Signatures with Flexible Public Key: A Unified Approach to Privacy-Preserving Signatures (Full Version)

Michael Backes[1,3], Lucjan Hanzlik[2,3], Kamil Kluczniak[4], and Jonas Schneider[2,3]

[1] CISPA Helmholtz Center i.G.,
backes@cispa.saarland
[2] CISPA, Saarland University,
{hanzlik, jonas.schneider}@cispa.saarland
[3] Saarland Informatics Campus,
[4] The Hong Kong Polytechnic University, Department of Computing,
kkklucz@polyu.edu.hk

Abstract. We introduce a new cryptographic primitive called signatures with flexible public key. We divide the key space into equivalence classes induced by a relation $\mathcal{R}$. A signer can efficiently change his key pair to a different representative of the same class, but without a trapdoor it is hard to distinguish if two public keys are related.
This primitive offers a unified approach to the modular construction of signature schemes with privacy-preserving components. Namely, we show how to build the first ring signature scheme in the plain model without trusted setup, where signature size depends only sub-linearly on the number of ring members. Moreover, we show how to combine our primitive with structure-preserving signatures on equivalence classes (SPS-EQ) to construct static group signatures and self-blindable certificates. When properly instantiated, the result is a group signature scheme that has a shorter signature size than the current state-of-the-art scheme by Libert, Peters, and Yung from Crypto'15.
In its own right, our primitive has stand-alone applications in the cryptocurrency domain. In particular it enables the straightforward implementation of so-called stealth addresses.

Keywords: flexible public key, equivalence classes, stealth addresses, ring signatures, group signatures

1 Introduction

Digital signatures aim to achieve two security goals: Integrity of the signed message and authenticity of the signature. A great number of proposals relax these goals or introduce new ones to accommodate the requirements of specialized application scenarios. As one example, consider sanitizable signatures [1] where the goal of preserving the integrity of the message is relaxed to allow for authorized

modification and redaction of the signed message. This paper introduces a novel characterization of authenticity. The goal is not a complete relaxation, such that any impostor can sign messages on behalf of a legitimate signer, but rather that authenticity holds with respect to *some established legitimate signer*, but who it is exactly remains hidden. Achieving the latter without enabling the former is one of the main challenges we tackle in this paper.

Our new primitive, which we call *signatures with flexible public key* (SFPK) formalizes a signature scheme, where verification and signing keys live in a system of equivalence classes induced by a relation $\mathcal{R}$. Given a signing or verification key it is possible to transform the key into a different representative of the same equivalence class, i.e., the pair of old key and new key is contained in relation $\mathcal{R}$. Thus, we extend the requirement of unforgeability of signatures to the whole equivalence class of the given key under attack. However, an additional requirement we make is that it should be infeasible, without a trapdoor, to even check whether two keys are in the same class. This property, which we call computational *class-hiding*, ensures that given an old verification key, a signature under a fresh representative is indistinguishable from a signature under a different newly generated key, which lives in a different class altogether with overwhelming probability. Intuitively this means that signers can produce signatures for their whole class of keys, but they cannot sign for a different class (because of unforgeability) and they are able to hide which class the signature belongs to, i.e., to hide their own identity in the signature (because of class-hiding).

The property of class-hiding is especially useful in cases where there is a (possibly pre-defined) set of known verification keys and a verifier only needs to know that the originator of a given signature was part of that set. Indeed, upon reading the first description of the scheme's properties, what should come to mind immediately is the setting of group signatures [13] and to some extent ring signatures [29] where the group is chosen at signing time and considered a part of the signature. Our primitive yields highly efficient, cleanly constructed group and ring signature schemes, but it should be noted, that SFPK on its own is neither of the two.

The basic idea to build a group signature scheme from signatures with flexible public key is to combine them with an equally re-randomizable *certificate* on the signing key. Such a certificate is easily created through structure-preserving signatures on equivalence classes [23] by the group manager on the members' verification key. A group signature is then produced by signing the message under a fresh representative of the flexible public key and tying that signature to the group by also providing a blinded certificate corresponding to the fresh flexible key. This fresh certificate can be generated from the one provided by the group manager. Opening of group signatures is done using the trapdoor that can be used to distinguish if public keys belong to the same equivalence class. In the case of ring signatures, the certification of keys becomes slightly more complex, since we cannot make any assumption on the presence of a trusted group manager. Therefore, the membership certificate is realized through a perfectly sound proof

of membership. The basic principle, however, remains the same, pointing to an elegant, unified approach to both group and ring signatures.

Our contributions. This paper develops a new cryptographic building block from the ground up, presenting security definitions, concrete instantiations and applications. The main contributions are as follows:

Signatures with flexible public key. Our new primitive is a natural abstraction and formalization of design principles that are already at the heart of many ring and group signature constructions found in the literature. Thus it offers a unified perspective on these two primitives and aids in modular design of efficient constructions by making explicit properties which have to be achieved by the identity-hiding component as outlined above.

Generic constructions & tailored instantiations. We demonstrate how SFPK can be used to build group and ring signatures in a modularized fashion. For each construction, we give an efficient standard model SFPK instantiation which takes into account the differences in setting between group and ring signature. The resulting group and ring signature schemes have smaller (asymptotic and concrete) signature sizes than the previous state of the art schemes, including schemes with non-standard assumptions as long as one requires the strongest level of security.

For instance, the static group signature scheme due to Libert, Peters, and Yung achieves fully anonymous signatures secure under standard non-interactive assumptions at a size of 8448 bits per signature. Our scheme based on comparable assumptions achieves the same security using 7680 bits per signature. Another variant of our scheme under an interactive assumption achieves signature sizes of only 3072 bits per signature, thus more than halving the size achieved in [25] and not exceeding by more than factor 3 the size of signatures in the scheme due to Bichsel et al. [6] which produces signatures of size 1280 bits but only offers a weaker form of anonymity under an interactive assumption in the random oracle model. A comprehensive comparison between our scheme and known group signature constructions can be found in Section 5.4. Our ring signature construction is the first to achieve signature sizes in $\mathcal{O}(\sqrt{N})$ without trusted setup and security under standard assumptions in the strongest security model by Bender, Katz and Morselli [5]. Thereby, we settle an issue that was stated as an open problem in the ASIACRYPT'2017 presentation of [27].

Applications of independent interest. Constructions of signatures with flexible public key that allow for a straightforward key recovery property lend themselves to numerous stand-alone applications in the field of cryptocurrencies. We exemplify this by showing how to implement stealth addresses for Bitcoin [30, 28], which allow a party to transfer currency to an anonymous address that the sender has generated from the receivers long-term public key. No interaction with the receiver is necessary for this transaction and the receiver can recover and subsequently spend the funds without linking them to their long-term identity.

1.1 Further Related Work

At first glance, signatures with flexible public keys are syntactically reminiscent of structure-preserving signatures on equivalence classes[23]. While both primitives are similar in spirit, the former considers equivalence classes of key pairs while the latter only considers equivalence classes on messages.

Another related primitive are signatures with re-randomizable keys[16]. The crucial difference to our new primitive is that re-randomization is akin to and indeed indistinguishable from sampling a fresh key from the whole key space. This means a signature scheme with re-randomizable keys cannot achieve class hiding and unforgeability under flexible public keys simultaneously.

The ring signature built from signatures with flexible public keys is the first ring signature scheme to achieve signature size $O(\sqrt{N})$ where N is the size of the ring without any trusted setup. With a trusted setup, constant size constructions are known, the most recent one being [27] which is based on signatures with re-randomizable keys and SNARKs.

2 Preliminaries

We denote by $y \leftarrow \mathcal{A}(x, \omega)$ the execution of algorithm $\mathcal{A}$ outputting y, on input x with randomness ω, writing just $y \xleftarrow{\$} \mathcal{A}(x)$ if the specific randomness used is not important. We will sometimes omit the usage of random coin in the description of algorithms if it is obvious from the context (e.g. sampling group elements). The superscript $\mathcal{O}$ in $\mathcal{A}^{\mathcal{O}}$ means that algorithm $\mathcal{A}$ has access to oracle $\mathcal{O}$. Moreover, we say that $\mathcal{A}$ is probabilistic polynomial-time (PPT) if $\mathcal{A}$ uses internal random coins and the computation for any input $x \in \{0,1\}^*$ terminates in polynomial time. By $r \xleftarrow{\$} S$ 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 $\{1, \ldots, n\}$, $\boldsymbol{u}$ to denote a vector and $\left(x_0 \ldots x_{|x|}\right)_{\mathsf{bin}}$ to denote the binary representation of x.

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

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

Definition 2 (Bilinear-group generator). *A bilinear-group generator is a deterministic polynomial-time algorithm* BGGen *that on input a security parameter 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.*

Bilinear map groups with an efficient bilinear-group generator are known to be instantiable with ordinary elliptic curves introduced by Barreto and Naehrig [3] (in short BN-curves).

Invertible Sampling. We use a technique due to Damgård and Nielsen [15]:

- A standard sampler returns a group element X on input coins ω.
- A "trapdoor" sampler returns coins ω' on input a group element X.

Invertible sampling requires that (X, ω) and (X, ω') are indistinguishably distributed.

This technique was also used by Bender, Katz and Morselli [5] to prove full anonymity (where the adversary receives the random coins used by honest users to generate their keys) of their ring signature scheme.

2.1 Number Theoretical Assumptions

In this section we recall assumptions relevant to our schemes. They are stated relative to bilinear group parameters $\mathsf{BG} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow_{\$} \mathsf{BGGen}(\lambda)$.

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

Definition 4 (Square Decisional Diffie-Hellman Assumption in $\mathbb{G}_i$ [2]). *Given* BG *and elements* $g_i^a, g_i^z \in \mathbb{G}_i$ *it is hard for all PPT adversaries* $\mathcal{A}$ *to decide whether* $z = a^2 \mod p$ *or* $z \leftarrow_{\$} \mathbb{Z}_p^*$. *We will use* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sddh}}(\lambda)$ *to denote the advantage of the adversary in solving this problem.*

We now state the bilateral variant of the well known decisional linear assumption, where the problem instance is given in both $\mathbb{G}_1$ and $\mathbb{G}_2$. This definition was also used by Ghadafi, Smart and Warinschi [20].

Definition 5 (Symmetric Decisional Linear Assumption). *Given* BG*, elements* $f_1 = g_1^f, h_1 = g_1^h, f_1^a, h_1^b, g_1^z \in \mathbb{G}_1$ *and elements* $f_2 = g_2^f, h_2 = g_2^h, f_2^a, h_2^b, g_2^z \in \mathbb{G}_2$ *for uniformly random* $f, h, a, b \in \mathbb{Z}_p^*$ *it is hard for all PPT adversaries* $\mathcal{A}$ *to decide whether* $z = a + b \mod p$ *or* $z \leftarrow_{\$} \mathbb{Z}_p^*$. *We will use* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{linear}}(\lambda)$ *to denote the advantage of the adversary in solving this problem.*

In this paper we use a variant of the 1-Flexible Diffie-Hellman assumption [26]. We show that this new assumption, which we call the co-Flexible Diffie-Hellman (co-Flex) assumption, holds if the decisional linear assumption holds. We also introduce a similar assumption called square-Flexible Diffie-Hellman (sq-Flex).

Definition 6 (co-Flexible Diffie-Hellman Assumption). *Given* BG*, elements* $g_1^a, g_1^b, g_1^c, g_1^d \in \mathbb{G}_1$ *and* $g_2^a, g_2^b, g_2^c, g_2^d \in \mathbb{G}_2$ *for uniformly random* $a, b, c, d \in \mathbb{Z}_p^*$*, it is hard for all PPT adversaries* $\mathcal{A}$ *to output* $(g_1^c)^r, (g_1^d)^r, g_1^{r \cdot a \cdot b}$. *We will use* $\mathsf{Adv}_{\mathcal{A}}^{\text{co-flexdh}}(\lambda)$ *to denote the advantage of the adversary in solving this problem.*

Lemma 1. *The co-Flexible Diffie-Hellman assumption holds for* BG *if the decisional linear assumption holds for* BG.

Definition 7 (sq-Flexible Diffie-Hellman Assumption). *Given* BG*, elements* $g_1^a, g_1^b, g_1^c, g_1^d \in \mathbb{G}_1$ *and* $g_2^a, g_2^b, g_2^c, g_2^d \in \mathbb{G}_2$ *it is hard for all PPT adversaries* $\mathcal{A}$ *to output* $(g_1^c)^r, (g_1^d)^r, g_1^{r^2 \cdot a \cdot b}$*. We will use* $\mathsf{Adv}_{\mathcal{A}}^{\text{sq-flexdh}}(\lambda)$ *to denote the advantage of the adversary in solving this problem.*

Unfortunately, it is unknown whether for this assumption we can state a lemma similar to 1. However, under the Knowledge-of-Exponent (KEA) assumption [14], the sq-FlexDH and co-FlexDH assumptions are equivalent. This implies that the sq-FlexDH holds in the generic group model.

2.2 Non-Interactive Proof Systems

In this paper we make use of non-interactive proof systems. Although we define the proof system for arbitrarily languages, in our schemes we use the efficient Groth-Sahai (GS) proof system for pairing product equations [22]. Let $\mathcal{R}$ be an efficiently computable binary relation, where for $(x, w) \in \mathcal{R}$ we call x a statement and w a witness. Moreover, we will denote by $L_{\mathcal{R}}$ the language consisting of statements in $\mathcal{R}$, i.e. $L_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\}$.

Definition 8 (Non-Interactive Proof System). *A non-interactive proof system* Π *consists of the following three algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*:*

$\mathsf{Setup}(\lambda)$**:** *on input security parameter* λ*, this algorithm outputs a common reference string* ρ*.*

$\mathsf{Prove}(\rho, x, w)$**:** *on input common reference string* ρ*, statement* x *and witness* w*, this algorithm outputs a proof* π*.*

$\mathsf{Verify}(\rho, x, \pi)$**:** *on input common reference string* ρ*, statement* x *and proof* π*, this algorithm outputs either* $\texttt{accept}(1)$ *or* $\texttt{reject}(0)$*.*

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

Definition 9 (Soundness). *A proof system* Π *is called sound, if for all PPT algorithms* $\mathcal{A}$ *the following probability, denoted by* $\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{sound}}(\lambda)$*, is negligible in the security parameter* λ*:*

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

We say that the proof system is perfectly sound if $\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{sound}}(\lambda) = 0$*.*

Definition 10 (Witness Indistinguishability (WI)). *A proof system* Π *is witness indistinguishable, if for all PPT algorithms* $\mathcal{A}$ *we have that the advantage* $\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{wi}}(\lambda)$ *computed as:*

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

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

$\mathsf{Prove}(x, w)$

1 : $\rho_1 := (f_1, f_2, h_1, h_2, \ldots) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{PPE}}(\lambda); r, s \xleftarrow{\$} \mathbb{Z}_p^*$
2 : $\rho_2 := (f_1, f_2, h_1, h_2, f_1^r, f_2^r, h_1^s, h_2^s, g_1^{r+s}, g_2^{r+s})$
3 : $\pi_{\text{Linear}} \xleftarrow{\$} \mathsf{Prove}_{\text{Linear}}((\rho_1, \rho_2), (r, s))$
4 : $\pi_1 \xleftarrow{\$} \mathsf{Prove}_{\mathsf{PPE}}(\rho_1, x, w)$; $\pi_2 \xleftarrow{\$} \mathsf{Prove}_{\mathsf{PPE}}(\rho_2, x, w)$
5 : **return** $\pi := (\rho_1, \rho_2, \pi_{\text{Linear}}, \pi_1, \pi_2)$

$\mathsf{Verify}(x, \pi)$

1 : **parse** $\pi = (\rho_1, \rho_2, \pi_{\text{Linear}}, \pi_1, \pi_2)$
2 : **return** $\mathsf{Verify}_{\mathsf{PPE}}(\rho_1, x, \pi_1) = 1 \wedge$
3 : $\mathsf{Verify}_{\mathsf{PPE}}(\rho_2, x, \pi_2) = 1 \wedge$
4 : $\mathsf{Verify}_{\text{Linear}}((\rho_1, \rho_2), \pi_{\text{Linear}}) = 1$

Scheme 1: Perfectly Sound Proof System for Pairing Product Equations

Perfectly Sound Proof System for Pairing Product Equations We briefly recall the framework of pairing product equations that is used for the languages of the Groth-Sahai proof system [22]. For constants $A_i \in G_1$, $B_i \in G_2$, $t_T \in \mathbb{G}_T$, $\gamma_{ij} \in \mathbb{Z}_p$ which are either publicly known or part of the statement, and witnesses $X_i \in \mathbb{G}_1$, $Y_i \in \mathbb{G}_2$ given as commitments, we can prove that:

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

The system $(\mathsf{Setup}_{\mathsf{PPE}}, \mathsf{Prove}_{\mathsf{PPE}}, \mathsf{Verify}_{\mathsf{PPE}})$ has several instantiations based on different assumptions. In this paper we only consider the instantiation based on the symmetric linear assumption given by Ghadafi, Smart and Warinschi [20].

For soundness it must be ensured, that $\mathsf{Setup}_{\mathsf{PPE}}$ outputs a valid DLIN tuple. This can be enforced by requiring a trusted party performs the setup. However, in our schemes we require a proof system which is perfectly sound, even if a malicious prover executes the $\mathsf{Setup}_{\mathsf{PPE}}$ algorithm.

To achieve this we use the ideas by Groth, Ostrovsky and Sahai [21]. The authors propose a perfectly sound and perfectly witness indistinguishable proof system $(\mathsf{Prove}_{\text{Linear}}, \mathsf{Verify}_{\text{Linear}})$ which does not require a trusted setup. Using it one can show that given tuples T_1, T_2 as a statement, at least one of T_1 and T_2 is a DLIN tuple. The results were proposed for type 1 pairing but the proof itself is only given as elements in $\mathbb{G}_2$. Moreover, our variant of the DLIN assumption gives the elements in both groups. Thus, we can apply the same steps as in [21]. The cost of such a proof is 6 elements in $\mathbb{G}_2$.

Next is the observation that the tuples T_1 and T_2 can each be used as common reference strings for the pairing product equation proof system. Since at least one of the tuples is a valid DLIN tuple, at least one of the resulting proofs will be perfectly sound. Witness-indistinguishability will be only computational, since we have to provide T_1 and T_2 to the verifier but that is sufficient in our case. The full scheme is presented in Scheme 1. The size of the proofs produced this way is $2 \cdot (3 \cdot e + 3 \cdot w_1 + 5)$ elements in $\mathbb{G}_1$ and $2 \cdot (3 \cdot e + 3 \cdot w_2 + 5) + 6$ elements in $\mathbb{G}_2$, where e is the number of equations proven, w_1 is the number of witnesses in $\mathbb{G}_1$ and w_2 is the number of witnesses in $\mathbb{G}_2$.

Theorem 1. *Scheme 1 is a perfectly sound proof system for pairing product equations if the system* $(\mathsf{Setup}_{\mathsf{PPE}}, \mathsf{Prove}_{\mathsf{PPE}}, \mathsf{Verify}_{\mathsf{PPE}})$ *is perfectly sound in the common reference string model.*

Proof (Sketch). Because Π_{Linear} is perfectly sound $\mathsf{Verify}_{\text{Linear}}((\rho_1, \rho_2), \pi_{\text{Linear}}) = 1$ means that at least one of ρ_1 and ρ_2 is a DLIN tuple. It follows that at least one of π_1 and π_2 is a perfectly sound proof for the statement x. Thus, statement x must be true.

Theorem 2. *Scheme 1 is a computational witness indistinguishable proof system if the system* $(\mathsf{Setup}_{\mathsf{PPE}}, \mathsf{Prove}_{\mathsf{PPE}}, \mathsf{Verify}_{\mathsf{PPE}})$ *is perfectly witness indistinguishable in the common reference string model.*

Proof (Sketch). Because the proof system for the pairing product equations is witness indistinguishable, we change the witness we use in proof π_1. Note that this change may include the change of ρ_1 to a non-DLIN tuple but the proof π_{Linear} is still valid because ρ_2 is a DLIN tuple. Next we replace ρ_1 with ρ_2 and use $\mathsf{Setup}_{\mathsf{PPE}}$ to compute ρ_2. Finally, we change the witness used to compute π_2.

2.3 Structure-Preserving Signatures on Equivalence Classes

Hanser and Slamanig introduced a cryptographic primitive called structure-preserving signatures on equivalence classes [23]. Their work was further extended by Fuchsbauer, Hanser and Slamanig in [18] and [19]. The idea is simple but provides a powerful functionality. The signing $\mathsf{Sign}_{\mathsf{SPS}}(M, \mathsf{sk}_{\mathsf{SPS}})$ algorithm defines an equivalence relation $\mathcal{R}$ that induces a partition on the message space. By signing one representative of a partition, the signer in fact provides a signature for all elements in it. Moreover, there exists a procedure $\mathsf{ChgRep}_{\mathsf{SPS}}(M, \sigma_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}})$ that can be used to change the signature to a different representative without knowledge of the secret key. Existing instantiations allow to sign messages from the space $(\mathbb{G}_i^*)^\ell$, for $\ell > 1$, and for the following relation $\mathcal{R}_{exp}$: given two messages $M = (M_1, \ldots, M_\ell)$ and $M' = (M'_1, \ldots, M'_\ell)$, we say that M and M' are from the same equivalence class (denoted by $[M]_{\mathcal{R}}$) if there exists a scalar $r \in \mathbb{Z}_p^*$, such that $\forall_{i \in [\ell]} (M_i)^r = M'_i$.

