign in the queries made to PrivChall. We observe that $\Pr[S_3] \leq \Pr[S_2] + \mathsf{Adv}^{\mathsf{c-h}}_{\mathsf{SFPK},\mathcal{A}}(\lambda)$. 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

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

□

Theorem 2 (Leave Privacy). *Our construction has leave privacy if the encryption scheme used by the signers is* IND − CPA *secure and has* 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 Leave − Privacy game, where bit b is fixed to 0.

$\mathbf{GAME}_1$ 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 $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{ik-cpa}}_{\mathcal{A},\mathsf{PKE}}(\lambda)$.

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

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

□

Theorem 3 (Anonymity). *Our construction is anonymous if the* SPS-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.*

GS.Setup(1^λ)

$(\rho_{\text{SFPK}}, \cdot) \xleftarrow{s} \text{SFPK.CRSGen}(1^\lambda)$

$\text{BG} \xleftarrow{s} \text{SPS.BGGen}(1^\lambda)$

$\rho_{\mathcal{J}} \xleftarrow{s} \Pi_{\text{GS.Judge}}.\text{Setup}(1^\lambda)$

$\rho_{\mathcal{T}} \xleftarrow{s} \Pi_{\text{GS.Trace}}.\text{Setup}(1^\lambda);\ \tau := 0$

return param := $(1^\lambda, \text{BG}, \rho_{\text{SFPK}}, \rho_{\mathcal{J}}, \rho_{\mathcal{T}})$

GS.Issue($\text{info}_{\tau_{current}}, \text{msk}, \text{uid}, \vec{upk}[\text{uid}]$)

$\text{msk} = (\text{sk}_{\text{DS}}, \text{sk}_{\text{SPS}}),\ \vec{upk}[\text{uid}] = (\text{pk}_{\text{SFPK}}, \text{pk}_{\text{Enc}})$

$\text{info}_{\tau_{current}}[\text{uid}] = (\text{pk}_{\text{SPS}}, \sigma_{\text{DS}}, \text{Active})$

abort if $\neg\text{IsCanonical}(\text{pk}_{\text{SFPK}})$

$k \xleftarrow{s} \text{coin}; c \xleftarrow{s} \text{PKE.Enc}(\text{pk}_{\text{Enc}}, k);$

$\sigma_{\text{SPS}} \xleftarrow{s} \text{SPS.Sign}(\text{SFPK.ChgPK}(\text{pk}_{\text{SFPK}}, k), \text{sk}_{\text{SPS}})$

$\text{Active}' := \text{Active} \cup \{(c, \sigma_{\text{SPS}})\}$

$\text{info}_{\tau_{current}}[\text{uid}] := (\text{pk}_{\text{SPS}}, \sigma_{\text{DS}}, \text{Active}')$

$\vec{reg}[\text{uid}] := \vec{upk}[\text{uid}]$

GS.RevokeMember(gpk, msk, $\text{info}_{\tau_{current}}, \mathcal{S}, \vec{reg}$)

$\text{msk} = (\text{sk}_{\text{DS}}, \text{sk}_{\text{SPS}})$

$(\text{sk}'_{\text{SPS}}, \text{pk}'_{\text{SPS}}) \xleftarrow{s} \text{SPS.KGen}(\text{BG}, 2)$

$\text{msk} := (\text{sk}_{\text{DS}}, \text{sk}'_{\text{SPS}})$

$\text{info}_{\tau_{current}} = (\cdot, \cdot, \text{Active});\ A := \{i | \text{ user } i \text{ is active}\}$

foreach $i \in A \setminus \mathcal{S}$

$\vec{reg}[\text{uid}] = (\text{pk}^i_{\text{SFPK}}, \text{pk}^i_{\text{Enc}})$

$k \xleftarrow{s} \text{coin}; c \xleftarrow{s} \text{Enc}(\text{pk}^i_{\text{Enc}}, k);$

$\sigma_{\text{SPS}} \xleftarrow{s} \text{SPS.Sign}(\text{SFPK.ChgPK}(\text{pk}^i_{\text{SFPK}}, k), \text{sk}_{\text{SPS}})$

$\text{Active}' := \text{Active}' \cup (c, \sigma_{\text{SPS}})$

return $\text{info}_{\tau_{new}} = (\text{pk}'_{\text{SPS}}, \text{DS.Sign}(\text{sk}_{\text{DS}}, \text{pk}'_{\text{SPS}}), \text{Active}')$

GS.KGen$_{\mathcal{M}}$(param)

$(\text{sk}_{\text{DS}}, \text{pk}_{\text{DS}}) \xleftarrow{s} \text{DS.KGen}(1^\lambda)$

$(\text{sk}_{\text{SPS}}, \text{pk}_{\text{SPS}}) \xleftarrow{s} \text{SPS.KGen}(\text{BG}, \ell)$

$\text{info} := (\text{pk}_{\text{SPS}}, \text{DS.Sign}(\text{sk}_{\text{DS}}, \text{pk}_{\text{SPS}}), \emptyset)$

return $(\text{msk} := (\text{sk}_{\text{DS}}, \text{sk}_{\text{SPS}}),$
$\text{mpk} := \text{pk}_{\text{DS}}, \text{info})$