Security Definition. We formally define structure-preserving signatures on equivalence classes as follows:

Definition 11 (Structure-preserving signatures for equivalence relation $\mathcal{R}$). *A* SPS-EQ *scheme on* $(\mathbb{G}_i^*)^\ell$ *(for* $i \in \{1, 2\}$*) consists of the following algorithms:*

$\mathsf{BGGen}_{\mathsf{SPS}}(\lambda)$**:** *a deterministic algorithm that on input a security parameters* λ*, outputs bilinear-group parameters* BG*.*

$\mathsf{KGen}_{\mathsf{SPS}}(\mathsf{BG}, \ell)$**:** *on input a parameter* BG *and a vector length* $\ell > 1$*, this probabilistic algorithm outputs a key pair* $(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}})$*.*

$\mathsf{Sign}_{\mathsf{SPS}}(M, \mathsf{sk}_{\mathsf{SPS}})$**:** *on input a message* $M \in (\mathbb{G}_i^*)^\ell$ *and secret key* $\mathsf{sk}_{\mathsf{SPS}}$*, this probabilistic algorithm outputs a signature* σ_{SPS} *on the equivalence class* $[M]_{\mathcal{R}}$*.*

$\mathsf{ChgRep}_{\mathsf{SPS}}(M, \sigma_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}})$**:** *on input a representative* M *of an equivalence class* $[M]_{\mathcal{R}}$*, signature* σ_{SPS} *for* M*, scalar* r *and a public key* $\mathsf{pk}_{\mathsf{SPS}}$*, this probabilistic algorithm returns an updated message-signature pair* $(M', \sigma'_{\mathsf{SPS}})$*, where* $M' = (M)^r$ *(component-wise exponentiation) is the new representative and* σ'_{SPS} *its updated signature.*

$\mathsf{Verify}_{\mathsf{SPS}}(M, \sigma_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}})$**:** *on input a representative* M*, signature* σ_{SPS} *and a public key* $\mathsf{pk}_{\mathsf{SPS}}$*, this deterministic algorithm outputs* 1 *if* σ_{SPS} *is a valid signature for* M *under public key* $\mathsf{pk}_{\mathsf{SPS}}$ *and* 0 *otherwise.*

$\mathsf{VKey}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}})$**:** *on input a secret key* $\mathsf{sk}_{\mathsf{SPS}}$ *and public key* $\mathsf{pk}_{\mathsf{SPS}}$*, this deterministic algorithm outputs* 1 *if both keys are consistent and* 0 *otherwise.*

The original paper defines two properties of SPS-EQ namely unforgeability under chosen-message attacks and class-hiding. Fuchsbauer and Gay [17] recently introduced a weaker version of unforgeability called unforgeability under chosen-open-message attacks, which restricts the adversaries' signing queries to messages where it knows all exponents.

Definition 12 (Signing Oracles). *A* signing oracle *is an* $\mathcal{O}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$ *(resp.* $\mathcal{O}_{\mathsf{op}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$*) oracle, which accepts messages* $(M_1, \ldots, M_\ell) \in (\mathbb{G}_i^*)^\ell$ *(resp. vectors* $(e_1, \ldots, e_\ell) \in (\mathbb{Z}_p^*)^\ell$*) and returns signature under* $\mathsf{sk}_{\mathsf{SPS}}$ *on those messages (resp. on messages* $(g_1^{e_1}, \ldots, g_1^{e_\ell}) \in (\mathbb{G}_i^*)^\ell$*).*

Definition 13 (EUF-CMA (resp. EUF-CoMA)). *A* SPS-EQ *scheme* $(\mathsf{BGGen}_{\mathsf{SPS}}, \mathsf{KGen}_{\mathsf{SPS}}, \mathsf{Sign}_{\mathsf{SPS}}, \mathsf{ChgRep}_{\mathsf{SPS}}, \mathsf{Verify}_{\mathsf{SPS}}, \mathsf{VKey}_{\mathsf{SPS}})$ *on* $(\mathbb{G}_i^*)^\ell$ *is called existentially unforgeable under chosen message attacks (resp. adaptive chosen-open-message attacks), if for all PPT algorithms* $\mathcal{A}$ *having access to an open signing oracle* $\mathcal{O}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$ *(resp.* $\mathcal{O}_{\mathsf{op}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$*) the following adversary's advantage (with templates* T_1, T_2 *defined below) is negligible in the security parameter* λ*:*

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

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

A stronger notion of class hiding, called perfect adaptation of signatures, was proposed by Fuchsbauer et al. in [19]. Informally, this definition states that signatures received by changing the representative of the class and new signatures for the representative are identically distributed. In our schemes we will only use this stronger notion.

Definition 14 (Perfect Adaption of Signatures). *A* SPS-EQ *scheme on* $(\mathbb{G}_i^*)^\ell$ *perfectly adapts signatures if for all* $(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}, M, \sigma, r)$*, where*

$\mathsf{VKey}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 1$, $M \in (\mathbb{G}_1^*)^\ell$, $r \in \mathbb{Z}_p^*$ *and* $\mathsf{Verify}_{\mathsf{SPS}}(M, \sigma, \mathsf{pk}_{\mathsf{SPS}}) = 1$, *the distribution of*

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

are identical.

3 Signatures with Flexible Public Key

We begin by introducing the idea behind our primitive. In the notion of existential unforgeability of digital signatures, the adversary must return a signature valid under the public key given to him by the challenger. Imagine now that we allow a more flexible forgery. The adversary can return a signature that is valid under a public key that is in some relation $\mathcal{R}$ to the public key chosen by the challenger. Similar to the message space of SPS-EQ signatures, this relation induces a system of equivalence classes on the set of possible public keys. A given public key, along with the corresponding secret key can be transformed to a different representative in the same class using an efficient, randomized algorithm. The adversary has access to this functionality by providing random coins which the challenger uses to change the representative before signing. Since there might be other ways of obtaining a new representative, the forgery on the challenge equivalence class is valid as long as the relation holds, even without knowledge of the explicit randomness that leads to the given transformation.

Note, that the challenger thus needs a way to ascertain whether the forgery is valid, which cannot be verification through the transformation algorithm. Indeed, for the full definition of our schemes' security we will require that it should not be feasible, in absence of the concrete transformation randomness, to determine whether a given public key belongs to one class or another. This property —called *class-hiding* in the style of a similar property for SPS-EQ signatures— should hold even for an adversary who has access to the randomness used to create the key pairs in question.

The apparent conflict is resolved by introducing a trapdoor key generation algorithm TKeyGen which outputs a key pair $(\mathsf{sk}, \mathsf{pk})$ and a class trapdoor τ for the class the key pair is in. The trapdoor allows the challenger to reveal whether a given key is in the same class as pk, even if doing so efficiently is otherwise assumed difficult. Since we require that the keys generated using the trapdoor key generation and the regular key generation are distributed identically, unforgeability results with respect to one also hold with respect to the other.

Definition 15 (Signature with Flexible Public Key). *A signature scheme with flexible public key* (SFPK) *is a tuple of* PPT *algorithms* (KeyGen, TKeyGen, Sign, ChkRep, ChgPK, ChgSK, Verify) *such that:*

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

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

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

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

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

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

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

A signature scheme with flexible public key is correct *if for all* $\lambda \in \mathbb{N}$*, all random coins* $\omega, r \in \mathsf{coin}$ *the following conditions hold:*

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

Definition 16 (Class-hiding). *For scheme* SFPK *with relation* $\mathcal{R}$ *and adversary* $\mathcal{A}$ *we define the following experiment:*

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

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

$(\mathsf{sk}_i, \mathsf{pk}_i) \xleftarrow{\$} \mathsf{KeyGen}(\lambda, \omega_i)$ *for* $i \in \{0,1\}$

$m \xleftarrow{\$} \mathcal{A}(\omega_0, \omega_1)$

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

$\mathsf{sk}' \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}_b, r); \mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}_b, r)$

$\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m)$

$\hat{b} \xleftarrow{\$} \mathcal{A}(\omega_0, \omega_1, m, \sigma, \mathsf{pk}')$

return $b = \hat{b}$

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

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

Definition 17 (Existential Unforgeability under Flexible Public Key). *For scheme* SFPK *with relation* $\mathcal{R}$ *and adversary* $\mathcal{A}$ *we define the following experiment:*

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

$\omega \xleftarrow{\$} \mathsf{coin}$
$(\mathsf{sk}, \mathsf{pk}, \tau) \xleftarrow{\$} \mathsf{TKeyGen}(\lambda, \omega); Q := \emptyset$
$(\mathsf{pk}', m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^1(\mathsf{sk},\cdot),\mathcal{O}^2(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk}, \tau)$
return $(m^*, \cdot) \notin Q \wedge$
$\mathsf{ChkRep}(\tau, \mathsf{pk}') = 1 \wedge$
$\mathsf{Verify}(\mathsf{pk}', m^*, \sigma^*) = 1$

$\underline{\mathcal{O}^1(\mathsf{sk}, m)}$

$\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}, m)$
$Q := Q \cup \{(m, \sigma)\}$
return σ

$\underline{\mathcal{O}^2(\mathsf{sk}, m, r)}$

$\mathsf{sk}' \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}, r)$
$\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m)$
$Q := Q \cup \{(m, \sigma)\}$
return σ

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

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

Definition 18 (Strong Existential Unforgeability under Flexible Public Key). *A* SFPK *is* strong existentially unforgeable with flexible public key under chosen message attacks *if for all* PPT *adversaries* $\mathcal{A}$ *the advantage* $\mathsf{Adv}^{\mathsf{seuf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$ *in the above experiment, where we replace the line* $(m^*, \cdot) \notin Q$ *with* $(m^*, \sigma^*) \notin Q$*, is negligible.*

Finally, we define an optional property of SFPK signature schemes called key recovery. In a standard application, the public key and secret key are randomized by the signer. Obviously, the ChgPK algorithm can be executed by any third party using random coins r, which can be later shared with the signer. This way the signer can compute the corresponding secret key. For some application we would like to work without any interaction. It is easy to see that allowing the user to extract the new secret key only using his old secret key would break class-hiding. Fortunately, we can use the additional trapdoor returned by the TKeyGen algorithm. More formally, we define this optional property as follows.

Definition 19 (Key Recovery Property). *A* SFPK *has* recoverable signing keys *if there exists an efficient algorithm* Recover *such that for all security parameters* $\lambda \in \mathbb{N}$*, random coins* ω, r *and all* $(\mathsf{sk}, \mathsf{pk}, \tau) \xleftarrow{\$} \mathsf{TKeyGen}(\lambda, \omega)$ *and* $\mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}, r)$ *we have* $\mathsf{ChgSK}(\mathsf{sk}, r) = \mathsf{Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$.

3.1 Flexible Public Key in the Multi-user Setting

In this subsection, we address applications where a part of the public key of the user is generated by some trusted third party and is common among several users, e.g. the definition of the hash function used in Waters signatures. We therefore define an additional algorithm CRSGen that, given a security parameter, outputs a common reference string ρ. We assume that this string is an implicit input to

all algorithms. If the KeyGen is independent from ρ, we say that such a scheme supports *key generation without setup*.

We will now discuss the implication of this new algorithm on the security definitions. Usually, we require that the common reference string is generated by an honest and trusted party (i.e. by the challenger in definitions 16 and 17). We additionally define those notions under maliciously generated ρ. We call a scheme *class-hiding under malicious reference string* if the class-hiding definition holds even if in definition 16 the adversary is allowed to generate the string ρ. Similarly, we call a SFPK scheme *unforgeable under malicious reference string* if the unforgeability definition 17 holds if ρ is generated by the adversary.

3.2 On Signatures with Re-Randomizable Keys

Fleischhacker et al. [16] introduced signatures with re-randomizable keys, which allow a re-randomization of signing and verification keys such that re-randomized keys share the same distribution as freshly generated keys and a signature signed under a randomized key can be verified using an analogously randomized verification key.

They also define a notion of unforgeability under re-randomized keys, which allows an adversary to learn signatures under the adversaries choice of randomization of the signing key under attack. The goal of the adversary is to output a forge under the original key or under one of its randomizations. Regular existential unforgeability for signature schemes is a special case of this notion, where the attacker does not make use of the re-randomization oracle.

The difference to signatures with flexible public keys is that re-randomization in [16] is akin to sampling a fresh key from the space of all public keys, while changing the representative in our case is restricted to the particular key's equivalence class. Note that one might intuitively think that signatures under re-randomizable keys are just signatures with flexible keys where there is only one class of keys and because re-randomizing is indistinguishable from fresh sampling. In this case class hiding would be perfect. However, such a scheme cannot achieve unforgeability under flexible keys, since it would be enough for an attacker to sample a fresh keypair and use a signature under that key as the forgery.

Another way of mapping signatures with re-randomizable keys to the flexible public key world would be to make the set of equivalence classes the singleton sets of all public keys, i.e. each key is the unique representative of its own equivalence class. This collapses unforgeability under flexible public keys to the standard unforgeability notion of digital signatures. In this case, however, class hiding would be impossible to achieve, since there is just one unique representative for each class. Note, that in the class-hiding definition, the challenge key pairs are not required to be in the same or separate classes. Therefore, even if both keys are from different classes, the property guaranties indistinguishability of those keys and corresponding signatures. It is easy to see, that in the above-mentioned situation, if keys would be from different classes, the adversary would always be able to distinguish between them.

Since we require a secure signature scheme with flexible public keys to achieve both class hiding and unforgeability under flexible public keys and any signature scheme with re-randomizable keys can achieve at most one of these properties the primitives are clearly seperable.

4 Applications

In this section we present natural applications of signatures with flexible public key. First we show how to implement cryptocurrency stealth addresses from signatures with flexible public key which have the additional key recovery property.

Then follow generic constructions of group and ring signature schemes. As we will see in Section 5, each of the schemes presented in this section can be instantiated with a signature scheme with flexible public key such that the result improves on the respective state-of-the-art in terms of concrete efficiency, necessary assumptions or both.

4.1 Cryptocurrency Stealth Addresses

A direct application of signatures with flexible public keys in the cryptocurrency domain is the implementation of *stealth addresses* [30]. In cryptocurrency systems such as Bitcoin, transactions are digitally signed, such that the original owner of the funds signs the transaction to transfer funds to another party. In this transaction, the receiving party is also identified by its public key. Using stealth addresses, it is possible for the sender to create a fresh public key for the receiving party from their known public key such that these two keys cannot be linked. The receiving party can recognize the fresh key as its own and generate a corresponding private key, which subsequently enables it to spend any fund send to the fresh unlinkable key. Crucially, there is no interaction necessary between sender and receiver to establish the fresh key and only the legitimate receiver can recover the right secret key corresponding to the fresh key.

This can be implemented via a straightforward augmentation of signatures with flexible public keys by a signing key recovery algorithm which allows the holder of the signing key to recover an equivalent signing key from the trapdoor and the fresh public key alone. Scheme 4 is an instances of signatures with flexible public key which achieve this property. We also show how to extend schemes 5 and 6 to support it.

4.2 Group Signatures/Self-blindable Certificates

We now present an efficient generic construction of static group signatures that uses SFPK as a building block and which is secure in the model by Bellare, Micciancio and Warinschi [4]. The idea is to generate a SFPK secret/public key pair and "certify" the public part with a SPS-EQ signature. To sign a message, the signer changes the representation of its SFPK key, and changes the representation of the SPS-EQ certificate. The resulting signature is the SFPK signature, the randomized public key and the SPS-EQ certificate.

$\mathsf{KeyGen}_{\mathsf{GS}}(1^\lambda, n)$

1 : $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}_{\mathsf{SPS}}(1^\lambda); (\mathsf{pk}_{\mathsf{SPS}}, \mathsf{sk}_{\mathsf{SPS}}) \xleftarrow{\$} \mathsf{KGen}_{\mathsf{SPS}}(\mathsf{BG}, \ell)$
2 : $\rho \xleftarrow{\$} \mathsf{CRSGen}(1^\lambda)$ // *optional*
3 : **foreach** user $i \in [n]$:
4 : $(\mathsf{pk}^i, \mathsf{sk}^i, \tau^i) \xleftarrow{\$} \mathsf{TKeyGen}(1^\lambda, \omega)$
5 : $\sigma^i_{\mathsf{SPS}} \xleftarrow{\$} \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}^i, \mathsf{sk}_{\mathsf{SPS}})$
6 : **return** $(\mathsf{gpk} := (\mathsf{BG}, \mathsf{pk}_{\mathsf{SPS}}, \rho), \mathsf{gmsk} := ([(\tau^i, \mathsf{pk}^i)]_{i=1}^n),$
7 : $\mathsf{gski} := (\mathsf{pk}^i, \mathsf{sk}^i, \sigma^i_{\mathsf{SPS}}))$

$\mathsf{Sign}_{\mathsf{GS}}(\mathsf{gski}, m)$

1 : **parse** $\mathsf{gski} = (\mathsf{pk}, \mathsf{sk}, \sigma_{\mathsf{SPS}})$
2 : $r \xleftarrow{\$} \mathbb{Z}_p^*; \mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{pk}, r); \mathsf{sk}' \leftarrow \mathsf{ChgSK}(\mathsf{sk}, r)$
3 : $(\mathsf{pk}', \sigma'_{\mathsf{SPS}}) \leftarrow \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}, \sigma_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}})$
4 : $M := m || \sigma'_{\mathsf{SPS}} || \mathsf{pk}'$
5 : $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', M)$
6 : **return** $\sigma_{\mathsf{GS}} := (\mathsf{pk}', \sigma, \sigma'_{\mathsf{SPS}})$

Scheme 2: Generic Group Signature Scheme

Opening of signatures work as follows. The group manager generates the SFPK keys with a trapdoor (using TKeyGen) and keeps it along in a list. Note that this means that the group manager's secret key depends linearly on the size of the group. In order to open a signature the manager uses the stored trapdoor to run the ChkRep algorithm thereby determining the equivalence class of the group signature's public key. The group manager can also generate the common reference string $\rho \xleftarrow{\$} \mathsf{CRSGen}$ for the SFPK signatures and use it as part of the group public key. This allows us to use schemes which are secure in the multi-user setting, e.g. Scheme 5.

Due to space limitations, we only present the setup and signing algorithm for Scheme 2. Verification and opening procedures should be clear from the context.

Remark 1 (Self-blindable Certificates). If we use the KeyGen algorithm instead of TKeyGen to compute the SFPK key pair, then there exists no efficient opening procedure and the combination of SFPK and SPS-EQ signature scheme yields a self-blindable certificate scheme [31].

Theorem 3. *Scheme 2 is a correct static group signature scheme.*

Proof. Let $\lambda, n \in \mathbb{N}$ and let the output of $\mathsf{KeyGen}_{\mathsf{GS}}(1^\lambda, n)$ be

$$\begin{aligned}(\mathsf{gpk} &= (\mathsf{BG}, \mathsf{pk}_{\mathsf{SPS}}, \rho),\\ \mathsf{gmsk} &= ([(\tau^i, \mathsf{pk}^i)]_{i=1}^n),\\ \mathsf{gski} &= (\mathsf{pk}^i, \mathsf{sk}^i, \sigma^i_{\mathsf{SPS}}))\end{aligned}$$

Let $i \in [n]$ and m a message, then $\mathsf{Sign}_{\mathsf{GS}}(\mathsf{gski}, m)$ will output $(\mathsf{pk}', \sigma, \sigma'_{\mathsf{SPS}})$ where $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{pk}, r)$, $(\mathsf{pk}', \sigma'_{\mathsf{SPS}}) \leftarrow \mathsf{ChgRep}_{\mathsf{SPS}}(\mathsf{pk}, \sigma_{\mathsf{SPS}}, r, \mathsf{pk}_{\mathsf{SPS}})$ and $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', M)$. Since the relation $\mathcal{R}_{Flex}$ is the same for SFPK and the SPS-EQ scheme, $\mathsf{ChgRep}_{\mathsf{SPS}}$ and ChgPK will output the same pk' and because of the correctness of SPS-EQ, $\mathsf{Verify}_{\mathsf{SPS}}(\mathsf{pk}', \sigma'_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}})$ will succeed. Similarly $\mathsf{Verify}(\mathsf{pk}', m, \sigma)$ will succeed because of the correctness of the SFPK scheme. Hence verification will succeed. Since the signature was honestly generated, we have $\mathsf{pk}' \in [\mathsf{pk}^i]_{\mathcal{R}}$, which will be detected by the group manager in the opening procedure by trying all possible trapdoors in the group manager secret key.

Theorem 4. *Scheme 2 is fully traceable if the* SPS-EQ *signature scheme is existential unforgeable under chosen-message attacks and the* SFPK *scheme is existential unforgeable.*

Proof (Theorem 4). We will use the game base approach. Let us denote by S_i the event that the adversary wins the full traceability experiment in $\mathbf{GAME}_i$. Let $(m^*, \sigma^*_{\mathsf{GS}} = (\mathsf{pk}^*, \sigma^*, \sigma^*_{\mathsf{SPS}}))$ be the forgery outputted by the adversary.

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

$\mathbf{GAME}_1$: We abort in case $\mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk}, m^*, \sigma^*_{\mathsf{GS}}) = \perp$ but $\mathsf{Verify}_{\mathsf{GS}}(\mathsf{gpk}, m^*, \sigma^*_{\mathsf{GS}}) = 1$. Informally, we exclude the case that the adversary creates a new user from outside the group, i.e. a new SPS-EQ signature.

We will show that this only decreases the adversary's advantage by a negligible fraction. In particular, we will show that any adversary $\mathcal{A}$ returns a forgery for which we abort, can be used to break the existential unforgeability of the SPS-EQ signature scheme. The reduction algorithm uses the signing oracle to compute all signature σ^i_{SPS} of honest users. Finally, if the adversary returns $(m^*, \sigma^*_{\mathsf{GS}} = (\mathsf{pk}^*, \sigma^*, \sigma^*_{\mathsf{SPS}}))$, the reduction algorithm returns $(\mathsf{pk}^*, \sigma^*_{\mathsf{SPS}})$ as a valid forgery. We note that by correctness of the SFPK scheme, if pk^* is in a relation to a public key of an honest user, then we can always open this signature. It follows that pk^* is from a different equivalence class and the values returned by the reduction algorithm are a valid forgery against the SPS-EQ signature scheme.

It follows that $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{\ell,\mathsf{euf\text{-}cma}}_{\mathsf{SPS\text{-}EQ},\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_2$: We choose a random user identifier $j \xleftarrow{\$} [n]$ and abort in case $\mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk}, m^*, \sigma^*_{\mathsf{GS}}) \neq j$

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

We now show that any adversary $\mathcal{A}$ that has non-negligible advantage in winning full-traceability experiment in $\mathbf{GAME}_2$ can be used by a reduction algorithm $\mathcal{R}$ to break the existential unforgeability of the SFPK scheme.