GS.KGen$_{\mathcal{T}}$(param)

$(\text{tsk}, \text{tpk}) \xleftarrow{s} \text{PKE.KeyGen}(1^\lambda)$

return (tsk, tpk)

GS.KGen$_{\mathcal{U}}(1^\lambda)$

$(\text{sk}_{\text{SFPK}}, \text{pk}_{\text{SFPK}}) \xleftarrow{s} \text{SFPK.KGen}(1^\lambda)$

$(\text{sk}_{\text{Enc}}, \text{pk}_{\text{Enc}}) \xleftarrow{s} \text{PKE.KeyGen}(1^\lambda)$

return $(\vec{usk}[\text{uid}] := (\text{sk}_{\text{SFPK}}, \text{sk}_{\text{Enc}}),$
$\vec{upk}[\text{uid}] := (\text{pk}_{\text{SFPK}}, \text{pk}_{\text{Enc}}))$

GS.Sig(gpk, $\vec{gsk}[\text{uid}], \text{info}_\tau, m$)

$\text{info}_{\tau_{current}} = (\text{pk}_{\text{SPS}}, \cdot, \text{Active})$

$\vec{gsk}[\text{uid}] = (\text{sk}_{\text{SFPK}}, \text{sk}_{\text{Enc}});\ \text{gpk}[\text{uid}] = (\text{pk}_{\text{SFPK}}, \text{pk}_{\text{Enc}})$

abort if $\neg\exists\,(c, \sigma_{\text{SPS}}) \in \text{Active}$ s. t.

$k \leftarrow \text{PKE.Dec}(c, \text{sk}_{\text{Enc}})$ and

$\text{SPS.Verify}(\text{SFPK.ChgPK}(\text{pk}_{\text{SFPK}}, k), \sigma_{\text{SPS}}, \text{pk}_{\text{SPS}}) = 1$

$r \xleftarrow{s} \text{coin};\ (\text{sk}'_{\text{SFPK}}, \text{pk}'_{\text{SFPK}}) \leftarrow \text{SFPK.ChgKeys}(\text{sk}_{\text{SFPK}}, \text{pk}_{\text{SFPK}}, r)$

$\sigma'_{\text{SPS}} \xleftarrow{s} \text{SPS.ChgRep}(\text{pk}_{\text{SFPK}}, \sigma_{\text{SPS}}, r \cdot k^{-1}, \text{pk}_{\text{SPS}})$

$c_{\text{SFPK}} \xleftarrow{s} \text{PKE.Enc}(\text{pk}_{\text{SFPK}}, \text{tpk})$

$\Pi_{\text{SFPK}} \xleftarrow{s} \Pi_{\text{GS.Trace}}.\text{Prove}(\rho_{\mathcal{T}}, x_{\text{Sign}}, w = (\text{pk}_{\text{SFPK}}, r))$

$\text{Sig}_{\text{SFPK}} \xleftarrow{s} \text{SFPK.Sign}(\text{sk}'_{\text{SFPK}}, m||\tau_{current}||\text{pk}'_{\text{SFPK}}||\sigma'_{\text{SPS}}||\Pi_{\text{SFPK}}||c_{\text{SFPK}})$

return $\Sigma := (\text{pk}'_{\text{SFPK}}, \sigma'_{\text{SPS}}, \Pi_{\text{SFPK}}, c_{\text{SFPK}}, \text{Sig}_{\text{SFPK}})$

GS.Vf(gpk, $\text{info}_\tau, m, \Sigma$)

$\text{info}_\tau = (\text{pk}_{\text{SPS}}, \sigma_{\text{DS}}, \cdot);\ \text{mpk} = \text{pk}_{\text{DS}}$

$\Sigma = (\text{pk}_{\text{SFPK}}, \sigma_{\text{SPS}}, \Pi_{\text{SFPK}}, c_{\text{SFPK}}, \text{Sig}_{\text{SFPK}})$

/ x_{Sign} is the same statement as in GS.Sig

reject if $\text{DS.Verify}(\text{pk}_{\text{DS}}, \text{pk}_{\text{SPS}}, \sigma_{\text{DS}}) = \text{reject}$ or

$\Pi_{\text{PPE}}.\text{Verify}(\rho_\Pi, x_{\text{Sign}}, \Pi_{\text{SFPK}}) = \text{reject}$ or

$\text{SPS.Verify}(\text{pk}_{\text{SPS}}, \text{pk}_{\text{SFPK}}, \sigma_{\text{SPS}}) = \text{reject}$

$M := m||\tau||\text{pk}_{\text{SFPK}}||\sigma_{\text{SPS}}||\Pi_{\text{SFPK}}||c_{\text{SFPK}}$

return $\text{SFPK.Verify}(\text{pk}_{\text{SFPK}}, M, \text{Sig}_{\text{SFPK}})$

GS.Trace(gpk, tsk, $\text{info}_\tau, \vec{reg}, m, \Sigma$)

$\Sigma = (\text{pk}_{\text{SFPK}}, \sigma_{\text{SPS}}, \Pi_{\text{SFPK}}, c_{\text{SFPK}}\text{Sig}_{\text{SFPK}})$

$(\text{pk}_{\text{SFPK}}) \xleftarrow{s} \text{PKE.Dec}(\text{tsk}, c_{\text{SFPK}})$

abort if $\neg\exists\text{uid}$ s. t. $\vec{reg}[\text{uid}] = (\text{pk}_{\text{SFPK}}, \cdot)$

$\pi \xleftarrow{s} \Pi_{\text{GS.Judge}}.\text{Prove}(\rho_{\mathcal{J}}, x_{\text{Trace}}, w = (\text{tsk}))$

return (uid, π)

GS.Judge(gpk, uid, $\text{info}_\tau, \pi_{\text{Trace}}, \vec{upk}[\text{uid}], m, \Sigma$)

reject if $\text{GS.Vf}(\text{gpk}, \text{info}_\tau, m, \Sigma) = \text{reject}$

$\Sigma = (\cdot, \cdot, \Pi_{\text{SFPK}}, c_{\text{SFPK}}\cdot);\ \vec{upk}[\text{uid}] = (\text{pk}_{\text{SFPK}}, \cdot)$

/ Statement x_{Trace} as in GS.Trace

return $\Pi_{\text{GS.Judge}}.\text{Verify}(\rho_{\mathcal{J}}, x_{\text{Trace}}, \pi)$

Figure 3: Our generic construction of fully dynamic group signatures.

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

$$\Sigma^* = (\text{pk}^*_{\text{SFPK}}, \sigma^*_{\text{SPS}}, \Pi^*_{\text{SFPK}}, c^*_{\text{SFPK}}, \text{Sig}^*_{\text{SFPK}}).$$

GAME$_1$: We simulate the proof generated in 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 $|\Pr[S_1] - \Pr[S_0]| \leq \text{Adv}^{\text{zk}}_{\mathcal{A}, \Pi_{\text{GS.Judge}}}(\lambda)$.

GAME$_2$: We change the way the Trace oracle works. Instead of using tsk to decrypt pk_{SFPK} from c_{SFPK}, we first extract the witness $(\text{pk}_{\text{SFPK}}, r)$ and use pk_{SFPK} instead. What is more, we simulate the proof Π^*_{SFPK}, which is part of the challenges signature.

Note that since the proof system $\Pi_{\text{GS.Trace}}$ is simulation-sound extractable it follows that $|\Pr[S_2] - \Pr[S_1]| \leq \text{Adv}^{\text{sse}}_{\mathcal{A}, \Pi_{\text{GS.Trace}}}(\lambda)$.

GAME$_3$: We change the way the ciphertext c^*_{SFPK} is computed. Instead of encrypting the canonical representative, we encrypt the

value 0.

Note that because of the changes made in the previous game, the Trace oracle works as in $\mathbf{GAME}_2$. Thus, we have that $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$

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

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

$\mathbf{GAME}_5$: 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 $\Pr[S_4] = n \cdot \Pr[S_5]$.

$\mathbf{GAME}_6$: 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 $\mathsf{KGen}_{\mathsf{SFPK}}$.

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 $\mathbf{GAME}_3$, computes Π^*_{SFPK} as in $\mathbf{GAME}_2$, computes $c^*_{\mathsf{SFPK}} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}'_{\mathsf{SFPK}}, \mathsf{tpk})$, asks its signing oracle for $\mathsf{Sig}^*_{\mathsf{SFPK}}$ under message $m^*||\tau^*||\mathsf{pk}^*_{\mathsf{SFPK}}||\sigma^*_{\mathsf{SPS}}||\Pi^*_{\mathsf{SFPK}}||c^*_{\mathsf{SFPK}}$, and returns $\Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, c^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$. 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 $|\Pr[S_6] - \Pr[S_5]| \leq \mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$.

We now argue that the only way the adversary $\mathcal{A}$ can break anonymity is by creating a randomization

$$\Sigma' = (\mathsf{pk}'_{\mathsf{SFPK}}, \sigma'_{\mathsf{SPS}}, \Pi'_{\mathsf{SFPK}}, c'_{\mathsf{SFPK}}, \mathsf{Sig}'_{\mathsf{SFPK}})$$

of the signature $\Sigma^* = (\mathsf{pk}^*_{\mathsf{SFPK}}, \sigma^*_{\mathsf{SPS}}, \Pi^*_{\mathsf{SFPK}}, c^*_{\mathsf{SFPK}}, \mathsf{Sig}^*_{\mathsf{SFPK}})$ and use Σ' in a query to the Trace oracle. Since in $\mathbf{GAME}_5$ 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:

$$\begin{aligned}\Pr[S_0] \leq & n \cdot \left(\mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\mathsf{seuf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)\right) \\ & + \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{sse}}_{\mathcal{A},\Pi_{\mathsf{GS.Trace}}}(\lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{A},\Pi_{\mathsf{GS.Judge}}}(\lambda).\end{aligned}$$

□

Theorem 4 (Traceability). *Our construction is traceable if the* SPS-EQ *scheme is existential unforgeable under chosen-message attacks, the* SFPK *scheme is existential unforgeable and the signature scheme used by the Issuer is existential unforgeable under chosen-message attacks.*

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

Theorem 6 (Functional Tracing Soundness). *Our construction 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.*

The full proofs for theorems 4, 5 and 6 can be found in [6].