$\mathcal{R}$ computes all the public keys of group members according to protocol, except for user j. For this user, the algorithm sets pk^j to the public key given to $\mathcal{R}$ by the challenger in the unforgeability experiment of the SFPK scheme. It is worth noting, that the adversary $\mathcal{A}$ is given the group manager's secret key $\mathsf{gmsk} = ([(\tau^i, \mathsf{pk}^i)]^n_{i=1})$. Fortunately, the reduction $\mathcal{R}$ is also given τ^j by the challenger and can compute a valid secret key gmsk that it gives as input to $\mathcal{A}$. To simulate signing queries for the j-th user, $\mathcal{R}$ uses its own signing oracle. By the change made in $\mathbf{GAME}_2$, $\mathcal{A}$ will never ask for the secret key of the j-th user, for which $\mathcal{R}$ is unable to answer (unlike for the other users).

Finally, $\mathcal{A}$ outputs a valid group signature $(m^*, \sigma^*_{\mathsf{GS}} = (\mathsf{pk}^*, \sigma^*, \sigma^*_{\mathsf{SPS}}))$ and the reduction algorithm outputs $(m^* || \sigma^*_{\mathsf{SPS}} || \mathsf{pk}^*, \sigma^*)$ as a valid SFPK forgery. By

the changes made in the previous games we know that pk^* and pk^j must be in a relation. Moreover, the message m^* could not be used by $\mathcal{A}$ in any signing query made to $\mathcal{R}$. Thus we know that $(m^*||\sigma^*_{\mathsf{SPS}}||\mathsf{pk}^*)$ was never queried by $\mathcal{R}$ to its signing oracle, which show that $\mathcal{R}$ returns a valid forgery against the unforgeability of the SFPK scheme.

Finally, we have

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

Theorem 5. *Scheme 2 is fully anonymous if the* SPS-EQ *signature scheme perfectly adapts signatures and is existential unforgeable under chosen-message attacks, the* SFPK *scheme is class-hiding and strongly existential unforgeable.*

Proof (Theorem 5). We will use the game-based approach. Let us denote by S_i the event that the adversary wins the full anonymity experiment in $\mathbf{GAME}_i$.

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

$\mathbf{GAME}_1$: In this game we change the way we compute the challenge signature $\sigma^*_{\mathsf{GS}} \xleftarrow{\$} \mathsf{Sign}_{\mathsf{GS}}(\mathsf{gsk}[i_b], m^*)$. Let $\sigma^*_{\mathsf{GS}} = (\mathsf{pk}', \sigma, \sigma'_{\mathsf{SPS}})$. We compute (pk', σ) as in the original experiment but instead of randomizing the SPS-EQ signature σ_{SPS}, we compute $\sigma_{\mathsf{SPS}} \leftarrow \mathsf{Sign}_{\mathsf{SPS}}(\mathsf{pk}', \mathsf{sk}_{\mathsf{SPS}})$.

Because the SPS-EQ signature scheme perfectly adapts signatures, we have $\Pr[S_1] = \Pr[S_0]$.

$\mathbf{GAME}_2$: We pick a random user identifier $j \xleftarrow{\$} [n]$ and abort in case $j \neq i_b$.

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

$\mathbf{GAME}_3$: We now abort in case the adversary queries a valid signature $(m, \sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma, \sigma'_{\mathsf{SPS}}))$ to the $\mathsf{Open}_{\mathsf{GS}}$ oracle and it fails to open, i.e. the opening algorithm returns $\perp$.

By perfect correctness of the SFPK scheme, it follows that the only way an adversary can make the experiment abort if he is able to create a new user, i.e. create a valid SPS-EQ signature under a public key pk^* that is not in relation with any of the honest public keys. It follows that we can use such an adversary to break the existential unforgeability of the SPS-EQ signature scheme, i.e. we just use the signing oracle to generate all σ^i_{SPS} and return $(\mathsf{pk}', \sigma'_{\mathsf{SPS}})$ as a valid SPS-EQ forgery.

It follows that $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\ell,\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{SPS\text{-}EQ}}(\lambda)$.

$\mathbf{GAME}_4$: We now change the way, we compute the secret key for user j. Instead of using $(\mathsf{pk}^j, \mathsf{sk}^j, \tau^j) \leftarrow \mathsf{TKeyGen}_{\mathsf{FW}}(\lambda, \omega)$, we use $(\mathsf{pk}^j, \mathsf{sk}^j) \leftarrow \mathsf{KeyGen}(\lambda, \omega)$.

Obviously, in such a case we cannot answer the $\mathsf{Open}_{\mathsf{GS}}$ queries for user j, as the value τ^j is unknown. However, we note that if the adversary's query $(m, \sigma_{\mathsf{GS}})$ is a valid group signature, then the $\mathsf{Open}_{\mathsf{GS}}$ must return a valid user identifier (because of the change in $\mathbf{GAME}_3$, we do not return $\perp$ in such a case). Therefore, if there exists no identifier $i \in [n]/\{j\}$ for which $\mathsf{ChkRep}(\tau^i, \mathsf{pk}^i, \mathsf{pk}') = 1$, we return j.

It is easy to note that this is just a conceptual change (because of the change in $\mathbf{GAME}_3$) and we have $\Pr[S_4] = \Pr[S_3]$.

$\mathbf{GAME}_5$: We now compute a random SFPK key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda, \omega)$, choose a random blinding factor r, compute public key $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{pk}, r)$, secret key $\mathsf{sk}' \leftarrow \mathsf{ChgSK}(\mathsf{sk}, r)$ and change the way we compute the challenged signature $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma, \sigma_{\mathsf{SPS}})$ under message m. We set $M = m||\sigma_{\mathsf{SPS}}||\mathsf{pk}'$ and run $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}', M)$. In other words, instead of using the secret of user i_b to generate the signature σ, we use a fresh key pair for this (i.e. a user from outside the system).

We note that any adversary that is able to distinguish between $\mathbf{GAME}_4$ and $\mathbf{GAME}_5$, can be used to break the class-hiding property of the SFPK signature scheme. The reduction algorithm can just set one of the public keys from the class-hiding challenge to be part of the public key of the j-th user. In case, the signature given by the challenger in the class-hiding game was created by this user, we are in $\mathbf{GAME}_4$. If it was created by the second user, then we are in $\mathbf{GAME}_5$. Of course, it might happen that the one of the users in the other group member (other than the j-th user) has a public key from the same relation as the second user in the class-hiding experiment. However, this event occurs with negligible probability and we omit it.

Lastly, we notice that the challenger in the class-hiding experiment is given the random coins used to generate the secret key to the adversary. Thus, our reduction can reuse those coins and compute the secret key, which he can give to the distinguishing algorithm, as required to fully simulate the anonymity experiment.

It follows that $|\Pr[S_5] - \Pr[S_4]| \leq \mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$.

The above changes ensure that the challenged signature is independent from the user i_b, i.e. we use a random SFPK public key and a freshly generated $\mathsf{SPS\text{-}EQ}$ signature on it. However, an adversary $\mathcal{A}$ can still use the way we implemented the $\mathsf{Open}_{\mathsf{GS}}$ in $\mathbf{GAME}_4$. Note that in case, he is somehow able to randomize the signature $\sigma_{\mathsf{GS}} = (\mathsf{pk}, \sigma, \sigma_{\mathsf{SPS}})$ and ask the $\mathsf{Open}_{\mathsf{GS}}$ oracle, then we will return i_b as the answer.

We will now show that the adversary cannot create a valid and distinct signature from $\sigma_{\mathsf{GS}} = (\mathsf{pk}, \sigma, \sigma_{\mathsf{SPS}})$. Let $(m^*, \sigma^*_{\mathsf{GS}} = (\mathsf{pk}^*, \sigma^*, \sigma^*_{\mathsf{SPS}}))$ be the query made by the adversary and σ^*_{GS} is a randomized version of σ_{GS}.

The first observation is that by the change made in $\mathbf{GAME}_5$, we must have that pk and pk^* are in a relation, otherwise the above attack does not work. Thus, we can use such an adversary to break the strong existential unforgeability of the SFPK signature scheme. Note that by the change made in $\mathbf{GAME}_5$, pk is a fresh public key and the reduction algorithm can use the one from the strong existential uforgeability game. Moreover, in order to generate σ, the reduction algorithm uses its signing oracle. Finally, the reduction algorithm returns $((m^*||\sigma^*_{\mathsf{SPS}}||\mathsf{pk}^*), \sigma^*)$ as a valid forgery.

It is easy to see that in case $\mathsf{pk} \neq \mathsf{pk}^*$ or $\sigma_{\mathsf{SPS}} \neq \sigma^*_{\mathsf{SPS}}$, the reduction algorithm wins the strong existential unforgeability game. Thus, the only part of the group signature that the adversary could potentially change is σ. This is the SFPK signature and would mean that the adversary was able to create a new signature under the message asked by the reduction algorithm to the signing algorithm. However, the case that $\sigma \neq \sigma^*$ also means that the reduction algorithm breaks the strong existential unforgeability of the SFPK scheme. We conclude, $\Pr[S_5] = \mathsf{Adv}^{\mathsf{seuf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$.

Finally, we have

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

4.3 Ring Signatures

In ring signatures there is no trusted entity such as a group manager and groups are chosen ad hoc by the signers themselves. Thus, we cannot use $\mathsf{SPS\text{-}EQ}$ to prove that a given ring signature was created by a valid member. Instead we give a membership proof, which is perfectly sound even if the common reference string is generated by the signer. In other words, the actual ring signature is a SFPK signature (pk', σ) and a proof Π that there exists a public key $\mathsf{pk} \in \mathtt{Ring}$ that is in relation to the public key pk', i.e. the signer proves knowledge of the random coins used to get pk'. The signature's anonymity relies on the class-hiding property of SFPK. Unfortunately, in the proof, the reduction does not know a valid witness for proof Π, since it does not choose the random coins for the challenged signature. Thus, we introduce a trapdoor witness. We extend the signer's public keys by a tuple of three group elements (A, B, C) and prove an OR statement which allows the reduction to compute a valid proof Π if (A, B, C) is a non-DDH tuple. More details are given in Scheme 3.

We can instantiate this scheme with a membership proof based on the $O(\sqrt{n})$ size ring signatures by Chandran, Groth, Sahai [11] and the perfectly sound proof system for NP languages by Groth, Ostrovsky, Sahai [21]. The resulting membership proof is perfectly sound and of sub-linear size in the size of the set. It follows, that our ring signature construction yields the first sub-linear

$\mathsf{RKeyGen}(1^\lambda)$

1 : $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(\lambda, \omega)$
2 : $I := (A, B, C) \xleftarrow{\$} \mathbb{G}_1^3$
3 : **return** $(\mathsf{pk}_{\mathsf{RS}} := (\mathsf{pk}, I), \mathsf{sk}_{\mathsf{RS}} := \mathsf{sk})$

$\mathsf{RVerify}(m, \Sigma, \mathtt{Ring})$

1 : **parse** $\Sigma = (\mathsf{pk}', \sigma, \Pi, \rho_\Pi)$
2 : **return** $\mathsf{Verify}(x, \Pi) \wedge$
3 : $\mathsf{Verify}(\mathsf{pk}', m, \sigma)$

$\mathsf{RSign}(m, \mathsf{sk}_{\mathsf{RS}}, \mathtt{Ring})$

1 : $r \xleftarrow{\$} \mathbb{Z}_p^*;\ \mathsf{sk}' \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}, r);\ \mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}, r)$
2 : $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m||\mathtt{Ring})$
3 : $\Pi \xleftarrow{\$}$ $\mathsf{Prove}(x, (\mathsf{pk}, r))$ where x is statement
$\exists_{\mathsf{pk}, r} \left((i, \mathsf{pk}, \cdot) \in \mathtt{Ring} \ \wedge\ \mathsf{ChgPK}(\mathsf{pk}, r) = \mathsf{pk}'\right) \ \vee$
$\exists_I \left((i, \cdot, I) \in \mathtt{Ring} \ \wedge\ I \text{ is not a DDH tuple}\right)$
4 : **return** $\Sigma := (\mathsf{pk}', \sigma, \Pi)$

Scheme 3: Generic Ring Signature Scheme

ring signature from standard assumptions without a trusted setup. The usage of a generic proof system makes the scheme inefficient in practice. Therefore, we instantiate our signature with flexible public key in a way such that we can use the same idea as in [11] but with the perfectly sound proof system for pairing product equations presented in Subsection 2.2. This SFPK instantiation is given in Scheme 7.

Theorem 6. *Scheme 3 is a correct ring signature scheme.*

Proof. Let $\lambda \in \mathbb{N}, n = \mathsf{poly}(\lambda)$, and

$$\left\{(\mathsf{sk}_{\mathsf{RS}}^{(i)} = \mathsf{sk}^{(i)}, \mathsf{pk}_{\mathsf{RS}}^{(i)} = (\mathsf{pk}^{(i)}, A^{(i)}, B^{(i)}, C^{(i)}))\right\}_{i=1}^{n}$$

a set of key pairs generated by $\mathsf{RKeyGen}(1^\lambda)$, and $s \in \{1, \ldots, n\}$ as well as m a message.

The signature of (s) on m will be of the form $(\mathsf{pk}', \sigma, \Pi)$ where $\mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}, r)$ and $\sigma \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}, r)$ for some random r and $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m||\mathtt{Ring})$ and Pi a proof of the given statement with witness pk, r. Because of the completeness of the proof system verification of Π will succeed and because of the correctness of the SFPK scheme, signature σ will also be valid. Thus $\mathsf{RVerify}(m, \Sigma, \mathtt{Ring}) = \mathtt{accept}$.

Theorem 7. *The generic construction of ring signatures presented in scheme 3 is unforgeable w.r.t. insider corruption assuming the* SFPK *scheme is existential unforgeable, the proof system used is perfectly sound and the decisional Diffie-Hellman assumption holds.*

Proof (Theorem 7). We will use the game based approach to prove this theorem. The first change we do is to fix the instance I to be a DDH tuple. This way our

reduction algorithm (as well as the adversary) must use a witness that fulfils the first part of the statement proven by Π. The next step is simple. The reduction algorithm translates this game to the existential unforgeability experiment of the SFPK scheme. Note that the reduction algorithm will choose one of the users at random and use the challenged public key as the user's public key. For the other users, the reduction algorithm will use a randomly choose key pair. This allows the reduction to answer all corruption queries. More formally. Let us denote by S_i the event that the adversary wins the unforgeability w.r.t insider corruption experiment in $\mathbf{GAME}_i$.

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

$\mathbf{GAME}_1$: We make a small change in the way we generate the instance I for the public keys of users. Instead of generating A, B, C as random elements of $\mathbb{G}_1$, we first chose $a, b \leftarrow_{\$} \mathbb{Z}_p^*$ and then set $A = g_1^a$, $B = g_1^b$ and $C = g^{a \cdot b}$.

It is obvious that this change only decreases the adversary's advantage by a negligible fraction. In particular any distinguishing adversary can be used to break the decisional Diffie-Hellman assumption. Moreover, note that since any DDH instance can be randomized (i.e. (A^r, B^r, C^r) is a DDH tuple if and only if (A, B, C) is a DDH tuple) we can apply this change to all honest users at once. Thus, we get $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

We now show how to use any adversary $\mathcal{A}$ that has non-negligible advantage in winning the unforgeability w.r.t insider corruption experiment in $\mathbf{GAME}_1$ to create a reduction algorithm $\mathcal{R}$ that has non-negligible advantage in winning the existential unforgeability experiment of the SFPK scheme. Let us by l denote the total number of users in the unforgeability w.r.t insider corruption experiment. The reduction algorithm works as follows.

In the first step $\mathcal{R}$ chooses a random $j \xleftarrow{\$} [l]$ and generates $(\mathrm{SK}_i, \mathrm{PK}_i) \leftarrow \mathsf{RKeyGen}(\rho, \omega_i)$ for all $i \in [l]/\{j\}$. For the j-th user it uses the public key $\mathrm{PK}_j = \mathsf{pk}_j$ given to him by the challenger in the existential unforgeability experiment for the SFPK scheme for relation $\mathcal{R}_{exp}$. $\mathcal{R}$ is able to answer all corruption queries of $\mathcal{A}$, beside for the j-th user. However, we hope that the adversary chooses this user to be part of the ring $\mathtt{Ring}^*$ for which he has to output a forgery. In such a case the adversary cannot ask the corruption query for the secret key of this user. We will later calculate the corresponding probability of the adversary asking for the j-th user's key but now we assume that in such a case the reduction $\mathcal{R}$ aborts. The reduction algorithm is also able to answer all signing queries. Note that for the j-th user instead of using the RSign algorithm, we choose a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ and query the signing oracle $\mathcal{O}^2$ with input (m, r).

Finally, the adversary $\mathcal{A}$ outputs a ring signature $\Sigma^* = (\mathsf{pk}^*, \sigma^*, \Pi^*, \rho^*_\Pi)$ under message m^* for ring $\mathtt{Ring}^*$. The reduction returns (m^*, Σ^*) as its forgery for the SFPK scheme. We will now calculate the success probability of $\mathcal{R}$. We first notice that by the change made in $\mathbf{GAME}_1$ and since the proof Π^* is

perfectly sound, it follows that there exists a public key $\mathsf{pk} \in \mathtt{Ring}^*$ for which $(\mathsf{pk}, \mathsf{pk}^*) \in \mathcal{R}_{exp}$. Finally we have that the probability that $\mathsf{pk} = \mathsf{pk}_j$ is $1/l$, i.e. from the j-th user's public key. Note that in such a case the adversary will not ask for the j-th user public key.

It follows that

$$\Pr[S_1] \leq l \cdot \mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}_{Flex}}(\lambda), \text{ and}$$
$$\Pr[S_0] \leq l \cdot \mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SFPK},\mathcal{R}_{Flex}}(\lambda) + \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda).$$

Theorem 8. *The generic construction of ring signatures presented in Scheme 3 is anonymous against full key exposure assuming the* SFPK *scheme is class-hiding and the used proof system is computationally witness-indistinguishable.*

Proof (Theorem 8). Let us denote by S_i the event that the adversary wins the anonymity experiment in $\mathbf{GAME}_i$.

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

$\mathbf{GAME}_1$: We make a small change we compute the instance $I = (A, B, C)$ in all the public keys of users. Instead of choosing A, B, C at random from $\mathbb{G}_1$, we first choose $a, b \xleftarrow{\$} \mathbb{Z}_p^*$ and then compute $A = g_1^a$, $B = g_1^b$, $C = g_1^{a \cdot b - 1}$. In other words, we make sure that I is not a DDH tuple.

Similar as in the proof for unforgeability, we have $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_2$: We now change the witness that we use to compute the proof Π in the challenged signature Σ. Instead of using the public key pk_{i_b}, we will use a witness for the second part of the statement. Note that by the change made in the previous game, all instances I in the public keys of honest users are non-DDH tuples. Moreover, instead of using the witness for the instance I_{i_b} (where b is the challenged bit b and i_b is the identifier of the user for which the experiment generates the signature), we will choose a random bit $\hat{b}$ and use the witness for instance $I_{i_{\hat{b}}}$. Note that the proof inside the signature Σ is now valid and independent of the bit b.

Because the proof system is computational witness-indistinguishable, it follows that $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_3$: We will now change the way we compute the signature $\Sigma = (\mathsf{pk}', \sigma', \Pi, \rho_\Pi)$. In particular we will change the way we compute pk' and σ'. Instead of computing it them using

$$\mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}_{i_b}, r),$$
$$\mathsf{sk}' \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}_{i_b}, r),$$
$$\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m || \mathtt{Ring}),$$

we will choose a fresh random bit $\hat{b}$ and compute it as

$$\begin{aligned}\mathsf{pk}' &\xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}_{i_{\hat{b}}}, r),\\ \mathsf{sk}' &\xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}_{i_{\hat{b}}}, r),\\ \sigma &\xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m||\mathtt{Ring}).\end{aligned}$$

We now show that any adversary $\mathcal{A}$ that has non-negligible advantage in distinguishing the difference between games 2 and 3, can be used as part of a reduction algorithm $\mathcal{R}$ that breaks the class-hiding property of the SFPK scheme. Let us by l denote the total number of users in the anonymity experiment. The reduction first chooses $j, k \xleftarrow{\$} [l]$ and generates $(\mathrm{SK}_i, \mathrm{PK}_i) \leftarrow \mathsf{RKeyGen}(\rho, \omega_i)$ for all $i \in [l]/\{j,k\}$. Let (ω_0^*, ω_1^*) be the random coins given to $\mathcal{A}$ by the class-hiding challenger. The reduction $\mathcal{R}$ runs $(\mathsf{sk}_0, \mathsf{pk}_0) \xleftarrow{\$} \mathsf{KeyGen}(\lambda, \omega_0^*)$ and $(\mathsf{sk}_1, \mathsf{pk}_1) \xleftarrow{\$} \mathsf{KeyGen}(\lambda, \omega_1^*)$. Then it computes random (A_0, B_0, C_0) and (A_1, B_1, C_1) as in $\mathbf{GAME}_1$ and the corresponding random coins ω_{I_0} and ω_{I_1}. It then sets $\omega_i = (\omega_0^*, \omega_{I_0})$, $\omega_k = (\omega_1^*, \omega_{I_1})$ and gives $\{\omega_i\}_{i=1}^{l}$ to $\mathcal{A}$. The adversary now outputs $(m, i_0, i_1, \mathtt{Ring})$. The reduction $\mathcal{R}$ aborts in case $i_0, i_1 \notin \{j,k\}$. Note that since, $\mathcal{A}$ advantage is non-negligible, we have that $i_0 \neq i_1$, $i_0 \in \mathtt{Ring}$ and $i_1 \in \mathtt{Ring}$. $\mathcal{R}$ then forwards $m||\mathtt{Ring}$ to the class-hiding challenger and receives a SFPK signature σ' under the randomized public key pk'. The reduction computes the ring signature as $\Sigma = (\mathsf{pk}', \sigma', \Pi, \rho_\Pi)$, where Π is a proof computed as in $\mathbf{GAME}_2$. Obviously, the success of $\mathcal{R}$ depends on the probability of guessing the correct identifiers i_0 and i_1. The probability is greater than $\frac{2}{l^2}$.

It follows that $|\Pr[S_3] - \Pr[S_2]| \leq \frac{l^2}{2} \cdot \mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$.