5 DISCUSSION AND EFFICIENT INSTANTIATION

The generic construction presented above can be easily instantiated in the standard model, without random oracles, using known schemes. In particular, we can use the standard model signatures on equivalence classes by Fuchsbauer and Gay [31] and one of the compatible SFPK signature schemes by Backes et al. [5]. For the encryption scheme one can use El Gamal encryption and standard model digital signatures. Finally, both proof systems can be instantiated using the simulation-sound system by Groth [34]. However, due to the simulation-sound proof system and the large public keys of the SFPK schemes, the signature size is not competitive with existing schemes. We will now show how to minimize the signature size, while still using only building blocks that are secure under standard assumptions and without random oracles. The objective is to instantiate our construction in a way that it has shorter signatures than the current state-of-the-art scheme by Libert-Peters-Yung [45] presented at Crypto'15, which is only secure in a weaker model.

Optimization. To decrease the signature size we have to solve the following problems:

(1) The proof system $\Pi_{\mathsf{GS.Trace}}$ must allow the security reduction for the anonymity experiment to simulate the challenged proof and at the same time extract witnesses to properly simulate the Trace oracle,

Scheme	Signature size* $[bits]$	Group public* key size	Membership	Assumptions
Libert-Peters-Yung [45]	8 448	$O(\lambda)$	static	standard
Boyen-Waters [20]‡	6 656	$O(\lambda)$	static	q-type
Boneh-Boyen-Shacham [13]	2 304	2048-bit	static	q-type
Bichsel et al. [12]	1 280	1024-bit	partially dynamic†	interactive
Groth [35]	13 056	$O(1)$	partially dynamic	q-type
Libert-Peters-Yung [45]	14 848	$O(\lambda)$	partially dynamic	standard
Bootle et al. [17]	$O(\log N)$	$O(1)$	fully dynamic$^\clubsuit$	standard
Our generic construction	$O(1)$	$O(\lambda)$	fully dynamic + membership hiding$^\clubsuit$	standard
…instantiated with Scheme 5	13 056	$O(\lambda)$	fully dynamic + membership hiding$^\clubsuit$	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

$^\clubsuit$ The size of the epoch information is $O(N)$

† The scheme defines additionally a join↔issue procedure

‡ Adapted from type 1 to type 3 pairings as in [45]

Figure 4: Comparison of Group Signature Schemes for N Active Members

(2) The public key of the SFPK signature must be short and allow for a simple proof of canonical representation,
(3) If possible, simplification of the statement proven in $\Pi_{\mathsf{GS.Trace}}$.

First, we replace the simulation-sound system with a simple NIWI proof system. In fact, we instantiate all building blocks such that we can use the popular Groth-Sahai proofs for pairing product equations. To do so, we introduce a trapdoor witness that can be used by the reduction to simulate the proof, while still being able to extract the witness. Of course, we have to prevent the adversary from using this trapdoor to create valid proofs. We achieve this by introducing a new element $K_2 = g_2^k$ as part of the groups public key that will be part of the statement. The trapdoor witness are then two values w_1 and w_2, such that $e(w_1, K_2) = e(w_2, g_2)$. It is easy to see that any adversary that is able to compute such a witness can be used to break the DDH assumption in $\mathbb{G}_2$.

To solve the second problem we propose an SFPK scheme that works analogously to the schemes presented in [5], but allows us to use canonical representatives by moving to our weaker class-hiding definition. The scheme uses public keys in $\mathbb{G}_1 \times \mathbb{G}_1$ with the established projective equivalence relation, i.e. $\mathsf{pk} \in [\mathsf{pk}']_{\mathcal{R}}$ if there is a $\mu \in \mathbb{Z}_p^*$ such that $\mathsf{pk}_1^\mu = \mathsf{pk}'_1$ and $\mathsf{pk}_2^\mu = \mathsf{pk}'_2$. For such classes of public keys, we define the canonical representative as the public key for which the first element is just g_1. We give a full SFPK scheme based on this approach and secure under the bilinear Diffie-Hellman assumption in Section 6.

To simplify the statement proven in $\Pi_{\mathsf{GS.Trace}}$, we get rid of the ciphertext c_{SFPK}, that is used by the tracing authority to identify signers. To preserve this functionality, we allow the tracing authority to generate the parameters for the proof system $\Pi_{\mathsf{GS.Trace}}$, including an extraction trapdoor which allows to extract the used witness and compute the corresponding canonical representative.

When applying all the above techniques the statement proven by the signer will have the form:

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

The construction and proofs can be found in the full paper [6].

Efficiency of the Instantiation. The signature itself is composed of an SFPK public key $\mathsf{pk}'_{\mathsf{SFPK}}$, an SFPK signature $\mathsf{Sig}_{\mathsf{SFPK}}$, an SPS-EQ signature σ'_{SPS} and proof Π_{SFPK}. To instantiate SFPK signatures we use Scheme 5, 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 $\mathbb{Z}_p^*$. Similar to the static group signature in [5], we will instantiate the SPS-EQ with the scheme from [31]. This means that the SPS-EQ signature takes 10 elements in $\mathbb{G}_1$ and 4 elements in $\mathbb{G}_2$.

Taking into account that we will use Scheme 5, the above statement can instantiated as follows. 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 it can correctly open signatures.

Instantiating those equations using the fine-tuned Groth-Sahai proofs presented in [30] (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: 2 group elements in $\mathbb{G}_2$ for the first equation, which is linear; 4 elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ for the second equation; 6 elements in $\mathbb{G}_1$ for the three witnesses in $\mathbb{G}_1$; 2 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 $\mathbb{Z}_p^*$.

The digital signature scheme DS and the public key encryption scheme PKE are standard components, an example of a key private PKE scheme is ElGamal encryption. The proof system $\Pi_{\mathsf{GS.Judge}}$ can also be instantiated using Groth-Sahai proofs for pairing product equations [30]. Note that this means that the tracing authority has

to prove correct decryption of a ciphertext (witnesses are encoded in form of El Gamal encryptions) and that its public key was generated using a DDH tuple, which can easily be expressed as pairing product equations.

We provide a comparison with existing group signature schemes in Figure 4. We omit lattice-based schemes in our comparison, because the only constant-size scheme was proposed by Ling et al. [46] and as argued by the authors the size is impractical.

Comparison with Previous Constructions. Some of the techniques used in our construction are similar to the static group signatures by Backes et al. [5] and the dynamic group signatures presented by Derler and Slamanig [29]. In particular, we use signatures on equivalence classes as a certificate of membership. The latter uses signatures of knowledge to allow for traceability and to have an actual signature. Thus, their scheme can only be efficiently instantiated in the random oracle model, since standard model instantiations of proofs of knowledge of an exponent are logarithmic in the size of the exponent. The construction of Backes et al. uses SFPK signatures and standard proof systems like ours. In their scheme, the tracing authority uses the SFPK trapdoor to distinguish public key equivalence classes. Since their scheme is static, the issuer/tracing authority can obtain this trapdoor during user key generation. In the dynamic setting, where we want to achieve non-frameability, the members generate their keys themselves in an interactive protocol with the issuer, preventing this approach. We address this by introducing the canonical representative into the notion of SFPK signature. We then allow the tracing authority to check the randomized public key, given as part of the group signature, against the canonical representative used during the joining procedure.

Group Update and Signing Complexity. We will first focus on the computational complexity of the group manager in regards to adding/removing members. To add a new member the manager creates a SPS-EQ signature under the member's public key that acts as a kind of certificate. Furthermore it encrypts the random coins used to blind the member's public key. The signature and ciphertext are then added to the current epoch information. It is easy to see that the complexity of adding new users is constant in the number of active group members. By contrast, removing members requires an update of the whole epoch information, i.e. the group manager generates a fresh SPS-EQ public key and issues fresh certificates for each active member. From a practical point of view this trade-off is acceptable. In particular, we note that adding a new member is a process that requires the interaction between the group manager and the user, at least for the group manager to obtain the new member's public key. On the other hand, to revoke a member no interaction is required and the group manager can perform the operation offline. What is more, this process can be easily parallelized (each entry can be computed independently) and changes can be batched (removing more users can be done at once).

It is worth noting that linear complexity in the number of active members is not inherent for membership revocation, since one could use cryptographic accumulators [27] to store information about active members. In such a case the complexity of updates for the group manager would be linear only in the number of changes (additions/removes) and not in the number of active users. Unfortunately, we are not aware of any construction that could utilize this idea without violating the membership privacy properties. Note that cryptographic accumulators require auxiliary information about the added/removed user to be published in order for the other members to be able to generate membership proofs for the new accumulator. This information could easily be used by an adversary to break membership privacy.

The form of the epoch information influences the complexity of the signing procedure. In our construction, the signer has to find the entry that corresponds to its identity in the current epoch. To simplify presentation we made this process linear in the number of active members. However, it is easy to see that this process need only be performed once per epoch, i.e. once a member updates their certificate it can be used to create all subsequent signatures by this member in the same epoch. In the current construction the epoch information is represented as a set but it can easily be represented as an ordered list. To facilitate this, the group manager can create unique identifiers that can be used to sort the list and members can find the correct entry using binary search. To preserve membership privacy, during registration members generate an additional public key for a key agreement scheme and store the private key securely. Each epoch the group manager also generates a fresh public key for the key agreement scheme and uses the shared key as a secret to generate the unique and hidden identifier, e.g. using a pseudorandom function. This identifier can easily be reconstructed by the corresponding member but is indistinguishable from a random value for all other members and third parties.

Hiding the Group Size. In this paper we do not consider the problem of hiding the group size but we show how it can be partially solved using dummy members and a trade-off in the group update complexity. Our current construction does not hide the size of the group, since the epoch information size is linear in the number of active group members. The simplest idea is to create the first epoch information with dummy users and update it with every addition/removing. It is easy to see that the size of the epoch information will be constant (during setup a maximal number of members has to be chosen) if we replace dummy members by real ones and vice versa. Unfortunately, this approach requires the whole epoch information to be updated for both addition and revocation of members and not only during the revocation of a member. It also leaks an upper bound on the group size and requires an update complexity that is at least linear in the number of active members. However, we argue that from a practical point of view using dummy users can be acceptable to protect the size of the group.