We now notice that the only value that depends on the challenged bit b in the original game is the ring signature $\Sigma = (\mathsf{pk}', \sigma', \Pi, \rho_\Pi)$. By the changes we made in $\mathbf{GAME}_2$, the values (Π, ρ_Π) are independent from b. What is more, by the changes made in $\mathbf{GAME}_3$, the values (pk', σ') are also independent from b. It follows that:

$$\begin{aligned}&\Pr[S_3] = 0\\ &\Pr[S_0] \leq \frac{l^2}{2} \cdot \mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda).\end{aligned}$$

5 Efficient Instantiation from Standard Assumptions

In this section we present three efficient instantiations of signatures with flexible public key. All schemes support the same exponentiation relation $\mathcal{R}_{exp}$. We say that public keys $\mathsf{pk}_1 = (\mathsf{pk}_{1,1}, \ldots, \mathsf{pk}_{1,k})$ and $\mathsf{pk}_2 = (\mathsf{pk}_{2,1}, \ldots, \mathsf{pk}_{2,k})$ are in this relation, denoted $(\mathsf{pk}_1, \mathsf{pk}_2) \in \mathcal{R}_{exp}$, if and only if there exists a value $r \in \mathbb{Z}_p^*$ such that $\forall_{i \in [k]} (\mathsf{pk}_{1,i})^r = \mathsf{pk}_{2,i}$.

We assume that the plain model schemes (i.e. without a common reference string) the public key contains the implicit security parameter λ and bilinear

group parameters BG. However, since the bilinear-group generation algorithm $\mathsf{BGGen}(\lambda)$ is deterministic, it follows that this does not influence the class-hiding property or unforgeability property. Therefore, for readability we omit those parameters.

The first instantiation we present is a warm-up scheme that is simple and efficient. Its security relies on the decisional linear and decisional Diffie-Hellman assumptions. It is based on a modified version of Waters signatures [32] for type-2 and type-3 pairings due to Chatterjee and Menezes [12]. The scheme has the key recovery property and can hence be used to implement stealth addresses as described in Section 4.

The second scheme works in the multi-user setting and features small public key size, independent of the security parameter λ. It is also based on the modified version of Waters signatures. This scheme is the ideal candidate for instantiating the group signature scheme presented in section 4. In combination with the $\mathsf{SPS\text{-}EQ}$ from [17] it results in the shortest static group signatures under standard assumptions. Further, using type-2 pairing and the random oracle model allows to use this scheme without the need for a trusted party.

The last of the presented schemes is introduced to efficiently instantiate our ring signatures. The scheme requires a proof system to show relation between public key and signature. We require the proof system to be perfectly sound even if the common reference string is computed by the signer. Thus, the proven statement must always be true, which presents a challenge in the security proof. Therefore, we consider a disjunction of this statement and a statement that is always false in regular use of the scheme but provides a "trapdoor" witness for the security reduction in the proof of unforgeability. Because of witness indistinguishability these proofs will not leak that the used witness was not for the first part of the disjunction.

5.1 Warm-up Scheme

Theorem 9. *Scheme 4 is a correct signature scheme with flexible public key.*

Proof. In the following, let $\lambda \in \mathbb{N}$ and $\omega, r \in \mathsf{coin}$.

1. Identical distribution of the keypairs output by $\mathsf{KeyGen_{FW}}$ and $\mathsf{TKeyGen_{FW}}$ holds trivially.
2. Let $(\mathsf{sk_{FW}} = (y, X\mathsf{pk_{FW}}), \mathsf{pk_{FW}} = (A, B, C, D, t, \boldsymbol{u})) \xleftarrow{\$} \mathsf{KeyGen_{FW}}(\lambda, \omega)$ and $m = (M_0 \dots M_1)_{\mathsf{bin}}$ be a message. For $\sigma \xleftarrow{\$} \mathsf{Sign_{FW}}(\mathsf{sk_{FW}}, m)$ we have for some random s,

$$\sigma = \left(X^y \cdot \left(\prod_{i=1}^{\lambda} u_i^{M_i} \right)^s, g_1^s, g_2^s \right)$$

Therefore, it holds that $e(g_1^s, g_2) = e(g_1, g_2)^s = e(g_1, g_2^s)$ and

$$t \cdot e(\prod_{i=1}^{\lambda} u_i^{M_i}, g_2^s) = e(X^y, g_2) \cdot e(\prod_{i=1}^{\lambda} u_i^{M_i \cdot s}, g_2)$$
$$= e(X^y \cdot \left(\prod_{i=1}^{\lambda} u_i^{M_i}\right)^s, g_2),$$

so verification before transformation succeeds.
The output $\mathsf{ChgSK}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, r)$ and $\mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$ will be the key pair $(\mathsf{sk}_{\mathsf{FW}}{}' = (y, (X)^r, \mathsf{pk}_{\mathsf{FW}}{}'), \mathsf{pk}_{\mathsf{FW}}{}' = (A^r, B^r, C^r, D^r, t^r, \boldsymbol{u}^r))$. Hence for a random s', a signature under the fresh signing key on m will be

$$\sigma' = \left(X^{yr} \cdot \left(\prod_{i=1}^{\lambda} u_i^{rM_i}\right)^{s'}, g_1^{s'}, g_2^{s'}\right)$$
$$= \left(\left(X^y \cdot \left(\prod_{i=1}^{\lambda} u_i^{M_i}\right)^{s'}\right)^r, g_1^{s'}, g_2^{s'}\right)$$

and in the verification equation with $\mathsf{pk}_{\mathsf{FW}}{}'$, the missing r will be provided by t^r.

3. Let the output of $\mathsf{TKeyGen}(\lambda, \omega)$ be

$$(\mathsf{sk}_{\mathsf{FW}} = (y, g_1^x, \mathsf{pk}_{\mathsf{FW}}),$$
$$\mathsf{pk}_{\mathsf{FW}} = (g_1^a, g_1^b, g_1^c, g_1^{x \cdot d}, t = e(g_1^{x \cdot y}, g_2), \boldsymbol{u}),$$
$$\tau = (d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\mu_1}, \ldots, g_2^{\mu_\lambda}), \omega')$$

and let $\mathsf{pk}_{\mathsf{FW}}{}' = (A, B, C, D, t', \boldsymbol{u}')$ be a different public key. Let $r = DLOG_t(t')$. We have $e(D^{1/d}, g_2^y) = t' = e(g_1^{xy}, g_2)^r$ if and only if $D = g_1^{xrd} = (g_1^{xd})^r$. The remaining checks ensure that $\forall_{i \in [\lambda+2]} \mathsf{pk}_{\mathsf{FW}}{}_i^r = \mathsf{pk}_{\mathsf{FW}}{}_i'$, i.e. $\mathsf{pk}_{\mathsf{FW}}{}' \in [\mathsf{pk}_{\mathsf{FW}}]_{\mathcal{R}}$.

Theorem 10. *Scheme 4 is existential unforgeable under flexible public key, assuming the decisional linear assumption holds.*

Proof. We begin the proof with the following remark.

Remark 2. For the simplicity of the proof and a tighter security reduction we rely on the results by Hofheinz et al. [24]. Applying expander codes on the signed message before signing it, allows to improve the tightness of the reduction from $O(\ell \cdot q)$ to $O(\sqrt{\ell} \cdot q)$, where ℓ is the bit length of the signed message and q is the number of signing queries. To improve readability and to maintain simplicity of the scheme, we do not describe this idea in the above scheme. However, we notice that in such a case the size of the public key increases (i.e. the size of vector $\boldsymbol{u}$), as we expand the message before signing.

Let $(f_1, f_2, h_1, h_2, f_1^\alpha, f_2^\alpha, h_1^\beta, h_2^\beta, g_1^\gamma, g_2^\gamma)$ be an instance of the decisional linear problem and let $\mathcal{A}$ be an PPT adversary the has non-negligible advantage $\mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$. We will show an algorithm $\mathcal{R}$ that uses $\mathcal{A}$ to break the above problem instance.

In the first step, the reduction $\mathcal{R}$ prepares the public key $\mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, \boldsymbol{u})$ as follows. It sets:

$$X = g_1^\gamma \qquad A = f_1^\alpha \qquad B = h_1^\beta$$
$$C = h_1 \qquad t = e(X, f_2) = e(X^\phi, g_2) \qquad u_i = (g_1^\gamma)^{a_i} g_1^{b_i}$$

and $D = X^d$, where a_i's are computed as random walks in $\{-1, 0, 1\}$ of length depending on the chosen expander code (see [24] for more details) and b_i's and d are chosen uniformly at random from $\mathbb{Z}_p^*$. The reduction also prepares the trapdoor $\tau = (d, f_2, f_2^\alpha, h_2^\beta, h_2, (g_2^\gamma)^{a_1} g_1^{b_1}, \ldots, (g_2^\gamma)^{a_\lambda} g_1^{b_\lambda})$.

To answer signing queries of $\mathcal{A}$, algorithm $\mathcal{R}$ proceeds as follows. Let m be the message and l the random coins queried by A. We now define $a(m) = \sum_{i\in[m]} a_i$ and $b(m) = \sum_{i\in[m]} b_i$, where we view $[m]$ as a set, such that $j \in [m]$ if the j-th bit of m is set to 1. Now we can always write $H(m) = (g_1^\gamma)^{a(m)} g_1^{b(m)}$ and a valid response for such a query is: $(f_1^{\gamma\cdot l} H(m)^{r\cdot l}, g_1^r, g_2^r) = ((g_1^{\gamma\cdot\phi + r\cdot(\gamma\cdot a(m) + b(m))})^l, g_1^r, g_2^r)$, where $f_1 = g_1^\phi$. Now if we set $g_1^r = (g_1^\phi)^x g_1^y$ and $g_2^r = (g_2^\phi)^x g_2^y$, then a valid signature is of the form: $((g_1^{\gamma\cdot\phi + (\phi\cdot x + y)\cdot(\gamma\cdot a(m) + b(m))})^l, (g_1^\phi)^x g_1^y, (g_2^\phi)^x g_2^y)$. This gives us

$$(((g_1^{\gamma\cdot\phi})^{1 + x\cdot a(m)} (g_1^\phi)^{x\cdot b(m)} (g_1^\gamma)^{y\cdot a(m)} g_1^{y\cdot b(m)})^l, (g_1^\phi)^x g_1^y, (g_2^\phi)^x g_2^y). \tag{1}$$

We now distinguish two cases:

- if $a(m) \neq 0$, then $\mathcal{R}$ can simulate a valid signature via equation 1, by setting $x = -a(m)^{-1}$ and $y \xleftarrow{\$} \mathbb{Z}_q$ (note that in such a case the term $g_1^{\gamma\cdot\phi}$ vanishes),
- if $a(m) = 0$, then $\mathcal{R}$ cannot simulate a valid signature and fails.

Finally, $\mathcal{A}$ will output a valid signature under message m^*: $\sigma_{\hat{\mathsf{FW}}} = (\sigma_{\hat{\mathsf{FW}}}{}^1, \sigma_{\hat{\mathsf{FW}}}{}^2, \sigma_{\hat{\mathsf{FW}}}{}^3) = ((g_1^{\gamma\cdot\phi} H(m^*)^{r^*})^{l^*}, g_1^{r^*}, g_2^{r^*})$, for which we hope that $a(m^*) = 0$, and a public key $\hat{\mathsf{pk}_{\mathsf{FW}}}$ for which $(\mathsf{pk}_{\mathsf{FW}}, \hat{\mathsf{pk}_{\mathsf{FW}}}) \in \mathcal{R}$. Thus, we have $\sigma_{\hat{\mathsf{FW}}} = ((f_1^\gamma (g_1^{r^*})^{b(m^*)})^{l^*}, g_1^{r^*}, g_2^{r^*})$, for some unknown r^* but known $b(m^*)$. Since $(\mathsf{pk}_{\mathsf{FW}}, \hat{\mathsf{pk}_{\mathsf{FW}}}) \in \mathcal{R}$. This means that $\hat{\mathsf{pk}_{\mathsf{FW}}} = (A^{l^*}, B^{l^*}, C^{l^*}, D^{l^*}, t^{l^*}, \boldsymbol{u}^{l^*}) = ((f_1^\alpha)^{l^*}, (h_1^\beta)^{l^*}, (h_1)^{l^*}, (g_1^{\gamma\cdot d})^{l^*}, t^{l^*}, \boldsymbol{u}^{l^*})$. We now compute

$$T_1 = e(\sigma_{\hat{\mathsf{FW}}}{}^1, h_2) = e(f_1^\gamma (g_1^{r^*})^{b(m^*)}, h_2^{l^*}) \qquad T_2 = e(h_1^{l^*}, g_2^{r^*})^{b(m^*)} = e(g_1^{r^*\cdot b(m^*)}, h_2^{l^*})$$
$$T_3 = e((f_1^\alpha)^{l^*}, h_2) = e(f_1^\alpha, h_2^{l^*}) \qquad T_4 = e((h_1^\beta)^{l^*}, f_2) = e(f_1^\beta, h_2^{l^*})$$

Finally, the reduction $\mathcal{R}$ returns 1 if $T_1 \cdot T_2^{-1} = T_3 \cdot T_4$ and 0, otherwise. Note that $T_1 \cdot T_2^{-1} = e(f_1^\gamma, h_2^{l^*})$ and the above equation is correct only if $\gamma = \alpha + \beta$.

The success probability of the reduction $\mathcal{R}$ depends on whether it can answer all signing queries of $\mathcal{A}$ and on the returned forgery (i.e. for which we must have $a(m^*) = 0$). Reusing the arguments presented in [24], we can argue that the success probability is non-negligible. Thus, since $\mathcal{A}$ advantage is non-negligible, it follows that $\mathcal{R}$ has a non-negligible advantage in solving the decisional linear problem.

Theorem 11. *Scheme 4 is class-hiding, assuming the decisional Diffie-Hellman assumption in $\mathbb{G}_1$ holds.*

Proof. In this proof we will use the game based approach. We start with $\mathbf{GAME}_0$ which is the original class-hiding experiment and let S_0 be an event that the experiment evaluates to 1, i.e. the adversary wins. We then make small changes and show in the end that the adversary's advantage is zero. We will use S_i to denote the event that the adversary wins the class-hiding experiment in $\mathbf{GAME}_i$. Let $\sigma_{\mathsf{FW}} = (\sigma^1_{\mathsf{FW}}, \sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}})$ be the signature and $\mathsf{pk_{FW}}' = (A', B', C', D', t', \boldsymbol{u'})$ be the public key given to the adversary as part of the challenge. Moreover, let $\mathsf{pk}_{\mathsf{FW}0} = (A_0, B_0, C_0, D_0, t_0, \boldsymbol{u_0})$ and $\mathsf{pk}_{\mathsf{FW}1} = (A_1, B_1, C_1, D_1, t_1, \boldsymbol{u_1})$ be the public keys that are returned by the KeyGen algorithm on input of random coins ω_0 and ω_1 given to the adversary and $\hat{b}$ be the bit chosen by the challenger.

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

$\mathbf{GAME}_1$: In this game we change the way we sample $\mathsf{pk}_{\mathsf{FW}0}$ and $\mathsf{pk}_{\mathsf{FW}1}$. Instead of sampling directly from $\mathbb{G}_1$, we sample $a, b, c, d, x, \nu_1, \ldots, \nu_\lambda \xleftarrow{\$} \mathbb{Z}_p^*$ and set $A = g_1^a$, $B = g_1^b$, $C = g_1^c$, $D = g_1^d$, $X = g_1^x$ and $u_i = g_1^{\nu_i}$ for $i \in [\lambda]$. Moreover, we change the way $\mathsf{sk_{FW}}'$ and $\mathsf{pk_{FW}}'$ are computed from $\mathsf{sk}_{\mathsf{FW}\hat{b}}$ $\mathsf{pk}_{\mathsf{FW}\hat{b}}$, i.e. $\mathsf{pk_{FW}}' = (Q^a, Q^b, Q^c, Q^d, e(Q^x, g_2^y), (Q^{\nu_1}, \ldots, Q^{\nu_\lambda}))$, and $\mathsf{sk_{FW}}' = (y, Q^x, \mathsf{pk_{FW}}')$. In other words, instead of using the value r to randomize the public key and secret key, we use a group element Q to do it.

Because we can use the invertible sampling algorithm to retrieve the random coins ω_0 and ω_1 and since the distribution of the keys does not change, it follows that $\Pr[S_1] = \Pr[S_0]$. We can still compute the signature σ_{FW} using $\mathsf{sk_{FW}}'$.

$\mathbf{GAME}_2$: In this game instead of computing $\mathsf{pk_{FW}}' = (Q^a, Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_1}, \ldots, Q^{\nu_\lambda}))$ as in $\mathbf{GAME}_1$, we sample $A' \xleftarrow{\$} \mathbb{G}_1$ set $\mathsf{pk_{FW}}' = (A', Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_1}, \ldots, Q^{\nu_\lambda}))$.

We will show that this transition only lowers the adversary's advantage by a negligible fraction. In particular, we will show 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 a instance of this problem in $\mathbb{G}_1$. $\mathcal{R}$ samples $r_{0,A}, r_{1,A} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $A_0 = (g_1^\alpha)^{r_{0,A}}$, $A_1 = (g_1^\alpha)^{r_{1,A}}$.

Additionally, the reduction uses $Q = g_1^\beta$ and the public key

$$\mathsf{pk_{FW}}' = ((g_1^\gamma)^{r_{\hat{b},A}}, Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_1}, \ldots, Q^{\nu_\lambda})).$$

Note that since $\mathcal{R}$ knows the secret key $\mathsf{sk}_{\mathsf{FW}}{}'$ it can compute σ_{FW}. Finally notice, that if $\gamma = \alpha \cdot \beta$ then $(\mathsf{pk}_{\mathsf{FW}}{}', \sigma_{\mathsf{FW}})$ have the same distribution as in $\mathbf{GAME}_1$ and otherwise as in $\mathbf{GAME}_2$. Thus, we have $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_3$ (series of sub-games): In this game instead of computing $\mathsf{pk}_{\mathsf{FW}}{}' = (A', Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_1}, \ldots, Q^{\nu_\lambda}))$ as in $\mathbf{GAME}_2$, we sample $B', C', D', u'_1, \ldots, u'_\lambda \xleftarrow{\$} \mathbb{G}_1$ and set $\mathsf{pk}_{\mathsf{FW}}{}' = (A', B', C', D', e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (u'_1, \ldots, u'_\lambda))$.

This transition is composed of a number of sub-games, in which we change each element of the public key $\mathsf{pk}_{\mathsf{FW}}{}'$ separately. Obviously, we can use the same reduction as above and show that each change lowers the adversary's advantage by at most $\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$. It is worth noting, that the reduction can always create a valid signature σ_{FW}, since the secret key $(y_{\hat{b}}, Q^{x_{\hat{b}}}, \mathsf{pk}_{\mathsf{FW}}{}')$ can be computed by $\mathcal{R}$. Thus, we have $|\Pr[S_3] - \Pr[S_2]| \leq (3 + \lambda) \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

Let us now take a look at the randomized public key and signature given to the adversary. Because of all the changes, we have: $\mathsf{pk}_{\mathsf{FW}}{}' = (A', B', C'e(Q^{x_{\hat{b}} \cdot y_{\hat{b}}}, g_2), \boldsymbol{u'})$ and $\sigma_{\mathsf{FW}} = (Q^{x_{\hat{b}} \cdot y_{\hat{b}}} (\prod_{i \in [m]} u'_i)^r, g_1^r, g_2^r)$ for some $r \in \mathbb{Z}_p^*$ and $A', B', C', \boldsymbol{u'}, Q$, which are independent from the bit $\hat{b}$ and the original public keys. Since the value Q is random and only appears as part of the term $Q^{x_{\hat{b}} \cdot y_{\hat{b}}}$, we can always restate this term to $Q'^{x_{1-\hat{b}} \cdot y_{1-\hat{b}}}$ where $Q' = Q^{(x_{1-\hat{b}} \cdot y_{1-\hat{b}}) \cdot (x_{\hat{b}} \cdot y_{\hat{b}})^{-1}}$ and Q' is a random value. It follows that the adversaries advantage is zero, i.e. $\Pr[S_3] = 0$. Finally, we have $\mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \Pr[S_0] \leq (4 + \lambda) \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

5.2 Flexible Public Key Scheme in the Multi-user Setting

Theorem 12. *Scheme 5 is a correct signature scheme with flexible public key.*

Proof. Analogous to the correctness of Scheme 4.

Theorem 13. *Scheme 5 is existential unforgeable under flexible public key in the common reference string model, assuming the co-Flexible Diffie-Hellman assumption holds.*

Proof (Theorem 13). We will use similar steps to the proof of theorem 10 and rely on the results received by Hofheinz et al. [24].

Let $(g_1^\alpha, g_2^\alpha, g_1^\beta, g_2^\beta, g_1^\gamma, g_2^\gamma, g_1^\theta, g_2^\theta)$ be an instance of the co-Flexible Diffie-Hellman assumption problem and let $\mathcal{A}$ be an PPT adversary the has non-negligible advantage $\mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$. We will show an algorithm $\mathcal{R}$ that uses $\mathcal{A}$ to break the above problem instance.

In the first step, the reduction $\mathcal{R}$ prepares the common reference string $\rho = (\mathsf{BG}, Y_1, Y_2, \boldsymbol{u})$ and the public key $\mathsf{pk}_{\mathsf{FW}} = (A, B, X)$ as follows. It sets:

$$X = g_1^\beta \qquad Y_1 = g_1^\alpha \qquad Y_2 = g_2^\alpha$$
$$A = g_1^\gamma \qquad B = g_1^\theta \qquad u_i = (g_1^\beta)^{a_i} g_1^{b_i},$$

where a_i's are computed as random walks in $\{-1, 0, 1\}$ of length as defined in [24] and b_i's are chosen uniformly at random from $\mathbb{Z}_p^*$. Moreover, $\mathcal{R}$ sets $\tau = (g_2^{\gamma}, g_2^{\theta}, g_2^{\beta})$.