A different approach would be to again use cryptographic accumulators, since they can be constant size and independent of the number of accumulated values. Unfortunately, this solution is not better than the above one. To prove membership of a value in a new accumulator the witness corresponding to the previous state must be updated. Existing definitions and constructions of accumulators [27] require that the added/removed value must be used for this update. Thus, the epoch information must contain this value is some form (e.g. encrypted). However, in the end since this value must be part of the epoch information, an adversary can easily backtrack all previous published information and calculate the size of the group.

6 EFFICIENT SIGNATURES WITH FLEXIBLE PUBLIC KEY

Here we propose our signatures with flexible public key. We propose a scheme that is closely related to the ones proposed in [5]. 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 [5], this scheme only has class-hiding with key corruption but as already shown this is still sufficient for group signature constructions.

We assume that both the SFPK.KGen and SFPK.TKGen 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 SFPK.CRSGen. The SFPK.ChgPK and 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 5.

Theorem 7 (Unforgeability). *Scheme 5 is strongly existential unforgeable under flexible public key in the crs model, assuming the bilinear decisional Diffie-Hellman 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 of the adversary:

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 O^1 or O^2, where

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

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 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 O^1 or 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 $M^* = g_1^{e^*} \cdot \hat{g}^{s^*} = g_1^e \cdot \hat{g}^s = M$.

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 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 O^1 and O^2.

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 O^1), the reduction $\mathcal{R}$ follows the following steps:

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

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

(6) 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

$$M^* = g_1^{e^*} \hat{g}^{s^*} \text{ and } e^* = \mathsf{H}(m^*||\mathsf{Sig}^2_{\mathsf{SFPK}}||\mathsf{Sig}^3_{\mathsf{SFPK}}||\mathsf{pk}^*_{\mathsf{SFPK}}).$$

In such a case, the reduction works as follows:

(1) parse $\mathsf{Sig}^*_{\mathsf{SFPK}}$ as $(\mathsf{Sig}^1_{\mathsf{SFPK}}, \mathsf{Sig}^2_{\mathsf{SFPK}}, \mathsf{Sig}^3_{\mathsf{SFPK}}, s^*)$,
(2) compute

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

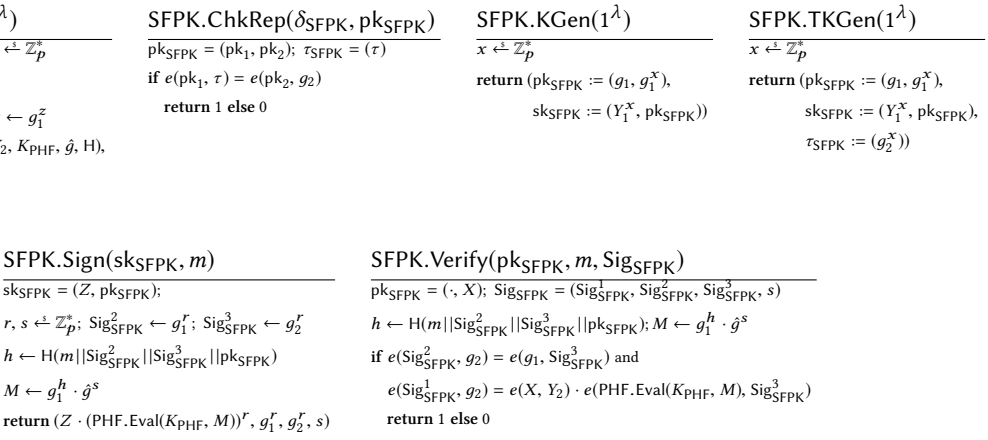

Figure 5: Our Flexible Signatures.

(3) parse $\mathsf{pk}^*_{\mathsf{SFPK}}$, 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 $g_1^{t^*}$,
(4) output 1 iff $e(g_1^{a \cdot b \cdot t^*}, g_2^c) = e(g_1^{t^*}, g_2^d)$.

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 O^1 and O^2, which is not the case for type 1 and type 2 adversaries.

□

Theorem 8 (Class-Hiding with Key Corruption). *Scheme 5 is class-hiding with key corruption in the crs model, assuming the decisional Diffie-Hellman assumption holds.*

Proof. 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 SFPK.KGen, $\mathsf{sk}_0 = (Y_1^{x_0}, \mathsf{pk}_0)$ and $\mathsf{sk}_1 = (Y_1^{x_1}, \mathsf{pk}_1)$ the corresponding secret keys and $\hat{b}$ be the bit chosen by the challenger.

$\mathbf{GAME}_1$: In this game we do not use the 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 $B' \xleftarrow{s} \mathbb{G}_1$ 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 shown 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 $r_0, r_1 \xleftarrow{s} \mathbb{Z}_p^*$ 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 $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

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 $\mathsf{Adv}^{\text{c-h}}_{\mathcal{A}, \mathsf{SFPK}}(\lambda) = \Pr[S_0] \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$. □

ACKNOWLEDGMENTS

This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for CISPA and the CISPA-Stanford Center for Cybersecurity (FKZ: 16KIS0762).

REFERENCES

[1] Michel Abdalla and Bogdan Warinschi. 2004. On the Minimal Assumptions of Group Signature Schemes. In *ICICS 2004*. 1–13. https://doi.org/10.1007/978-3-540-30191-2_1

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

[3] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. 2000. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO 2000*. 255–270. https://doi.org/10.1007/3-540-44598-6_16

[4] Giuseppe Ateniese and Breno de Medeiros. 2003. Efficient Group Signatures without Trapdoors. In *ASIACRYPT 2003*. 246–268. https://doi.org/10.1007/978-3-540-40061-5_15

[5] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. [n.d.]. Signatures with Flexible Public Key: Introducing Equivalence Classes for Public Keys. In *ASIACRYPT 2018*. To appear.

[6] Michael Backes, Lucjan Hanzlik, and Jonas Schneider. 2018. Membership Privacy for Fully Dynamic Group Signatures. *IACR Cryptology ePrint Archive* 2018 (2018), 641. https://eprint.iacr.org/2018/641

[7] Foteini Baldimtsi, Jan Camenisch, Maria Dubovitskaya, Anna Lysyanskaya, Leonid Reyzin, Kai Samelin, and Sophia Yakoubov. 2017. Accumulators with Applications to Anonymity-Preserving Revocation. In *EuroS&P 2017*. IEEE, 301–315. https://doi.org/10.1109/EuroSP.2017.13

[8] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. 2009. Randomizable Proofs and Delegatable Anonymous Credentials. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings (Lecture Notes in Computer Science)*, Shai Halevi (Ed.), Vol. 5677. Springer, 108–125. https://doi.org/10.1007/978-3-642-03356-8_7

[9] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. 2001. Key-Privacy in Public-Key Encryption. In *ASIACRYPT 2001*. 566–582. https://doi.org/10.1007/3-540-45682-1_33

[10] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. 2003. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *EUROCRYPT 2003*. 614–629. https://doi.org/10.1007/3-540-39200-9_38

[11] Mihir Bellare, Haixia Shi, and Chong Zhang. 2005. Foundations of Group Signatures: The Case of Dynamic Groups. In *CT-RSA 2005*. 136–153. https://doi.org/10.1007/978-3-540-30574-3_11

[12] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. 2010. Get Shorty via Group Signatures without Encryption. In *SCN 2010*. 381–398. https://doi.org/10.1007/978-3-642-15317-4_24

[13] Dan Boneh, Xavier Boyen, and Hovav Shacham. 2004. Short Group Signatures. In *CRYPTO 2004*. 41–55. https://doi.org/10.1007/978-3-540-28628-8_3

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

[15] Dan Boneh and Matthew K. Franklin. 2001. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*. 213–229. https://doi.org/10.1007/3-540-44647-8_13

[16] Dan Boneh and Matthew K. Franklin. 2003. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.* 32, 3 (2003), 586–615. https://doi.org/10.1137/S0097539701398521

[17] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. 2016. Foundations of Fully Dynamic Group Signatures. In *ACNS 2016*. 117–136. https://doi.org/10.1007/978-3-319-39555-5_7

[18] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. [n.d.]. Short Accountable Ring Signatures Based on DDH. In *ESORICS 2015*.

[19] Xavier Boyen and Brent Waters. 2006. Compact Group Signatures Without Random Oracles. In *EUROCRYPT 2006*. 427–444. https://doi.org/10.1007/11761679_26

[20] Xavier Boyen and Brent Waters. 2007. Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In *PKC 2007*. 1–15. https://doi.org/10.1007/978-3-540-71677-8_1

[21] Jan Camenisch and Jens Groth. 2004. Group Signatures: Better Efficiency and New Theoretical Aspects. In *SCN 2004*. 120–133. https://doi.org/10.1007/978-3-540-30598-9_9

[22] Jan Camenisch and Anna Lysyanskaya. 2004. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO 2004*. 56–72. https://doi.org/10.1007/978-3-540-28628-8_4

[23] Melissa Chase and Anna Lysyanskaya. 2006. On Signatures of Knowledge. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings (Lecture Notes in Computer Science)*, Cynthia Dwork (Ed.), Vol. 4117. Springer, 78–96. https://doi.org/10.1007/11818175_5

[24] David Chaum and Eugène Van Heyst. 1991. Group Signatures. In *EUROCRYPT'91 (LNCS)*, Donald W. Davies (Ed.), Vol. 547. Springer, Heidelberg, 257–265.