To answer signing queries of $\mathcal{A}$, algorithm $\mathcal{R}$ proceeds as follows. Let m be the message and l the random coins queried by A. We now define $a(m) = \sum_{i \in [m]} a_i$ and $b(m) = \sum_{i \in [m]} b_i$, where we view $[m]$ as a set, such that $j \in [m]$ if the j-th bit of m is set to 1. Now we can always write $H(m) = (g_1^{\beta})^{a(m)} g_1^{b(m)}$ and a valid response for such a query is:

$$(Y_1^{\beta \cdot l} H(m)^r, g_1^r, g_2^r) = ((g_1^{\alpha \cdot \beta})^l \cdot g_1^{r \cdot (\beta \cdot a(m) + b(m))}, g_1^r, g_2^r).$$

Now if we set $g_1^r = ((g_1^{\alpha})^x g_1^y)^l$ and $g_2^r = ((g_2^{\alpha})^x g_2^y)^l$, then a valid signature is of the form:

$$\left((g_1^{\alpha \cdot \beta + (\alpha \cdot x + y) \cdot (\beta \cdot a(m) + b(m))})^l, ((g_1^{\alpha})^x g_1^y)^l, ((g_2^{\alpha})^x g_2^y)^l \right).$$

This gives us

$$\begin{aligned} &(((g_1^{\alpha \cdot \beta})^{1 + x \cdot a(m)} (g_1^{\alpha})^{x \cdot b(m)} (g_1^{\beta})^{y \cdot a(m)} g_1^{y \cdot b(m)})^l, \\ &((g_1^{\alpha})^x g_1^y)^l, ((g_2^{\alpha})^x g_2^y)^l). \end{aligned}$$

We now distinguish two cases:

- if $a(m) \neq 0$, then $\mathcal{R}$ can simulate a valid signature via the above equation, by setting $x = -a(m)^{-1}$ and $y \xleftarrow{\$} \mathbb{Z}_q$ (note that in such a case the term $g_1^{\alpha \cdot \beta}$ vanishes),
- if $a(m) = 0$, then $\mathcal{R}$ cannot simulate a valid signature and fails.

Finally, $\mathcal{A}$ will output a valid signature under message m^*:

$$\sigma_{\hat{\mathsf{FW}}} = (\sigma_{\hat{\mathsf{FW}}}{}^1, \sigma_{\hat{\mathsf{FW}}}{}^2, \sigma_{\hat{\mathsf{FW}}}{}^3) = ((g_1^{\alpha \cdot \beta})^{l^*} H(m^*)^{r^*}, g_1^{r^*}, g_2^{r^*},)$$

for which we hope that $a(m^*) = 0$, and a public key $\mathsf{pk}_{\hat{\mathsf{FW}}}$ for which $(\mathsf{pk}_{\mathsf{FW}}, \mathsf{pk}_{\hat{\mathsf{FW}}}) \in \mathcal{R}$. Thus, we have

$$\sigma_{\hat{\mathsf{FW}}} = (g_1^{\alpha \cdot \beta \cdot l^*} (g_1^{r^*})^{b(m^*)}, g_1^{r^*}, g_2^{r^*}),$$

for some unknown r^* but known $b(m^*)$. Thus the reduction $\mathcal{R}$ can compute $\sigma_{\hat{\mathsf{FW}}}{}^1 \cdot (\sigma_{\hat{\mathsf{FW}}}{}^2)^{-b(m^*)} = g_1^{\alpha \cdot \beta \cdot l^*}$. Moreover, since $(\mathsf{pk}_{\mathsf{FW}}, \mathsf{pk}_{\hat{\mathsf{FW}}}) \in \mathcal{R}$. This means that $\mathsf{pk}_{\hat{\mathsf{FW}}} = (A^{l^*}, B^{l^*}, X^{l^*}) = ((g_1^{\gamma})^{l^*}, (g_1^{\theta})^{l^*}, g_1^{l^* \cdot \beta})$.

Finally, the reduction $\mathcal{R}$ returns $(A^{l^*}, B^{l^*}, \sigma_{\hat{\mathsf{FW}}}{}^1 \cdot (\sigma_{\hat{\mathsf{FW}}}{}^2)^{-b(m^*)})$, which as shown above is $((g_1^{\gamma})^{l^*}, (g_1^{\theta})^{l^*}, g_1^{\alpha \cdot \beta \cdot l^*})$. Again, the success probability of the reduction $\mathcal{R}$ depends on whether it can answer all signing queries of $\mathcal{A}$ and on the returned forgery. By the arguments presented in [24], we conclude that $\mathcal{R}$ has a non-negligible advantage in solving the co-Flexible Diffie-Hellman assumption if $\mathcal{A}$'s advantage is non-negligible.

Corollary 1. *Scheme 5 is existential unforgeability under flexible public key in the CRS model under the DLIN assumption.*

Theorem 14. *Scheme 5 is class-hiding under the DDH assumption in* $\mathbb{G}_1$.

Proof (Theorem 14). In this proof we will use the game based approach. We start with $\mathbf{GAME}_0$ which is the original class-hiding experiment and let S_0 be an event that the experiment evaluates to 1, i.e. the adversary wins. We then make small changes and show in the end that the adversary is unable to create a forged ring signature.We will use S_i to denote the event that the adversary wins the class-hiding experiment in $\mathbf{GAME}_i$.

Let $\sigma_{\mathsf{FW}} = (\sigma^1_{\mathsf{FW}}, \sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}})$ be the signature and $\mathsf{pk_{FW}}' = (A', B', X')$ be the public key given to the adversary. Moreover, let $\mathsf{pk_{FW}}_0 = (A_0, B_0, X_0)$ and $\mathsf{pk_{FW}}_1 = (A_1, B_1, X_1)$ be the public keys that are returned KeyGen on input of random coins ω_0 and ω_1 given to the adversary and $\hat{b}$ be the bit chosen by the challenger.

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

$\mathbf{GAME}_1$: In this game we change the way the public keys $\mathsf{pk_{FW}}_0$ and $\mathsf{pk_{FW}}_1$ are generated. Instead of sampling A, B, X from $\mathbb{G}_1$, we sample $a, b, x \leftarrow_{\$} \mathbb{Z}_p^*$ and set $A = g_1^a$, $B = g_1^b$, $X = g_1^x$. Moreover, we do not use the ChgSK algorithm to compute $\mathsf{sk_{FW}}'$ and $\mathsf{pk_{FW}}'$ but compute them as $\mathsf{pk_{FW}}' = (Q^{a_{\hat{b}}}, Q^{b_{\hat{b}}}, Q^{x_{\hat{b}}})$, and $\mathsf{sk_{FW}}' = (Q^{x_{\hat{b}} \cdot y}, \mathsf{pk_{FW}}')$, 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.

Observe that we can use the invertible sampling algorithm to retrieve the random coins ω_0 and ω_1. Moreover, since the distribution of the keys does not change, it follows that $\Pr[S_1] = \Pr[S_0]$. Note that we can still compute the signature σ_{FW} using $\mathsf{sk_{FW}}'$.

$\mathbf{GAME}_2$: In this game instead of computing

$$\mathsf{pk_{FW}}' = (Q^{a_{\hat{b}}}, Q^{b_{\hat{b}}}, Q^{x_{\hat{b}}})$$

as in $\mathbf{GAME}_1$, we sample $A' \xleftarrow{\$} \mathbb{G}_1$ and set

$$\mathsf{pk_{FW}}' = (A', Q^{b_{\hat{b}}}, Q^{x_{\hat{b}}}).$$

We will show that this transition only lowers the adversary's advantage by a negligible fraction. In particular, we will show a reduction $\mathcal{R}$ that uses an adversary $\mathcal{A}$ that can distinguish between those two games to break the decisional Diffie-Hellman assumption in $\mathbb{G}_1$. Let $(g_1^\alpha, g_1^\beta, g_1^\gamma)$ be an instance of this problem in $\mathbb{G}_1$. $\mathcal{R}$ samples $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $A_0 = (g_1^\alpha)^{r_0}$, $A_1 = (g_1^\alpha)^{r_1}$.

Additionally, the reduction uses $Q = g_1^\beta$ and the public key

$$\mathsf{pk_{FW}}' = ((g_1^\gamma)^{r_{\hat{b}}}, Q^{b_{\hat{b}}}, Q^{x_{\hat{b}}}).$$

Note that since A' is not used in the signing process, it follows that the reduction can simply generate the signature σ_{FW}.

Finally notice, that if $\gamma = \alpha \cdot \beta$ then $(\mathsf{pk_{FW}}', \sigma_{\mathsf{FW}})$ have the same distribution as in $\mathbf{GAME}_1$ and otherwise as in $\mathbf{GAME}_2$.

Thus, it follows that $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_3$: In this game instead of computing

$$\mathsf{pk_{FW}}' = (A', Q^{b_{\hat{b}}}, Q^{x_{\hat{b}}})$$

as in $\mathbf{GAME}_2$, we sample $B' \xleftarrow{\$} \mathbb{G}_1$ and set

$$\mathsf{pk_{FW}}' = (A', B', Q^{x_{\hat{b}}}).$$

We can use the same argument as above. Thus, it follows that $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

Let us now take a look at the randomized public key and signature given to the adversary. Because of all the changes, we have: $\mathsf{pk_{FW}}' = (A', B', Q^{x_{\hat{b}}})$ and $\sigma_{\mathsf{FW}} = ((Q^{x_{\hat{b}}})^y (\prod_{i \in [m]} u_i')^r, g_1^r, g_2^r)$ for some $r \in \mathbb{Z}_p^*$ and A', B', Q, which are independent from the bit $\hat{b}$. Since the value Q is random and only appears as part of the term $Q^{x_{\hat{b}}}$, we can always restate this term to $Q'^{x_{1-\hat{b}}}$ where $Q' = Q^{x_{1-\hat{b}} \cdot (x_{\hat{b}})^{-1}}$ and Q' is also a random value.

It follows that the adversaries advantage is zero, i.e. $\Pr[S_3] = 0$. Thus, we have $\mathsf{Adv}^{\mathsf{c\text{-}h}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \Pr[S_0] \leq 2 \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

Remark 3 (Key Recovery). To support key recovery, the public key must be extended to the form $\mathsf{pk_{FW}} = (A, B, C, X)$ for $C = Y_1^c$. The value c is then part of the trapdoor τ and can be used to restore the value Y_1^r, where r is the coin used to change the public key. Given Y_1^r we need to compute $Z^r = Y_1^{xr}$, therefore we also have to include x as part of the original secret key $\mathsf{sk_{FW}} = (x, Y_1^x) = (x, Z)$.

Transformation to Strong Existential Unforgeability. Scheme 5 is only existential unforgeable under flexible public key and this directly follows from the fact that given a signature $(g_1^{x \cdot y \cdot l} H(m)^r, g_1^r, g_2^r)$ on message m, we can compute a randomized signature $(\sigma^1_{\mathsf{FW}}, \sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}}) = (g_1^{x \cdot y \cdot l} H(m)^r \cdot H(m)^{r'}, g_1^r g_1^{r'}, g_2^r g_2^{r'})$ for a fresh value $r' \xleftarrow{\$} \mathbb{Z}_p^*$.

A generic transformation from existential unforgeable to strongly unforgeable signatures was proposed by Boneh, Shen and Waters [8]. In particular, they use Waters signatures as a case study. The transformation works for all signature scheme for which there exist two algorithms F_1 and F_2 with the following properties: 1) the output signature is (σ_1, σ_2), where $\sigma_1 \xleftarrow{\$} F_1(m, r, \mathsf{sk})$ and $\sigma_2 \xleftarrow{\$} F_2(r, \mathsf{sk})$, 2) given m and σ_2 there exists at most one σ_1 so that (σ_1, σ_2) is a valid signature under pk.

It is easy to see that those properties hold for the standard Waters signatures and for Scheme 5, since we can compute $\sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}}$ in algorithm F_2 and σ^1_{FW} in F_1. What is more, once the random value r is set, there exists exactly one value σ^1_{FW}, for which $(\sigma^1_{\mathsf{FW}}, \sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}})$ is valid under a *given* public key.

The high level idea of the solution is to bind the part computed by F_2 using a hash function, i.e. the output of F_2 is hashed together with the actual message m and sign the output. In a scenario, where we consider a given public key, this means that the signature cannot be randomized. Any manipulation of the values $(\sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}})$ would result in a different signed message, which would lead to an attack against existential unforgeability of the underlying scheme. Fixing $(\sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}})$ fixes σ^1_{FW}, as required by the properties above. Unfortunately, the second argument does not hold for strong unforgeability under flexible public key. Note that the adversary can still change σ^1_{FW} by randomizing the public key. We can overcome this by simply including the public key in the hash computation.

This high level idea prevents the randomization of the signature but breaks the security proof of the underlying scheme. To allow the security reduction to bypass this protection Boneh, Shen and Waters propose to sign a Pedersen commitment to this hash value, instead of the value itself. The reduction can use a trapdoor to bypass this protection using equivocality of the commitment scheme. At the same time the binding property still makes it impossible for the adversary to randomize the signature.

To apply this idea in our case, we first extend the common reference string ρ by and element $h \xleftarrow{s} \mathbb{G}_1$. This element is part of the commitment key for the Pedersen scheme. More details are given in scheme 6.

Theorem 15. *Scheme 6 is strong existential unforgeable under flexible public key in the CRS model, assuming the co-Flexible Diffie-Hellman assumption holds and the hash function* H *is collision-resistant.*

Proof (Sketch). The proof follows directly from the proof given in [8].

Theorem 16. *Scheme 6 is class-hiding under the DDH assumption in* $\mathbb{G}_1$.

Proof (Sketch). We can apply the same reasoning as in the proof of Theorem 14. Note that the additional element s is just a random value that is independent from the bit $\hat{b}$ chosen by the challenger.

5.3 Flexible Public Key with Public Key in $(\mathbb{G}_1^*)^k$

The key element of scheme 7 is the composition of Waters signatures and the perfectly-sound proof system for pairing product equations presented in section 2.2. The proof is used to bind parts of the signature with the public key. Since the proof is only computational witness-indistinguishable, we use OR composition to allow for a trapdoor witness (i.e. a witness that allows to show that the statement is true without the link between public key and signature). This trapdoor witness is based on the decisional linear assumption, i.e. the statement

is true if part of the user's public key is a DLIN tuple. More formally, we will use the proof system for the following relation $\mathcal{R}_{Flex}$:

$$
\begin{aligned}
&((\mathsf{BG}, X, Y_1, t, (D_0, D_1, D_2, D_3, D_4, D_5)); (Y_2, d_1, d_2, d_3, d_4, X')) \in \mathcal{R}_{Flex} \leftrightarrow \\
&\quad \exists_{Y_2}(\ e(Y_1, g_2) = e(g_1, Y_2) \ \ \wedge \ \ t = e(X, Y_2)\ ) \ \vee \\
&\quad \exists_{X', Y_2, d_1, d_2, d_3, d_4}(\ D_1 = D_0^{d_1} \ \ \wedge \ \ D_2 = D_0^{d_2} \ \ \wedge \ \ D_3 = D_0^{d_1 \cdot d_3} \ \ \wedge \\
&\quad D_4 = D_0^{d_2 \cdot d_4} \ \ \wedge \ \ D_5 = D_0^{d_3 + d_4} \ \ \wedge \ \ t = e(X', Y_2)\ ),
\end{aligned}
$$

where $(Y_2, d_1, d_2, d_3, d_4, X')$ is the witness and $(\mathsf{BG}, X, Y_1, t, (D_0, D_1, D_2, D_3, D_4, D_5))$ the statement. The above statement can be efficiently expressed by the following set of pairing product equations:

$$
\begin{aligned}
1_{\mathbb{G}_T} &= e(B_1, B_2) & 1_{\mathbb{G}_T} &= e(B_1, g_2) \cdot e(g_1, B_2) \cdot e(g_1, g_2)^{-1} \\
1_{\mathbb{G}_T} &= e(Y_1, g_2 \cdot B_2^{-1}) \cdot e(B_1, Y_2) & 1_{\mathbb{G}_T} &= e(B_1, Y_2) \cdot e(B_1, Y_2')^{-1} \\
1_{\mathbb{G}_T} &= e(X, g_2 \cdot B_2^{-1}) \cdot e(X', g_2 \cdot B_2^{-1})^{-1} & 1_{\mathbb{G}_T} &= e(D_1, B_2) \cdot e(D_0^{d_1}, B_2)^{-1} \\
1_{\mathbb{G}_T} &= e(D_2, B_2) \cdot e(D_0^{d_2}, B_2)^{-1} & 1_{\mathbb{G}_T} &= e(D_3, B_2) \cdot e(D_0^{d_1 \cdot d_3}, B_2)^{-1} \\
1_{\mathbb{G}_T} &= e(D_4, B_2) \cdot e(D_0^{d_2 \cdot d_4}, B_2)^{-1} & 1_{\mathbb{G}_T} &= e(D_5, B_2) \cdot e(D_0^{d_3 + d_4}, B_2)^{-1}
\end{aligned}
$$

and $t = e(X', Y_2')$, where $B_1, B_2, Y_2, Y_2', D_0^{d_1}, D_0^{d_2}, D_0^{d_1 \cdot d_3}, D_0^{d_2 \cdot d_4}, D_0^{d_3 + d_4}$ are variables. Note that the above set of equation could potentially be simplified, but we opted for this form because of simplicity and readability.

The idea is that the variables B_1 and B_2 are g_1, $1_{\mathbb{G}_2}$ or $1_{\mathbb{G}_1}$, g_2. This is ensured by the first two equations. Those "bit" like variables are used to simulate the OR statement. Let us assume that $B_1 = g_1$ and $B_2 = 1_{\mathbb{G}_2}$, then this set of equations is true only if $Y_2 = g_2^y$ and $t = e(X, g_2^y)$, where $Y_1 = g_1^y$. Let us now assume that $B_1 = 1_{\mathbb{G}_1}$ and $B_2 = g_2$, then this set is true for some Y_2' (because the equation $1_{\mathbb{G}_T} = e(1_{\mathbb{G}_1}, Y_2) \cdot e(1_{\mathbb{G}_1}, Y_2')^{-1}$ is always true, Y_2' does not have to be Y_2) and some X' such that $t = e(X', Y_2')$, and a DLIN tuple $(D_0, D_1, D_2, D_3, D_4, D_5)$.

We note that the statement is always true if $(D_1, D_2, D_3, D_4, D_5)$ is a DLIN tuple for the generator D_0 (instead the generator g_1 in the original assumption). What is more, the statement is also always true for $(D_0^r, D_1^r, D_2^r, D_3^r, D_4^r, D_5^r)$ if it was true for the tuple $(D_0, D_1, D_2, D_3, D_4, D_5)$. In other words, the former tuple is a DLIN tuple if and only if the latter tuple is a DLIN tuple.

Theorem 17. *Scheme 7 is a correct signature scheme with flexible public key.*

Proof. Analogous to the correctness of Scheme 4.

Theorem 18. *Scheme 7 is existential unforgeable under flexible public key under the sq-Flexible Diffie-Hellman assumption and assuming the proof system is perfectly sound in the plain model.*

Proof (Theorem 18). In this proof we will use the game based approach. We start with $\mathbf{GAME}_0$ which is the unforgeability experiment and let S_0 be an event that the experiment evaluates to 1, i.e. the adversary wins. We then make one small

change in the experiment i.e. we fix the tuple $(D_0, D_1, D_2, D_3, D_4, D_5)$ in the user's public key to be a non-DLIN tuple, which implies that for the forgery outputted by the adversary it must hold that $t = e(X', Y_2)$, $e(Y_1, g_2) = e(g_1, Y_2)$ and $e(X, g_2) = e(X', g_2)$. Finally, we show that if there exists an adversary $\mathcal{A}$ that has non-negligible advantage in the changed experiment, then we can build a reduction algorithm $\mathcal{R}$ that has non-negligible advantage in breaking the sq-Flexible Diffie-Hellman problem.

GAME$_0$: The original unforgeability experiment.

GAME$_1$: Let $\mathsf{pk_{FW}} = (A, B, C, X, g_1^y, (D_0, D_1, D_2, D_3, D_4, D_5), \boldsymbol{u})$ be the public key given to the adversary. We fix $(D_0, D_1, D_2, D_3, D_4, D_5)$ to be a non-DLIN tuple, i.e. we set $D_0 = g_1^{d_0}$, $D_1 = D_0^{d_1}$, $D_2 = D_0^{d_2}$, $D_3 = D_0^{d_1 \cdot d_3}$, $D_4 = D_0^{d_2 \cdot d_4}$ and $D_5 = D_0^{d_3 + d_4 - 1}$.

It is obvious that we have $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}^{\text{linear}}_{\mathcal{A}}(\lambda)$. Given a DLIN instance $(f_1, f_2, h_1, h_2, f_1^a, f_2^a, h_1^b, h_2^b, g_1^z, g_2^z)$, the reduction algorithm can set $D_0 = g_1^r$, $D_1 = f_1^r$, $D_2 = h_1^r$, $D_3 = (f_1^a)^r$, $D_4 = (h_1^b)^r$ and $D_5 = (g_1^z)^r g_1^{-r} = (D_0)^{z-1}$.

We now show how to solve the sq-Flexible Diffie-Hellman problem, given an adversary $\mathcal{A}$ which has a non-negligible advantage in **GAME**$_1$. Let $(g_1^\alpha, g_2^\alpha, g_1^\beta, g_2^\beta, g_1^\gamma, g_2^\gamma, g_1^\theta, g_2^\theta)$ be an instance of the sq-Flexible Diffie-Hellman assumption problem and let $\mathcal{A}$ have a non-negligible advantage $\mathsf{Adv}^{\mathsf{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$. We will show an algorithm $\mathcal{R}$ that uses $\mathcal{A}$ to break the above problem instance.

Setup. In the first step, the reduction $\mathcal{R}$ prepares the public key $\mathsf{pk_{FW}} = (A, B, C, X, Y_1, (D_0, D_1, D_2, D_3, D_4, D_5), \boldsymbol{u})$ as follows. It sets:

$$
\begin{array}{lll}
X = g_1^\beta & Y_1 = g_1^\alpha & A = g_1^\gamma \\
B = g_1^\theta & C = g_1^{r_C} & D_0 = g_1^{d_0} \\
D_1 = D_0^{d_1} & D_2 = D_0^{d_2} & D_3 = D_0^{d_1 \cdot d_3} \\
D_4 = D_0^{d_2 \cdot d_4} & D_5 = D_0^{d_3 + d_4 - 1} & u_i = (g_1^\beta)^{a_i} g_1^{b_i}
\end{array}
$$

where $r_C, d_0, d_1, d_2, d_3, d_4 \xleftarrow{\$} \mathbb{Z}_p^*$, a_i's are computed as random walks in $\{-1, 0, 1\}$ of length as defined in [24] and b_i's are chosen uniformly at random from $\mathbb{Z}_p^*$. Moreover, $\mathcal{R}$ sets

$$\tau = (g_2^\gamma, g_2^\theta, g_2^{r_C}, g_2^\beta, g_2^\alpha, (g_2^{d_0}, g_2^{d_0 \cdot d_1}, g_2^{d_0 \cdot d_2}, g_2^{d_0 \cdot d_1 \cdot d_3}, g_2^{d_0 \cdot d_2 \cdot d_4}, g_2^{d_0 \cdot (d_3 + d_4 - 1)}), ((g_2^\beta)^{a_i} g_2^{b_i})_{i=1}^\lambda).$$

Answering signing queries. To answer signing queries of $\mathcal{A}$, algorithm $\mathcal{R}$ proceeds as follows. Let m be the message and l the random coins queried by A. We define $a(m) = \sum_{i \in [m]} a_i$ and $b(m) = \sum_{i \in [m]} b_i$, where we view $[m]$ as a set, such that $j \in [m]$ if the j-th bit of m is set to 1. Now we can always write

$H(m) = \prod_{i\in[m]} u_i = (g_1^{\beta})^{a(m)} g_1^{b(m)}$ and a valid response for such a query is:

$$((Y_1^l)^{\beta\cdot l}(H(m)^l)^r, g_1^r, g_2^r) = ((g_1^{\alpha\cdot\beta})^{l^2} \cdot g_1^{l\cdot r\cdot(\beta\cdot a(m)+b(m))}, g_1^r, g_2^r, (C^l)^r, t, \pi_{Flex}),$$

where $t = e(X^l, Y_2^l)$ and

$$\pi_{Flex} \xleftarrow{\$} \Pi.\mathsf{Prove}((\mathsf{BG}, X^l, Y_1^l, t, \{D_i\}_{i=0}^5), (Y_2^l, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1})).$$

Now if we set $g_1^r = ((g_1^{\alpha})^x g_1^y)^l$ and $g_2^r = ((g_2^{\alpha})^x g_2^y)^l$, then a valid signature is of the form:

$$\left((g_1^{\alpha\cdot\beta+(\alpha\cdot x+y)\cdot(\beta\cdot a(m)+b(m))})^{l^2}, ((g_1^{\alpha})^x g_1^y)^l, ((g_2^{\alpha})^x g_2^y)^l, ((g_1^{\alpha})^x g_1^y)^{l^2\cdot r_C}, t, \pi_{Flex}\right).$$

This gives

$$\begin{aligned}(&((g_1^{\alpha\cdot\beta})^{1+x\cdot a(m)}(g_1^{\alpha})^{x\cdot b(m)}(g_1^{\beta})^{y\cdot a(m)} g_1^{y\cdot b(m)})^{l^2},\\ &((g_1^{\alpha})^x g_1^y)^l, ((g_2^{\alpha})^x g_2^y)^l, ((g_1^{\alpha})^x g_1^y)^{l\cdot r_C}, t, \pi_{Flex}).\end{aligned}$$

Note that t and π_{Flex} can be computed as shown above. We now distinguish two cases:

- if $a(m) \neq 0$, then $\mathcal{R}$ can simulate a valid signature via the above equation, by setting $x = -a(m)^{-1}$ and $y \xleftarrow{\$} \mathbb{Z}_q$ (note that in such a case the term $g_1^{\alpha\cdot\beta}$ vanishes),
- if $a(m) = 0$, then $\mathcal{R}$ cannot simulate a valid signature and fails.

Final steps. Eventually $\mathcal{A}$ outputs a valid signature under message m^*:

$$\begin{aligned}\sigma_{\hat{\mathsf{FW}}} &= (\sigma_{\hat{\mathsf{FW}}}{}^1, \sigma_{\hat{\mathsf{FW}}}{}^2, \sigma_{\hat{\mathsf{FW}}}{}^3, \sigma_{\hat{\mathsf{FW}}}{}^4, t, \pi_{Flex})\\ &= ((g_1^{\alpha\cdot\beta})^{(l^*)^2} H(m^*)^{r^*\cdot l}, g_1^{r^*}, g_2^{r^*}, C^{r^*\cdot l^*}, t, \pi_{Flex})\end{aligned}$$

for which we hope that $a(m^*) = 0$, and a public key $\hat{\mathsf{pk}_{\mathsf{FW}}}$ for which $(\mathsf{pk}_{\mathsf{FW}}, \hat{\mathsf{pk}_{\mathsf{FW}}}) \in \mathcal{R}$. We now argue that since, we use a perfectly sound proof system and the proven statement must be true, we know that $t = e(g_1^{\beta\cdot l}, g_2^{\alpha\cdot l})$ (note that there exists no trapdoor witness and the first part of the statement must be true).

Thus, we have

$$\sigma_{\hat{\mathsf{FW}}} = (g_1^{\alpha\cdot\beta\cdot l^*\cdot l^*}(g_1^{r^*\cdot l^*})^{b(m^*)}, g_1^{r^*}, g_2^{r^*}, (g_1^{r^*\cdot l^*})^{r_C}, t, \pi_{Flex}),$$

for some unknown r^* but known $b(m^*)$ and $g_1^{r^*\cdot l^*}$ (because r_C is known). Thus the reduction $\mathcal{R}$ can compute $\sigma_{\hat{\mathsf{FW}}}{}^1 \cdot (\sigma_{\hat{\mathsf{FW}}}{}^4)^{-b(m^*)\cdot r_C^{-1}} = g_1^{\alpha\cdot\beta\cdot(l^*)^2}$. Moreover, since $(\mathsf{pk}_{\mathsf{FW}}, \hat{\mathsf{pk}_{\mathsf{FW}}}) \in \mathcal{R}$. This means that $\hat{\mathsf{pk}_{\mathsf{FW}}} = (A^{l^*}, B^{l^*}, \ldots) = (\hat{g}^{l^*}, \hat{h}^{l^*}, \ldots)$.

Finally, the reduction $\mathcal{R}$ returns $(A^{l^*}, B^{l^*}, \sigma_{\hat{\mathsf{FW}}}{}^1 \cdot (\sigma_{\hat{\mathsf{FW}}}{}^4)^{-b(m^*)\cdot r_C^{-1}})$.Again, the success probability of the reduction $\mathcal{R}$ depends on whether it can answer all signing queries of $\mathcal{A}$ and on the returned forgery. By the arguments presented in [24], we conclude that $\mathcal{R}$ has a non-negligible advantage in solving the sq-Flexible Diffie-Hellman assumption if $\mathcal{A}$'s advantage is non-negligible.

Theorem 19. *Scheme 7 is class-hiding, assuming the square decisional Diffie-Hellman assumption in $\mathbb{G}_1$ holds, the decisional linear assumption holds and the used proof system is computational witness-indistinguishable.*

Proof (Theorem 19). In this proof we will use the game based approach. We start with $\mathbf{GAME}_0$ which is the original class-hiding experiment and let S_0 be an event that the experiment evaluates to 1, i.e. the adversary wins. We then make small changes and show in the end that the adversary is unable to create a forged ring signature.We will use S_i to denote the event that the adversary wins the class-hiding experiment in $\mathbf{GAME}_i$.

Let $\sigma_{\mathsf{FW}} = (\sigma^1_{\mathsf{FW}}, \sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}}, \sigma^4_{\mathsf{FW}}, t, \pi_{Flex})$ be the signature and $\mathsf{pk_{FW}}' = (A', B', C', X', Y'_1, (D'_0, D'_1, D'_2, D'_3, D'_4, D'_5), \boldsymbol{u}')$ be the public key given to the adversary and let l be the random coin used to randomize the original public key. Moreover, let $\mathsf{pk}_{\mathsf{FW}0}$ and $\mathsf{pk}_{\mathsf{FW}1}$ be the public keys that are returned KeyGen on input of random coins ω_0 and ω_1 given to the adversary and $\hat{b}$ be the bit chosen by the challenger.

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

$\mathbf{GAME}_1$: In this game we change the way the challenged public keys $\mathsf{pk}_{\mathsf{FW}0}$ and $\mathsf{pk}_{\mathsf{FW}1}$ are generated, i.e. instead of sampling ω_0, ω_1 and running the algorithm $\mathsf{KeyGen}_{\mathsf{FW}}$, we first choose the key pair and then use invertible sampling to compute ω_0 and ω_1.

Observe that since we use invertible sampling we have $\Pr[S_1] = \Pr[S_0]$.

$\mathbf{GAME}_2$: In this game we change the way we sample $A, B, C, X, D_0, D_1, D_2, D_3, D_4, D_5$ and the vector $\boldsymbol{u}$, while preparing $\mathsf{pk}_{\mathsf{FW}0}$ and $\mathsf{pk}_{\mathsf{FW}1}$. We sample $a, b, c, x, d_0, d_1, d_2, d_3, d_4, d_5, \nu_1, \ldots, \nu_\lambda \leftarrow_\$ \mathbb{Z}_p^*$ and set $A = g_1^a$, $B = g_1^b$, $C = g_1^c$, $D_0 = g_1^{d_0}$, $D_1 = g_1^{d_1}$, $D_2 = g_1^{d_2}$, $D_3 = g_1^{d_3}$, $D_4 = g_1^{d_4}$, $D_5 = g_1^{d_5}$, $X = g_1^x$ and $u_i = g_1^{\nu_i}$ for $i \in [\lambda]$.

Since in both cases the distribution of the keys does not change, it follows that $\Pr[S_2] = \Pr[S_1]$.

$\mathbf{GAME}_3$: We now make sure that $(D'_0, D'_1, D'_2, D'_3, D'_4, D'_5)$ is a DLIN tuple for both public keys $\mathsf{pk}_{\mathsf{FW}0}$ and $\mathsf{pk}_{\mathsf{FW}1}$ given to the adversary.

It is easy to see that any adversary $\mathcal{A}$ that distinguishes this difference can be used by a reduction $\mathcal{R}$ to break the DLIN problem. Note that $\mathcal{R}$ can simulate the class-hiding experiment to $\mathcal{A}$, since it knows the witness $(Y_2, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1})$. Thus, we have $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\text{linear}}_{\mathcal{A}}(\lambda)$. Note that we can make the change for both public keys at once.

GAME$_4$: We use the witness $(Y_2, d_1, d_2, d_3, d_4, X')$ to compute the proof π_{Flex}. In other words, we show that the statement proven in π_{Flex} is true because $(D'_1, D'_2, D'_3, D'_4, D'_5)$ is a DLIN tuple for generator D'_0 and we know a X', Y_2 such that $t = e(X', Y_2)$. Note that in those computations we use values that we stored during the generation of the public key $\mathsf{pk}_{\mathsf{FW}\hat{b}}$.

The proof system is computationally witness-indistinguishable and the common reference string is chosen by the signer. Thus, we have $|\Pr[S_4] - \Pr[S_3]| \leq \mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda)$.

GAME$_5$: We now change the way we compute $t = e(X', Y'_2) = e(X^l, Y_2^l) = e(g_1^{x \cdot l}, g_2^{y \cdot l})$, which is part of the challenged signature. In this case we compute $t = e(X, Y_2^r)$ for some random r. This basically means that instead of using the randomization factor l^2, we use a fresh random value r.

We will now show that any adversary distinguishing between **GAME**$_4$ and **GAME**$_5$ can be used to break the square decisional Diffie-Hellman problem. Let $(g_1, g_1^\alpha, g_1^\gamma)$ be an instance of this problem. The reduction algorithm first sets the public key $\mathsf{pk}_{\mathsf{FW}}{}'$, given as part of the challenge, as follows (we assume that $\mathcal{R}$ used $l = \alpha$ as the randomization):

$$\begin{aligned}
&((g_1^\alpha)^a, (g_1^\alpha)^b, (g_1^\alpha)^c, (g_1^\alpha)^x, (g_1^\alpha)^y,\\
&((g_1^\alpha)^{d_0}, (g_1^\alpha)^{d_0 \cdot d_1}, (g_1^\alpha)^{d_0 \cdot d_2}, (g_1^\alpha)^{d_0 \cdot d_1 \cdot d_3}, (g_1^\alpha)^{d_0 \cdot d_2 \cdot d_4}, (g_1^\alpha)^{d_0(d_3+d_4)}),\\
&((g_1^\alpha)^{\nu_1}, \ldots, (g_1^\alpha)^{\nu_\lambda}))
\end{aligned}$$

and $t = e(g_1^\gamma, g_2^{x \cdot y})$. Note that we assume that the reduction algorithm also set the original public key that is randomized:

$$\begin{aligned}
\mathsf{pk}_{\mathsf{FW}\hat{b}} =&(g_1^a, g_1^b, g_1^c, g_1^x, g_1^y,\\
&(g_1^{d_0}, g_1^{d_0 \cdot d_1}, g_1^{d_0 \cdot d_2}, g_1^{d_0 \cdot d_1 \cdot d_3}, g_1^{d_0 \cdot d_2 \cdot d_4}, g_1^{d_0(d_3+d_4)}),\\
&(g_1^{\nu_1}, \ldots, g_1^{\nu_\lambda})).
\end{aligned}$$

Then it computes the signature under message m, where M_i denotes the i-th bit of m, as

$$\begin{aligned}
\sigma^1_{\mathsf{FW}} =&(g_1^\gamma)^{x \cdot y} \prod_{i=1}^{\lambda} ((g_1^\alpha)^{\nu_i})^{M_i \cdot r} & \sigma^2_{\mathsf{FW}} =&g_1^r\\
\sigma^3_{\mathsf{FW}} =&g_2^r & \sigma^4_{\mathsf{FW}} =&((g_1^\alpha)^c)^r
\end{aligned}$$

for a random $r \xleftarrow{\$} \mathbb{Z}_p^*$, t as computed above and π_{Flex} with a valid witness $(g_2^{x \cdot y}, d_1, d_2, d_3, d_4, g_1^\gamma)$.

It is easy to see that if $\gamma = \alpha^2 \mod p$, then this is essentially **GAME**$_4$ and if γ is a random value then this is **GAME**$_5$. It follows that $|\Pr[S_5] - \Pr[S_4]| \leq \mathsf{Adv}^{\mathsf{sddh}}_{\mathcal{A}}(\lambda)$.

GAME$_6$: We now make a similar change but now instead of using $t = e(X^r, Y_2)$, we use $t = e(g_1, g_2^r)$.

By the changes made in **GAME**$_5$, this value is blinded by a common factor r, which is never used again. It follows that this is just a conceptual change, since there always exist values $l_{\hat{b}}$ and $l_{1-\hat{b}}$ for which $r = l_{\hat{b}} \cdot x_{\hat{b}} \cdot y_{\hat{b}}$ and $r = l_{1-\hat{b}} \cdot x_{1-\hat{b}} \cdot y_{1-\hat{b}}$, where $\hat{b}$ is the bit chosen by the challenger. It follows that $\Pr[S_6] = \Pr[S_5]$. The signature is now independent of the values $x_{\hat{b}}, x_{1-\hat{b}}$ and $y_{\hat{b}}, y_{1-\hat{b}}$. Finally, we note that that we can still use the witness $(g_2^r, d_1, d_2, d_3, d_4, g_1)$ for proof π_{Flex}.

GAME$_7$: We now change the way we compute the public key $\mathsf{pk_{FW}}'$ given as part of the challenge. Instead of computing it as:

$$\begin{aligned}\mathsf{pk_{FW}}' =&((g_1^a)^l, (g_1^b)^l, (g_1^c)^l, (g_1^x)^l, (g_1^y)^l,\\ &((g_1^{d_0})^l, (g_1^{d_0 \cdot d_1})^l, (g_1^{d_0 \cdot d_2})^l, (g_1^{d_0 \cdot d_1 \cdot d_3})^l, (g_1^{d_0 \cdot d_2 \cdot d_4})^l, (g_1^{d_0 \cdot (d_3 + d_4)})^l),\\ &((g_1^{\nu_1})^l, \ldots, (g_1^{\nu_\lambda})^l))\end{aligned}$$

we compute it as

$$\begin{aligned}\mathsf{pk_{FW}}' =&(Q^a, Q^b, Q^c, Q^x, Q^y,\\ &(Q^{d_0'}, Q^{d_0' \cdot d_1'}, Q^{d_0' \cdot d_2'}, Q^{d_0' \cdot d_1' \cdot d_3'}, Q^{d_0' \cdot d_2' \cdot d_4'}, Q^{d_0' \cdot (d_3' + d_4')}),\\ &(Q^{\nu_1'}, \ldots, Q^{\nu_\lambda'})),\end{aligned}$$

where $Q \xleftarrow{\$} \mathbb{G}_1$.

Note that this is a conceptual change, since $Q = g_1^l$ for some l. Thus, $\Pr[S_7] = \Pr[S_6]$.

GAME$_8$ (series of sub-games): We now change the way we compute the public key $\mathsf{pk_{FW}}'$ given as part of the challenge. Instead of computing it as:

$$\begin{aligned}\mathsf{pk_{FW}}' =&(Q^a, Q^b, Q^c, Q^x, Q^y,\\ &(Q^{d_0}, Q^{d_0 \cdot d_1}, Q^{d_0 \cdot d_2}, Q^{d_0 \cdot d_1 \cdot d_3}, Q^{d_0 \cdot d_2 \cdot d_4}, Q^{d_0 \cdot (d_3 + d_4)}),\\ &(Q^{\nu_1}, \ldots, Q^{\nu_\lambda})), \text{we compute it as}\\ \mathsf{pk_{FW}}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^y,\\ &(Q^{d_0}, Q^{d_0 \cdot d_1}, Q^{d_0 \cdot d_2}, Q^{d_0 \cdot d_1 \cdot d_3}, Q^{d_0 \cdot d_2 \cdot d_4}, Q^{d_0 \cdot (d_3 + d_4)}),\\ &(g_1^{\nu_1'}, \ldots, g_1^{\nu_\lambda'}))\end{aligned}$$

for fresh random values $a', b', c', x', \nu_1', \ldots, \nu_\lambda' \xleftarrow{\$} \mathbb{Z}_p^*$. Note that we cannot apply this change to Q^y, since y is known to the adversary. He can compute this value for both challenged public keys $\mathsf{pk_{FW}}_0$ and $\mathsf{pk_{FW}}_1$ using random coins ω_0 and ω_1.

We will now show that any adversary $\mathcal{A}$ that distinguishes this difference can be used to break the decisional Diffie-Hellman problem instance $(g_1^\alpha, g_1^\beta, g_1^\gamma)$. In

fact, we use a series of sub-games to apply this change to each element of the public key separately and show that each sub-change is indistinguishable under the DDH assumption. We will only show the reduction for the first change, i.e. from Q^a to $g_1^{a'}$, since the other changes are similar.

The reduction algorithm $\mathcal{R}$ works as follows, it sets the public keys given to the adversary as

$$\begin{aligned}\mathsf{pk}_{\mathsf{FW}0} =&((g_1^{\alpha})^{a_0}, g_1^{b_0}, g_1^{c_0}, g_1^{x_0}, g_1^{y_0},\\ &(g_1^{d_{0,0}}, g_1^{d_{0,0}\cdot d_{0,1}}, g_1^{d_{0,0}\cdot d_{0,2}}, g_1^{d_{0,0}\cdot d_{0,1}\cdot d_{0,3}}, g_1^{d_{0,0}\cdot d_{0,2}\cdot d_{0,4}}, g_1^{d_{0,0}\cdot(d_{0,3}+d_{0,4})}),\\ &(g_1^{\nu_{0,1}}, \ldots, g_1^{\nu_{0,\lambda}}))\end{aligned}$$

and

$$\begin{aligned}\mathsf{pk}_{\mathsf{FW}1} =&((g_1^{\alpha})^{a_1}, g_1^{b_1}, g_1^{c_1}, g_1^{x_1}, g_1^{y_1},\\ &(g_1^{d_{1,0}}, g_1^{d_{1,0}\cdot d_{1,1}}, g_1^{d_{1,0}\cdot d_{1,2}}, g_1^{d_{1,0}\cdot d_{1,1}\cdot d_{1,3}}, g_1^{d_{1,0}\cdot d_{1,2}\cdot d_{1,4}}, g_1^{d_{1,0}\cdot(d_{1,3}+d_{1,4})}),\\ &(g_1^{\nu_{1,1}}, \ldots, g_1^{\nu_{1,\lambda}}))\end{aligned}$$

We first notice that the witness set in $\mathbf{GAME}_4$ is still valid

Let $\hat{b}$ be the bit chosen by the reduction in the class-hiding experiment. The adversary is given the challenged public key $\mathsf{pk}_{\mathsf{FW}}{}'$, with the corresponding signature. The reduction computes this public key as follows:

$$\begin{aligned}\mathsf{pk}_{\mathsf{FW}}{}' =&((g_1^{\gamma})^{a_{\hat{b}}}, (g_1^{\beta})^{b_{\hat{b}}}, (g_1^{\beta})^{c_{\hat{b}}}, (g_1^{\beta})^{x_{\hat{b}}}, (g_1^{\beta})^{y_{\hat{b}}},\\ &((g_1^{\beta})^{d_{\hat{b},0}}, (g_1^{\beta})^{d_{\hat{b},0}\cdot d_{\hat{b},1}}, (g_1^{\beta})^{d_{\hat{b},0}\cdot d_{\hat{b},2}}, (g_1^{\beta})^{d_{\hat{b},0}\cdot d_{\hat{b},1}\cdot d_{\hat{b},3}}, (g_1^{\beta})^{d_{\hat{b},0}\cdot d_{\hat{b},2}\cdot d_{\hat{b},4}}, (g_1^{\beta})^{d_{\hat{b},0}\cdot(d_{\hat{b},3}+d_{\hat{b},4})}),\\ &(g_1^{\beta})^{\nu_{\hat{b},1}}, \ldots, (g_1^{\beta})^{\nu_{\hat{b},\lambda}}))\end{aligned}$$

We can see that if $\gamma = \alpha\cdot\beta \mod p$, then the distribution of the experiment is as in $\mathbf{GAME}_7$ and otherwise as in $\mathbf{GAME}_8$. Using a hybrid argument (counting all sub-games) we have $|\Pr[S_8] - \Pr[S_7]| \leq (\lambda+4)\cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

$\mathbf{GAME}_9$: We now again change the way we compute the public key $\mathsf{pk}_{\mathsf{FW}}{}'$. Instead of computing it as:

$$\begin{aligned}\mathsf{pk}_{\mathsf{FW}}{}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^{y},\\ &(Q^{d_0}, Q^{d_0\cdot d_1}, Q^{d_0\cdot d_2}, Q^{d_0\cdot d_1\cdot d_3}, Q^{d_0\cdot d_2\cdot d_4}, Q^{d_0\cdot(d_3+d_4)}),\\ &(g_1^{\nu'_1}, \ldots, g_1^{\nu'_\lambda}))\text{we compute it as}\\ \mathsf{pk}_{\mathsf{FW}}{}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^{y},\\ &(W^{d_0}, W^{d_0\cdot d_1}, W^{d_0\cdot d_2}, W^{d_0\cdot d_1\cdot d_3}, W^{d_0\cdot d_2\cdot d_4}, W^{d_0\cdot(d_3+d_4)}),\\ &(g_1^{\nu'_1}, \ldots, g_1^{\nu'_\lambda}))\end{aligned}$$

for a fresh and random $W \xleftarrow{\$} \mathbb{G}_1$.