[25] Sherman S. M. Chow, Haibin Zhang, and Tao Zhang. 2017. Real Hidden Identity-Based Signatures. In *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers (Lecture Notes in Computer Science)*, Aggelos Kiayias (Ed.), Vol. 10322. Springer, 21–38. https://doi.org/10.1007/978-3-319-70972-7_2

[26] Elizabeth C. Crites and Anna Lysyanskaya. 2019. Delegatable Anonymous Credentials from Mercurial Signatures. In *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings (Lecture Notes in Computer Science)*, Mitsuru Matsui (Ed.), Vol. 11405. Springer, 535–555. https://doi.org/10.1007/978-3-030-12612-4_27

[27] David Derler, Christian Hanser, and Daniel Slamanig. 2015. Revisiting Cryptographic Accumulators, Additional Properties and Relations to Other Primitives. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings (Lecture Notes in Computer Science)*, Kaisa Nyberg (Ed.), Vol. 9048. Springer, 127–144. https://doi.org/10.1007/978-3-319-16715-2_7

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

[29] David Derler and Daniel Slamanig. 2018. Highly-Efficient Fully-Anonymous Dynamic Group Signatures. In *AsiaCCS 2018*, Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim (Eds.). ACM, 551–565. https://doi.org/10.1145/3196494.3196507

[30] Alex Escala and Jens Groth. 2014. Fine-Tuning Groth-Sahai Proofs. In *PKC 2014*. 630–649. https://doi.org/10.1007/978-3-642-54631-0_36

[31] Georg Fuchsbauer and Romain Gay. [n.d.]. Weakly Secure Equivalence-Class Signatures from Standard Assumptions. In *PKC 2018*.

[32] Jun Furukawa and Shoko Yonezawa. 2004. Group Signatures with Separate and Distributed Authorities. In *SCN 2004*. 77–90. https://doi.org/10.1007/978-3-540-30598-9_6

[33] Taher El Gamal. 1984. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *CRYPTO 1984*. 10–18. https://doi.org/10.1007/3-540-39568-7_2

[34] Jens Groth. 2006. Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures. In *ASIACRYPT 2006 (Lecture Notes in Computer Science)*, Xuejia Lai and Kefei Chen (Eds.), Vol. 4284. Springer, 444–459. https://doi.org/10.1007/11935230_29

[35] Jens Groth. 2007. Fully Anonymous Group Signatures Without Random Oracles. In *ASIACRYPT 2007*. 164–180. https://doi.org/10.1007/978-3-540-76900-2_10

[36] Jens Groth and Amit Sahai. 2008. Efficient Non-interactive Proof Systems for Bilinear Groups. In *EUROCRYPT 2008*. 415–432. https://doi.org/10.1007/978-3-540-78967-3_24

[37] Christian Hanser and Daniel Slamanig. [n.d.]. Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials. In *ASIACRYPT 2014*.

[38] Dennis Hofheinz and Eike Kiltz. 2008. Programmable Hash Functions and Their Applications. In *CRYPTO 2008*. 21–38. https://doi.org/10.1007/978-3-540-85174-5_2

[39] Aggelos Kiayias and Moti Yung. 2005. Efficient Secure Group Signatures with Dynamic Joins and Keeping Anonymity Against Group Managers. In *Mycrypt 2005*. 151–170. https://doi.org/10.1007/11554868_11

[40] Aggelos Kiayias and Moti Yung. 2005. Group Signatures with Efficient Concurrent Join. In *EUROCRYPT 2005*. 198–214. https://doi.org/10.1007/11426639_12

[41] Aggelos Kiayias and Moti Yung. 2006. Secure scalable group signature with dynamic joins and separable authorities. *IJSN* 1, 1/2 (2006), 24–45. https://doi.org/10.1504/IJSN.2006.010821

[42] Aggelos Kiayias and Hong-Sheng Zhou. 2007. Hidden Identity-Based Signatures. In *Financial Cryptography and Data Security, 11th International Conference, FC 2007, and 1st International Workshop on Usable Security, USEC 2007, Scarborough, Trinidad and Tobago, February 12-16, 2007. Revised Selected Papers (Lecture Notes in Computer Science)*, Sven Dietrich and Rachna Dhamija (Eds.), Vol. 4886. Springer, 134–147. https://doi.org/10.1007/978-3-540-77366-5_14

[43] Benoît Libert, Thomas Peters, and Moti Yung. 2012. Group Signatures with Almost-for-Free Revocation. In *CRYPTO 2012*. 571–589. https://doi.org/10.1007/978-3-642-32009-5_34

[44] Benoît Libert, Thomas Peters, and Moti Yung. 2012. Scalable Group Signatures with Revocation. In *EUROCRYPT 2012*. 609–627. https://doi.org/10.1007/978-3-642-29011-4_36

[45] Benoît Libert, Thomas Peters, and Moti Yung. 2015. Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions. In *CRYPTO 2015*. 296–316. https://doi.org/10.1007/978-3-662-48000-7_15

[46] San Ling, Khoa Nguyen, Huaxiong Wang, and Yanhong Xu. [n.d.]. Constant-Size Group Signatures from Lattices. In *PKC 2018*.

[47] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. 2012. On the Security of Dynamic Group Signatures: Preventing Signature Hijacking. In *PKC 2012*. 715–732. https://doi.org/10.1007/978-3-642-30057-8_42

[48] Gene Tsudik and Shouhuai Xu. 2003. Accumulating Composites and Improved Group Signing. In *ASIACRYPT 2003*. 269–286. https://doi.org/10.1007/978-3-540-40061-5_16

[49] Brent Waters. 2005. Efficient Identity-Based Encryption Without Random Oracles. In *EUROCRYPT 2005*. 114–127. https://doi.org/10.1007/11426639_7