It is easy to see that this change only lowers the adversary's advantage by a negligible fraction. In particular, we can use the same reasoning as above to show that any algorithm that can distinguish this change, can be used to break the decisional Diffie-Hellman algorithm. However, this time we must apply the change directly to all elements. Otherwise, we would be unable to use the witness $(Y_2, d_1, d_2, d_3, d_4, X')$. It follows that $|\Pr[S_9] - \Pr[S_8]| \leq \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

GAME$_{10}$: We now revert the changes made in **GAME**$_5$ and **GAME**$_6$. Let

$$\begin{aligned}\mathsf{pk_{FW}}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, ((g_1)^y)^l,\\ &(W^{d_0}, W^{d_0 \cdot d_1}, W^{d_0 \cdot d_2}, W^{d_0 \cdot d_1 \cdot d_3}, W^{d_0 \cdot d_2 \cdot d_4}, W^{d_0 \cdot (d_3 + d_4)}),\\ &(g_1^{\nu_1'}, \ldots, g_1^{\nu_\lambda'}))\end{aligned}$$

be the public key given to the adversary as part of the challenge. Instead of computing $t = e(g_1, g_2^r)$, we compute it again as $t = e(g_1^{x'}, Y_2)$, where $Y_2 = g_2^{y \cdot l}$.

Since this reverts the changes made in games 5 and 6, we have $|\Pr[S_{10}] - \Pr[S_9]| \leq \mathsf{Adv}^{\mathsf{sddh}}_{\mathcal{A}}(\lambda)$.

GAME$_{11}$: We now switch the witness we use to compute the proof π_{Flex}. Instead of using $(Y_2, d_1, d_2, d_3, d_4, X')$, we use again $(Y_2, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1})$.

Because of the change made in **GAME**$_{10}$, we know that $Y_2 = g_2^{y \cdot l}$ is a valid witness for the statement proven by π_{Flex}. Note that in this case we have $X = g_1^{x'}$, $e(Y_1, g_2) = e(g_1, Y_2)$ and $t = e(X, Y_2)$. Thus, we have $|\Pr[S_{11}] - \Pr[S_{10}]| \leq \mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda)$.

GAME$_{12}$: We now change the way we compute the public key $\mathsf{pk_{FW}}'$ given as part of the challenge. Instead of computing it as:

$$\begin{aligned}\mathsf{pk_{FW}}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^y,\\ &(W^{d_0}, W^{d_0 \cdot d_1}, W^{d_0 \cdot d_2}, W^{d_0 \cdot d_1 \cdot d_3}, W^{d_0 \cdot d_2 \cdot d_4}, W^{d_0 \cdot (d_3 + d_4)}),\\ &(g_1^{\nu_1'}, \ldots, g_1^{\nu_\lambda'}))\text{we compute it as}\\ \mathsf{pk_{FW}}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^y,\\ &(W^{d_0}, W^{d_0 \cdot d_1}, W^{d_0 \cdot d_2}, W^{d_0 \cdot d_1 \cdot d_3}, W^{d_0 \cdot d_2 \cdot d_4}, W^{d_0 \cdot d_5}),\\ &(g_1^{\nu_1'}, \ldots, g_1^{\nu_\lambda'}))\end{aligned}$$

for a fresh and random $W \xleftarrow{\$} \mathbb{G}_1$ and $d_5 \xleftarrow{\$} \mathbb{Z}_p^*$.

Any adversary distinguishing a change can be used to break the DLIN assumption. Note that in the previous game we changed the witness back to the original one and we can now revert the changes made in **GAME**$_1$. It follows that $|\Pr[S_{12}] - \Pr[S_{11}]| \leq \mathsf{Adv}^{\text{linear}}_{\mathcal{A}}(\lambda)$.

GAME$_{13}$ (series of sub-games): Again we change the way we compute the public key $\mathsf{pk_{FW}}'$ given as part of the challenge. Instead of computing it as:

$$\begin{aligned}\mathsf{pk_{FW}}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^y,\\ &(W^{d_0}, W^{d_0 \cdot d_1}, W^{d_0 \cdot d_2}, W^{d_0 \cdot d_1 \cdot d_3}, W^{d_0 \cdot d_2 \cdot d_4}, W^{d_0 \cdot d_5}),\\ &(g_1^{\nu_1'}, \ldots, g_1^{\nu_\lambda'}))\text{we compute it as}\\ \mathsf{pk_{FW}}' =&(g_1^{a'}, g_1^{b'}, g_1^{c'}, g_1^{x'}, Q^y,\\ &(g_1^{d_0'}, g_1^{d_1'}, g_1^{d_2'}, g_1^{d_3'}, g_1^{d_4'}, g_1^{d_5'}),\\ &(g_1^{\nu_1'}, \ldots, g_1^{\nu_\lambda'}))\end{aligned}$$

for a fresh and random $d_1', d_2', d_3', d_4', d_5' \xleftarrow{\$} \mathbb{Z}_p^*$.

Following a similar approach to **GAME**$_8$ we show that any distinguishing adversary can be used to break the decisional Diffie-Hellman $(g_1^\alpha, g_1^\beta, g_1^\gamma)$ assumption. The key observation is that the values W and Q are independent and by setting $W = g_1^\beta$, the reduction algorithm is still able to compute the proof π_{Flex}. It is easy to see that we can start by setting $g_1^{d_0}$ to g_1^α and W^{d_0} to g_1^γ. We can then apply the same steps for the other values. We have $|\Pr[S_{13}] - \Pr[S_{12}]| \leq 5 \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda)$.

Finally, we conclude that $\Pr[S_{13}] = 0$. This follows from the fact that the only value that depends on bit $\hat{b}$ in the challenged public key is $Q^{y_{\hat{b}}}$. However, since Q is random and also only used for this value, we can always rewrite Q as $(Q')^{y_{\hat{b}}^{-1} \cdot y_{1-\hat{b}}}$. It follows that the signature itself is also independent of the bit $\hat{b}$ chosen by the challenger, which ends the proof.

Taking all game changes into account, we have:

$$\Pr[S_0] \leq 2 \cdot (\mathsf{Adv}^{\text{linear}}_{\mathcal{A}}(\lambda) + \cdot \mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{sddh}}_{\mathcal{A}}(\lambda)) + (\lambda + 10) \cdot \mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{A}}(\lambda).$$

5.4 Discussion

In this subsection we discuss the implications of the presented results. However, first we will instantiate the generic group signature scheme 2 and the generic ring signature scheme 3 with our SFPK instantiations.

We begin with the group signature scheme. We notice in this case we can use a SFPK scheme in the multi-user setting since the group manager can be trusted to perform a proper setup of public parameters. However, based on the security proof, the scheme must be strongly existential unforgeable. Thus, a natural candidate is our scheme 6. To fully instantiate the construction, we also require a SPS-EQ signature scheme. We instantiate it using the scheme presented in [17]. This scheme only support a one-time adaptation of the signature to a different representative (once adapted, the signature cannot be adapted further but the original signature can still be adapted). This does not impact our use of

Scheme	Public Key Size		Signature Size				CRS		Assumption	Key Recovery
	$[\mathbb{G}_1]$	$[\mathbb{G}_T]$	$[\mathbb{G}_1]$	$[\mathbb{G}_2]$	$[\mathbb{G}_T]$	$[\mathbb{Z}_p^*]$	$[\mathbb{G}_1]$	$[\mathbb{G}_2]$		
4	$(\lambda+4)$	1	2	1	-	-	-	-	DLIN + DDH	✓
5	3	-	2	1	-	-	$(\lambda+1)$	1	co-Flex (or DLIN) + DDH	✗/✓†
6	3	-	2	1	-	1	$(\lambda+2)$	1	co-Flex + DDH + CRHF	✗/✓†
7	$(\lambda+11)$	-	115	101	1	-	-	-	sq-Flex + SDDH + DLIN	✗

† The scheme can be transformed to support key recovery at an expense of a larger public key (one additional element in $\mathbb{G}_1$).

Fig. 1. Comparison of Presented Instantiations

the scheme since in our application the group member performs the adaptation only once per signing. Further, the scheme is only unforgeable under adaptive chosen-*open*-message attacks, but due to Lemma 2 it can still be used.

Lemma 2. *Let the public key of the* SFPK *scheme consist only of elements sampled directly from* $\mathbb{G}_1$ *or computed as* g_1^x*, where* $x \xleftarrow{\$} \mathbb{Z}_p^*$*. Theorems 4 and 5 still hold if the* SPS-EQ *scheme is only existential unforgeable under adaptive chosen-open-message attacks.*

Proof (Sketch). In the proof of theorem 4, instead of excluding the case the adversary creates a new user, we can toss a coin and chose the adversary's strategy (forging the SPS-EQ or SFPK signature). In case we end up choosing the SPS-EQ, we can freely choose the SFPK public keys and issue signing oracles to get all σ^i_{SPS}. In the proof of theorem 5 we use the unforgeability of SPS-EQ to exclude the case that the adversary issues an open query for a new user. Because this is the first change done, we can again freely choose the SFPK public keys and issue signing oracles to get all σ^i_{SPS}. Finally, we note that in such proofs we make a non-black-box use of the SFPK scheme.

For message space $(\mathbb{G}_1^*)^\ell$ the size of the SPS-EQ signature is $(4 \cdot \ell + 2)$ elements in $\mathbb{G}_1$ and 4 elements in $\mathbb{G}_2$. The security of the SPS-EQ scheme relies on the decisional linear assumption and the decisional Diffie-Hellman assumption in $\mathbb{G}_2$, while the security of our SFPK relies on the co-Flexible Diffie-Hellman assumption. All in all, the proposed instantiation yields a static group signature scheme that is secure under standard assumptions and has a signature size of 20 elements in $\mathbb{G}_1$ (counting elements in $\mathbb{Z}_q^*$ as $\mathbb{G}_1$) and 5 elements in $\mathbb{G}_2$. It therefore has shorter signatures than the current state-of-the-art scheme by Libert, Peters and Yung [25].

At the expense of introducing stronger assumptions even shorter signatures can be achieved by instantiating SPS-EQ with the scheme found in [18], which are unforgeable in the generic group model and have signatures of size 2 elements in

	Scheme	Signature size* [*bits*]	Anonymity	Assumptions
Random Oracle	Camenisch-Groth [10]	13 568	full	standard
	Boneh-Boyen-Shacham [7]	2 304	CPA-full	q-type
	Bichsel et al. [6]†	1 280	no key exposure	interactive
No Random Oracle	Boyen-Waters [9]	18 432	CPA-full	q-type
	Boyen-Waters [9]‡	6 656	CPA-full	q-type
	Libert-Peters-Yung [25]	9 216	full	standard
	Libert-Peters-Yung [25]	8 448	full	standard
	Ours with [18]	3 072	full	interactive
	Ours with [17]	7 680	full	standard

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

† The scheme defines additionally a join↔issue procedure

‡ Adapted from type 1 to type 3 pairings as in [25]

Fig. 2. Comparison of Static Group Signature Schemes

$\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. In this case we can instantiate our construction directly, without relying on lemma 2. More details are given in Figure 2.

We now focus on instantiating our ring signatures construction. Obviously we can use all schemes presented in section 5 and a generic perfectly sound proof system. The resulting ring signature scheme would not be interesting as there exist more efficient schemes with/without a trusted setup. However, we can improve the current state-of-the-art by proposing a ring signatures scheme, which is secure without a trusted party and has an efficient signature size which depends sub-linear on the number of ring members. As noted recently by Malavolta and Schröder [27] this is still an open problem.

To do so, we will use the membership proof presented by Chandran, Groth and Sahai [11]. The authors propose a perfectly sound proof that a public key $\mathsf{pk} \in \mathbb{G}_1$ (or $\mathsf{pk} \in \mathbb{G}_2$), is in a `Ring` of size n. The size of the proof is $O(\sqrt{n})$. The same idea can be applied to arbitrary public keys (i.e. consisting of group elements in different groups) in combination with a perfectly sound proof system for NP languages. However, this would yield inefficient ring signatures, but still of sub-linear size and without trust assumptions.

We achieve better results by using the proof system presented in scheme 1 and expressing the proven statement by a set of pairing product equations. Thus, we have to instantiate the SFPK scheme in order to support such system. The only schemes without a trusted party assumption are scheme 4 and 7. Unfortunately, the public key in scheme 4 contains an element in $\mathbb{G}_T$. Thus, we instantiate the SFPK scheme with scheme 7. The signature size of the resulting ring signature scheme is $O(\lambda \cdot \sqrt{n})$ and the public keys of signers have a size of $\lambda + 14$ elements

in $\mathbb{G}_1$. The instantiated ring signature scheme is the first efficient scheme that is secure under falsifiable assumptions, without a trusted party and with signature size that does not depend linearly on the number of ring members, which solves an open problem stated by Malavolta and Schröder.

6 Conclusion

We have presented signatures with flexible public keys, a novel primitive with applications to the design of primitives such as group and ring signatures.

The constructions we show are modular and cleanly separate identity-hiding and group membership certification components. Thus they allow for more efficient instantiation of the whole primitive by allowing for independent tuning of the components. As a result, the instantiations we present are among the most efficient in terms of signature sizes ever achieved under standard assumptions.

If the equivalence relation at the base of the SFPK instance fulfils additional properties, such as efficient key recovery, a multitude of additional stand-alone applications arise. We have demonstrated an example of this in the implementation of cryptocurrency stealth addresses.

References

[1] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. "Sanitizable Signatures". In: *ESORICS 2005*. Ed. by Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann. Springer, 2005.

[2] Feng Bao, Robert H. Deng, and Huafei Zhu. "Variations of Diffie-Hellman Problem". In: *Information and Communications Security ICICS 2003*. Ed. by Sihan Qing, Dieter Gollmann, and Jianying Zhou. Springer, 2003.

[3] Paulo S. L. M. Barreto and Michael Naehrig. "Pairing-Friendly Elliptic Curves of Prime Order". In: *SAC 2005*. Ed. by Bart Preneel and Stafford Tavares. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[4] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. "Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions". In: *EUROCRYPT 2003*. Ed. by Eli Biham. Springer, 2003.

[5] Adam Bender, Jonathan Katz, and Ruggero Morselli. "Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles". In: *TCC 2006*. Ed. by Shai Halevi and Tal Rabin. Springer, 2006.

[6] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. "Get Shorty via Group Signatures without Encryption". In: *SCN 2010*. Ed. by Juan A. Garay and Roberto De Prisco. Springer, 2010.

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *CRYPTO 2004*. Ed. by Matthew K. Franklin. Springer, 2004.

[8] Dan Boneh, Emily Shen, and Brent Waters. "Strongly Unforgeable Signatures Based on Computational Diffie-Hellman". In: *PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Springer, 2006.

[9] Xavier Boyen and Brent Waters. "Full-Domain Subgroup Hiding and Constant-Size Group Signatures". In: *PKC 2007*. Ed. by Tatsuaki Okamoto and Xiaoyun Wang. Springer, 2007.

[10] Jan Camenisch and Jens Groth. "Group Signatures: Better Efficiency and New Theoretical Aspects". In: *SCN 2004*. Ed. by Carlo Blundo and Stelvio Cimato. Springer, 2004.

[11] Nishanth Chandran, Jens Groth, and Amit Sahai. "Ring Signatures of Sublinear Size Without Random Oracles". In: *ICALP 2007*. Ed. by Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki. Springer, 2007.

[12] Sanjit Chatterjee and Alfred Menezes. "On cryptographic protocols employing asymmetric pairings - The role of Ψ revisited". In: *Discrete Applied Mathematics* 13 (2011).

[13] David Chaum and Eugène van Heyst. "Group Signatures". In: *EUROCRYPT '91*. Ed. by Donald W. Davies. Springer, 1991.

[14] Ivan Damgård. "Towards Practical Public Key Systems Secure Against Chosen Ciphertext attacks". In: *CRYPTO '91*. Ed. by Joan Feigenbaum. Springer, Heidelberg, 1992.

[15] Ivan Damgård and Jesper Buus Nielsen. "Improved Non-committing Encryption Schemes Based on a General Complexity Assumption". In: *CRYPTO 2000*. Ed. by Mihir Bellare. Springer, 2000.

[16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. "Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys." In: *PKC 2016, Part I*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Springer, 2016.

[17] Georg Fuchsbauer and Romain Gay. *Weakly Secure Equivalence-Class Signatures from Standard Assumptions*. Cryptology ePrint Archive, Report 2018/037. https://eprint.iacr.org/2018/037. 2018.

[18] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. *EUF-CMA-Secure Structure-Preserving Signatures on Equivalence Classes*. Cryptology ePrint Archive, Report 2014/944. 2014.

[19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. "Practical Round-Optimal Blind Signatures in the Standard Model". In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew Robshaw. Springer, Heidelberg, 2015.

[20] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. "Groth-Sahai Proofs Revisited". In: *PKC 2010*. Ed. by Phong Q. Nguyen and David Pointcheval. Springer, 2010.

[21] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Non-interactive Zaps and New Techniques for NIZK". In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Springer, 2006.

[22] Jens Groth and Amit Sahai. “Efficient Non-interactive Proof Systems for Bilinear Groups”. In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Springer, 2008.

[23] Christian Hanser and Daniel Slamanig. “Structure-Preserving Signatures on Equivalence Classes and Their Application to Anonymous Credentials”. In: *ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Springer, Heidelberg, 2014.

[24] Dennis Hofheinz, Tibor Jager, and Edward Knapp. “Waters Signatures with Optimal Security Reduction”. In: *PKC 2012*. Ed. by Marc Fischlin, Johannes A. Buchmann, and Mark Manulis. Springer, 2012.

[25] Benoît Libert, Thomas Peters, and Moti Yung. “Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions”. In: *CRYPTO 2015, Part II*. Ed. by Rosario Gennaro and Matthew Robshaw. Springer, 2015.

[26] Benoît Libert and Damien Vergnaud. “Multi-use Unidirectional Proxy Resignatures”. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. Alexandria, Virginia, USA: ACM, 2008.

[27] Giulio Malavolta and Dominique Schröder. “Efficient Ring Signatures in the Standard Model”. In: *ASIACRYPT 2017, Part II*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Springer, 2017.

[28] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. http://bitcoin.org/bitcoin.pdf.

[29] Ronald L. Rivest, Adi Shamir, and Yael Tauman. “How to Leak a Secret”. In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Springer, 2001.

[30] Peter Todd. *Stealth Addresses*. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2014-January/004020.html.

[31] Eric R. Verheul. “Self-Blindable Credential Certificates from the Weil Pairing”. In: *ASIACRYPT 2001*. 2001.

[32] Brent Waters. “Efficient Identity-Based Encryption Without Random Oracles”. In: *EUROCRYPT 2005*. Ed. by Ronald Cramer. Springer, Heidelberg, 2005.

$\mathsf{KeyGen}_{\mathsf{FW}}(\lambda, \omega)$

1 : $\boldsymbol{u} \xleftarrow{\$} \left(u_i \mid i \in [\lambda], u_i \xleftarrow{\$} \mathbb{G}_1\right)$
2 : $A, B, C, D, X \xleftarrow{\$} \mathbb{G}_1 \; y \xleftarrow{\$} \mathbb{Z}_p^*$
3 : $t \leftarrow e(X^y, g_2)$
4 : **return** $(\mathsf{pk}_{\mathsf{FW}} := (A, B, C, D, t, \boldsymbol{u}),$
5 : $\qquad \mathsf{sk}_{\mathsf{FW}} := (y, X, \mathsf{pk}_{\mathsf{FW}}))$

$\mathsf{TKeyGen}_{\mathsf{FW}}(\lambda, \omega)$

1 : $\boldsymbol{u} \xleftarrow{\$} \left(g_1^{\mu_i} \mid i \in [\lambda], \mu_i \xleftarrow{\$} \mathbb{Z}_p\right)$
2 : $a, b, c, d, x \xleftarrow{\$} \mathbb{Z}_p^* \; y \xleftarrow{\$} \mathbb{Z}_p^*$
3 : $t \leftarrow e(g_1^{x \cdot y}, g_2)$
4 : **return** $(\mathsf{pk}_{\mathsf{FW}} := (g_1^a, g_1^b, g_1^c, g_1^{x \cdot d}, t, \boldsymbol{u})$
5 : $\qquad \mathsf{sk}_{\mathsf{FW}} := (y, g_1^x, \mathsf{pk}_{\mathsf{FW}}),$
6 : $\qquad \tau := (d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\mu_1}, \ldots, g_2^{\mu_\lambda}))$

$\mathsf{Sign}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, m = (M_0 \ldots M_\lambda)_{\mathsf{bin}})$

1 : **parse** $\mathsf{sk}_{\mathsf{FW}} = (y, X, \mathsf{pk}_{\mathsf{FW}})$
2 : $r \xleftarrow{\$} \mathbb{Z}_p^*$
3 : **return**
4 : $\qquad \sigma_{\mathsf{FW}} := \left(X^y \cdot \left(\prod_{i=1}^{\lambda} u_i^{M_i}\right)^r, g_1^r, g_2^r\right)$

$\mathsf{Verify}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, m = (M_0 \ldots M_\lambda)_{\mathsf{bin}}, \sigma_{\mathsf{FW}})$

1 : **parse** $\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$
2 : $\qquad \mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, \boldsymbol{u})$
3 : **return** $e(\sigma_{\mathsf{W}}^2, g_2) = e(g_1, \sigma_{\mathsf{W}}^3) \wedge$
4 : $\qquad e(\sigma_{\mathsf{FW}}^1, g_2) = t \cdot e\left(\prod_{i=1}^{\lambda} u_i^{M_i}, \sigma_{\mathsf{FW}}^3\right)$

$\mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$

1 : **parse** $\mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, \boldsymbol{u})$
2 : **return** $\mathsf{pk}_{\mathsf{FW}}{}' := (A^r, B^r, C^r, D^r, t^r, \boldsymbol{u}^r)$

$\mathsf{ChkRep}_{\mathsf{FW}}(\tau, \mathsf{pk}_{\mathsf{FW}}, \mathsf{pk}_{\mathsf{FW}}{}')$

1 : **parse** $\mathsf{pk}_{\mathsf{FW}}{}' = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3, X, t, \mathsf{pk}_4, \ldots, \mathsf{pk}_{\lambda+3})$
2 : $\qquad \tau = (d, Y_2, \tau_1, \ldots, \tau_{\lambda+2})$
3 : **return** $e(X^{d^{-1}}, Y_2) = t \wedge$
4 : $\qquad \bigwedge_{i=1}^{\lambda+3} \bigwedge_{j=1}^{\lambda+3} e(\mathsf{pk}_i, \tau_j) = e(\mathsf{pk}_j, \tau_i)$

$\mathsf{Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$

1 : **parse** $\mathsf{sk} = (y, g_1^x, \mathsf{pk})$
2 : $\qquad \tau = (d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\mu_1}, \ldots g_2^{\mu_\lambda})$
3 : $\qquad \mathsf{pk}' = (A^r, B^r, C^r, D^r, t^r, \boldsymbol{u}^r)$
4 : $X' \leftarrow (D^r)^{1/d}$
5 : **return** $\mathsf{sk}' := (y, X', \mathsf{pk}')$

$\mathsf{ChgSK}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, r)$

1 : **parse** $\mathsf{sk}_{\mathsf{FW}} = (y, X, \mathsf{pk}_{\mathsf{FW}})$
2 : $\mathsf{pk}_{\mathsf{FW}}{}' \leftarrow \mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$
3 : **return** $\mathsf{sk}_{\mathsf{FW}}{}' := (y, (X^r), \mathsf{pk}_{\mathsf{FW}}{}')$

Scheme 4: Warm-up Scheme for Waters Signatures

$\mathsf{CRSGen}(\lambda, \omega)$

1 : $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(\lambda)$
2 : $\boldsymbol{u} \xleftarrow{\$} (u_i \mid i \in [\lambda], u_i \xleftarrow{\$} \mathbb{G}_1)$
3 : $y \xleftarrow{\$} \mathbb{Z}_p^*; Y_1 \leftarrow g_1^y; Y_2 \leftarrow g_2^y$
4 : **return** $\rho := (\mathsf{BG}, Y_1, Y_2, \boldsymbol{u})$

$\mathsf{KeyGen}_{\mathsf{FW}}(\lambda, \omega)$

1 : $A, B \xleftarrow{\$} \mathbb{G}_1; x \xleftarrow{\$} \mathbb{Z}_p^*$
2 : **return** $(\mathsf{pk}_{\mathsf{FW}} := (A, B, g_1^x)$
3 : $\mathsf{sk}_{\mathsf{FW}} := (Y_1^x, \mathsf{pk}_{\mathsf{FW}}))$

$\mathsf{TKeyGen}_{\mathsf{FW}}(\lambda, \omega)$

1 : $a, b, x \xleftarrow{\$} \mathbb{Z}_p^*$
2 : **return** $(\mathsf{pk}_{\mathsf{FW}} := (g_1^a, g_1^b, g_1^x),$
3 : $\mathsf{sk}_{\mathsf{FW}} := (Y_1^x, \mathsf{pk}_{\mathsf{FW}}),$
4 : $\tau := (g_2^a, g_2^b, g_2^x))$

$\mathsf{Sign}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, m = (M_0 \dots M_\lambda)_{\mathsf{bin}})$

1 : **parse** $\mathsf{sk}_{\mathsf{FW}} = (Z, \mathsf{pk}_{\mathsf{FW}})$
2 : $r \xleftarrow{\$} \mathbb{Z}_p^*$
3 : **return**
4 : $\sigma_{\mathsf{FW}} := (Z \cdot (\prod_{i=1}^{\lambda} u_i^{M_i})^r, g_1^r, g_2^r)$

$\mathsf{Verify}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, m = (M_0 \dots M_\lambda)_{\mathsf{bin}}, \sigma_{\mathsf{FW}})$

1 : **parse** $\mathsf{pk}_{\mathsf{FW}} = (A, B, X)$
2 : $\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$
3 : **return** $e(\sigma_{\mathsf{W}}^2, \hat{g}_2) = e(\hat{g}_1, \sigma_{\mathsf{W}}^3) \wedge$
4 : $e(\sigma_{\mathsf{FW}}^1, \hat{g}_2) = e(X, Y_2) \cdot e(\prod_{i=1}^{\lambda} u_i^{M_i}, \sigma_{\mathsf{FW}}^3)$

$\mathsf{ChgSK}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, r)$

1 : **parse** $\mathsf{sk}_{\mathsf{FW}} = (Z, \mathsf{pk}_{\mathsf{FW}})$
2 : $\mathsf{pk}_{\mathsf{FW}}{}' \leftarrow \mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$
3 : **return**
4 : $\mathsf{sk}_{\mathsf{FW}}{}' := ((Z)^r, \mathsf{pk}_{\mathsf{FW}}{}')$

$\mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$

1 : **parse** $\mathsf{pk}_{\mathsf{FW}} = (A, B, X)$
2 : **return** $\mathsf{pk}_{\mathsf{FW}}{}' := (A^r, B^r, X^r)$

$\mathsf{ChkRep}_{\mathsf{FW}}(\tau, \mathsf{pk}_{\mathsf{FW}}{}')$

1 : **parse** $\tau = (\tau_1, \tau_2, \tau_3)$
2 : $\mathsf{pk}_{\mathsf{FW}}{}' = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3)$
3 : **return**
4 : $\bigwedge_{i \in [3]} \bigwedge_{j \in [3]} e(\mathsf{pk}_i, \tau_j) = e(\mathsf{pk}_j, \tau_i)$

Scheme 5: Multi-user Flexible Public Key

$\mathsf{Sign}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, m)$

1 : **parse** $\mathsf{sk}_{\mathsf{FW}} = (Z, \mathsf{pk}_{\mathsf{FW}})$
2 : $r \xleftarrow{\$} \mathbb{Z}_p^*; s \xleftarrow{\$} \mathbb{Z}_p^*$
3 : $v \leftarrow \mathsf{H}(m, g_1^r, g_2^r, \mathsf{pk}_{\mathsf{FW}}) \in \mathbb{Z}_p^*$
4 : $(M_0 \dots M_\lambda)_{\mathsf{bin}} \leftarrow g_1^v h^s$
5 : **return** $\sigma_{\mathsf{FW}} := (Z \cdot (\prod_{i=1}^{\lambda} u_i^{M_i})^r, g_1^r, g_2^r, s)$

$\mathsf{Verify}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, m, \sigma_{\mathsf{FW}})$

1 : **parse** $\mathsf{pk}_{\mathsf{FW}} = (A, B, X)$
2 : $\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3, s)$
3 : $v \leftarrow \mathsf{H}(m, g_1^r, g_2^r, \mathsf{pk}_{\mathsf{FW}})$
4 : $(M_0 \dots M_\lambda)_{\mathsf{bin}} \leftarrow g_1^v h^s$
5 : **return** $e(\sigma_{\mathsf{W}}^2, \hat{g}_2) = e(\hat{g}_1, \sigma_{\mathsf{W}}^3) \wedge$
6 : $e(\sigma_{\mathsf{FW}}^1, \hat{g}_2) = e(X, Y_2) \cdot e(\prod_{i=1}^{\lambda} u_i^{M_i}, \sigma_{\mathsf{FW}}^3)$

Scheme 6: Strong Existential Unforgeable Variant of Scheme 5

$\mathsf{KeyGen}_{\mathsf{FW}}(\lambda, \omega)$

$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(\lambda)$

$\boldsymbol{u} \xleftarrow{\$} \left(u_i \mid i \in [\lambda], u_i \xleftarrow{\$} \mathbb{G}_1\right)$

$A, B, C, X, \{D_i\}_{i=0}^{5} \xleftarrow{\$} \mathbb{G}_1,\ y \xleftarrow{\$} \mathbb{Z}_p^*$

return $(\mathsf{pk}_{\mathsf{FW}} := (A, B, C, X, g_1^y, \{D_i\}_{i=0}^{5}, \boldsymbol{u}),$

$\mathsf{sk}_{\mathsf{FW}} := (X^y, g_2^y, \mathsf{pk}_{\mathsf{FW}}))$

$\mathsf{TKeyGen}_{\mathsf{FW}}(\lambda, \omega)$

$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(\lambda)$

$\boldsymbol{u}_1, \boldsymbol{u}_2 \xleftarrow{\$} \left(g_1^{\mu_i}, g_2^{\mu_i} \mid i \in [\lambda], \mu_i \xleftarrow{\$} \mathbb{Z}_p\right)$

$a, b, c, x, \{d_i\}_{i=0}^{5}, y \xleftarrow{\$} \mathbb{Z}_p^*$

return $(\mathsf{pk}_{\mathsf{FW}} := (g_1^a, g_1^b, g_1^c, g_1^x, g_1^y, \left\{g_1^{d_i}\right\}_{i=0}^{5}, \boldsymbol{u}_1),$

$\mathsf{sk}_{\mathsf{FW}} := (g_1^{x \cdot y}, g_2^y, \mathsf{pk}_{\mathsf{FW}}),$

$\tau := (g_2^a, g_2^b, g_2^c, g_2^x, g_2^y, \left\{g_2^{d_i}\right\}_{i=0}^{5}, \boldsymbol{u}_2))$

$\mathsf{Sign}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, m = (M_0 \dots M_\lambda)_{\mathsf{bin}})$

parse $\mathsf{sk}_{\mathsf{FW}} = (Z, Y_2, \mathsf{pk}_{\mathsf{FW}})$

$\mathsf{pk}_{\mathsf{FW}} = (A, B, C, X, Y_1, \{D_i\}_{i=0}^{5}, \boldsymbol{u})$

$r \xleftarrow{\$} \mathbb{Z}_p^*;\ t \leftarrow e(X, Y_2)$

$\pi_{Flex} \xleftarrow{\$} \Pi.\mathsf{Prove}((\mathsf{BG}, X, Y_1, t, \{D_i\}_{i=0}^{5}),$

$(Y_2, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}, 1_{\mathbb{G}_1}))$

return $\sigma_{\mathsf{FW}} := (Z \cdot (\prod_{i=1}^{\lambda} u_i^{M_i})^r, g_1^r, g_2^r, C^r, t, \pi_{Flex})$

$\mathsf{Verify}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, m = (M_0 \dots M_\lambda)_{\mathsf{bin}}, \sigma_{\mathsf{FW}})$

parse $\mathsf{pk}_{\mathsf{FW}} = (A, B, C, X, g_1^y, \{D_i\}_{i=0}^{5}, \boldsymbol{u})$

$\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3, \sigma_{\mathsf{FW}}^4, t, \pi_{Flex})$

return

$\Pi.\mathsf{Verify}((\mathsf{BG}, X, Y_1, t, \{D_i\}_{i=0}^{5}), \pi_{Flex}) = 1 \wedge$

$e(\sigma_{\mathsf{W}}^2, g_2) = e(g_1, \sigma_{\mathsf{W}}^3) \ \wedge\ e(C, \sigma_{\mathsf{FW}}^3) = e(\sigma_{\mathsf{FW}}^4, g_2) \wedge$

$e(\sigma_{\mathsf{FW}}^1, g_2) = t \cdot e(\prod_{i=1}^{\lambda} u_i^{M_i}, \sigma_{\mathsf{FW}}^3)$

$\mathsf{ChgSK}_{\mathsf{FW}}(\mathsf{sk}_{\mathsf{FW}}, r)$

parse $\mathsf{sk}_{\mathsf{FW}} = (X^y, \mathsf{pk}_{\mathsf{FW}})$

$\mathsf{pk}_{\mathsf{FW}}{}' \leftarrow \mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$

return $\mathsf{sk}_{\mathsf{FW}}{}' := ((X^y)^{r^2}, (g_2^y)^r, \mathsf{pk}_{\mathsf{FW}}{}')$

$\mathsf{ChkRep}_{\mathsf{FW}}(\tau, \mathsf{pk}_{\mathsf{FW}})$

parse $\tau = (\tau_1, \dots, \tau_{\lambda+11})$

$\mathsf{pk}_{\mathsf{FW}} = (\mathsf{pk}_1, \dots, \mathsf{pk}_{\lambda+11})$

return $\bigwedge_{i=1}^{\lambda+11} \bigwedge_{j=1}^{\lambda+11} e(\mathsf{pk}_i, \tau_j) = e(\mathsf{pk}_j, \tau_i)$

$\mathsf{ChgPK}_{\mathsf{FW}}(\mathsf{pk}_{\mathsf{FW}}, r)$

parse $\mathsf{pk}_{\mathsf{FW}} = (A, B, C, X, Y_1, \{D_i\}_{i=0}^{5}), \boldsymbol{u})$

return $\mathsf{pk}_{\mathsf{FW}}{}' := (A^r, B^r, C^r, X^r, Y^r, \{D_i^r\}_{i=0}^{5}), \boldsymbol{u}^r)$

Scheme 7: Flexible Public Key Scheme with Public Key in $\mathbb{G}_1$

A Group Signature Definitions

Let us recall the popular BMW model for static group signatures [4].

Definition 20 (Group Signatures). *A group signature scheme* $\mathsf{GS} = (\mathsf{KeyGen}_{\mathsf{GS}}, \mathsf{Sign}_{\mathsf{GS}}, \mathsf{Verify}_{\mathsf{GS}}, \mathsf{Open}_{\mathsf{GS}})$ *consists of the following polynomial-time algorithms:*

$\mathsf{KeyGen}_{\mathsf{GS}}(1^\lambda, n)$**:** *on input a security parameter* 1^λ *and the group size* $n \in \mathbb{N}$ *this randomized algorithm returns a tuple* $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk})$*, where* gpk *is the group public key,* gmsk *is the group manager's secret key and* gsk *is a vector of size* n *(with* $\mathsf{gsk}[i]$ *being a secret key of the* i*-th group member).*

$\mathsf{Sign}_{\mathsf{GS}}(\mathsf{gski}, m)$**:** *on input the secret key of* i*-th group member* gski *and a message* $m \in \mathcal{M}$ *this randomized algorithm returns a signature* σ_{GS} *on message* m *under* gski*.*

$\mathsf{Verify}_{\mathsf{GS}}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}})$**:** *on input the group public key* gpk*, a message* m *and a signature* σ_{GS} *this algorithm returns either* 1 *or* 0*.*

$\mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}})$**:** *on input the group manager's secret key* gmsk*, message* m *and a signature* σ_{GS} *on* m *this algorithm returns an identity* i *or the symbol* $\perp$ *in case of failure.*

For simplicity group members are assigned consecutive integer identities from the set $[n]$*.*

We say that a group signature scheme is correct if: for all $\lambda, n \in \mathbb{N}$*, all* $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \in [\mathsf{KeyGen}_{\mathsf{GS}}(1^\lambda, n)]$*, all* $i \in [n]$*, all* $m \in \mathcal{M}$ *and all* $\sigma_{\mathsf{GS}} \in [\mathsf{Sign}_{\mathsf{GS}}(\mathsf{gsk}[i], m)]$

$$\mathsf{Verify}_{\mathsf{GS}}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}}) = 1 \qquad \textit{and} \qquad \mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk}, m, \sigma_{\mathsf{GS}}) = i.$$

Compactness. We say that a group signature scheme is compact if there exist polynomials $p_1(\cdot,\cdot)$ and $p_2(\cdot,\cdot,\cdot)$ such that

$$|\mathsf{gpk}|, |\mathsf{gmsk}|, |\mathsf{gski}| \leq p_1(\lambda, \log n) \quad \wedge \quad |\sigma_{\mathsf{GS}}| \leq p_2(\lambda, \log n, |m|)$$

for all $\lambda, n \in \mathbb{N}$, all $(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{gsk}) \in [\mathsf{KeyGen}_{\mathsf{GS}}(\lambda, n)]$, all $i \in [n]$, all $m \in \mathcal{M}$ and all $\sigma_{\mathsf{GS}} \in [\mathsf{Sign}_{\mathsf{GS}}(\mathsf{gsk}[i], m)]$.

Full-Anonymity. Informally, anonymity means that it should be hard for an adversary to recover the identity of the signer from a signature without the knowledge of the group manager's secret key. To properly model collusion with group members the adversary is given the secret keys of all group members. Moreover, the adversary can use an opening oracle $\mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk}, \cdot, \cdot)$, which models the possibility of the adversary seeing previous openings.

Definition 21. *For group signature scheme* GS *and adversary* $\mathcal{A}$ *we define the following experiment:*

$\underline{\mathsf{Exp}^{\mathsf{anon}}_{\mathsf{GS},\mathcal{A}}\text{-}b(\lambda,n)}$

$(\mathsf{gpk},\mathsf{gmsk},\mathsf{gsk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{GS}}(1^\lambda,n);\ Q := \emptyset$

$(\mathsf{st},i_0,i_1,m^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{gpk},\mathsf{gsk})$

$\sigma^*_{\mathsf{GS}} \xleftarrow{\$} \mathsf{Sign}_{\mathsf{GS}}(\mathsf{gsk}[i_b],m^*)$

$\hat{b} \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\mathsf{gmsk},\cdot,\cdot)}(\mathsf{st},\sigma_{\mathsf{GS}})$

if $(m^*,\sigma^*_{\mathsf{GS}}) \in Q$ **return** 0

else return $\hat{b} = b$

$\underline{\mathcal{O}(\mathsf{gmsk},m,\sigma_{\mathsf{GS}})}$

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

return $\mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk},m,\sigma_{\mathsf{GS}})$

We say that a group signature scheme $\mathsf{GS} = (\mathsf{KeyGen}_{\mathsf{GS}},\mathsf{Sign}_{\mathsf{GS}},\mathsf{Verify}_{\mathsf{GS}},\mathsf{Open}_{\mathsf{GS}})$ *is* fully-anonymous *if for any efficient PPT algorithm* $\mathcal{A}$*, the advantage of adversary* $\mathcal{A}$ *in breaking the full-anonymity of* GS*,* $\mathbf{Adv}^{\mathsf{anon}}_{\mathsf{GS},\mathcal{A}}(\cdot,\cdot)$ *is negligible*

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{GS},\mathcal{A}}(\lambda)(\lambda,n) = |\Pr[\mathsf{Exp}^{\mathsf{anon}}_{\mathsf{GS},\mathcal{A}}\text{-}1(\lambda,n)=1] - \Pr[\mathsf{Exp}^{\mathsf{anon}}_{\mathsf{GS},\mathcal{A}}\text{-}0(\lambda,n)=1]|$$

Full-Traceability. The next required property is called full-traceability. In case of misuse, we would like the group manager to always be able to identity the signer. In particular, this means that is should not be possible to create a signature that cannot be opened. Moreover, a colluding set S of group members should not be able to frame an honest member, i.e. create a signature that opens to a member that is not in S.

Definition 22. *For group signature scheme* GS *and adversary* $\mathcal{A}$ *we define the following experiment:*

$\underline{\mathsf{Exp}^{\mathsf{trace}}_{\mathsf{GS},\mathcal{A}}(\lambda,n)}$

$(\mathsf{gpk},\mathsf{gmsk},\mathsf{gsk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathsf{GS}}(1^\lambda,n)$

$\mathsf{st} := (\mathsf{gmsk},\mathsf{gpk}); Q = \emptyset$

$\mathcal{C} = \emptyset; K = \epsilon; \mathsf{Cont} = \mathsf{true}$

while $(\mathsf{Cont} == \mathsf{true})$ **do**

  $(\mathsf{Cont},\mathsf{st},j) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\mathsf{gsk}[\cdot],\cdot)}(\mathsf{st},K)$

  if $\mathsf{Cont} == \mathsf{true}$ **then** $\mathcal{C} = \mathcal{C} \cup \{j\}$

  $K = \mathsf{gsk}[j]$

$(m^*,\sigma^*_{\mathsf{GS}}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\mathsf{gsk}[\cdot],\cdot)}(\mathsf{guess},\mathsf{st})$

if $\mathsf{Verify}_{\mathsf{GS}}(\mathsf{gpk},m^*,\sigma^*_{\mathsf{GS}}) = 0$ **then return** 0

if $\mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk},m^*,\sigma^*_{\mathsf{GS}}) = \perp$ **then return** 1

if $\exists i \in [n].\ \mathsf{Open}_{\mathsf{GS}}(\mathsf{gmsk},m^*,\sigma^*_{\mathsf{GS}}) = i \ \wedge\ i \notin C \ \wedge\ (i,m) \notin Q$

then return 1 **else return** 0

$\underline{\mathcal{O}(\mathsf{gsk}[i],m)}$

$Q := Q \cup (i,m)$

return $\mathsf{Sign}_{\mathsf{GS}}(\mathsf{gsk}[i],m)$

We say that a group signature scheme $\mathsf{GS} = (\mathsf{KeyGen}_{\mathsf{GS}},\mathsf{Sign}_{\mathsf{GS}},\mathsf{Verify}_{\mathsf{GS}},\mathsf{Open}_{\mathsf{GS}})$ *is* fully-traceable *if for any PPT algorithm* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in breaking the full-traceability of* GS*,* $\mathbf{Adv}^{\mathsf{trace}}_{\mathsf{GS},\mathcal{A}}(\cdot,\cdot)$ *is negligible:*

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

B Ring Signature Definitions

In applications such as cryptocurrencies or electronic voting it is desirable for privacy reasons, that the identity of the signer of a given message is hidden from the party interested in a valid signature. In these cases it is often enough to establish that the signer is part of a certain *group of eligible signers*. To this end, a ring signature scheme allows a signer to specify a set of additional *potential signers* and create signatures which do not reveal which signing key among this group was used to create the signature. Note, that this does not allow a signer to sign for another party, since the signature still has to be created using the signers own signing key. The intriguing property of ring signature schemes is merely that to a verifier, this information is obscured even though the signer only has access to her own signing key and just the public verification keys of the other parties in the chosen group.

Formally, we define the following scheme:

Definition 23 (Ring Signatures). *A ring signature scheme is a tuple of* PPT *algorithms* $(\mathsf{RCRSGen}, \mathsf{RKeyGen}, \mathsf{RSign}, \mathsf{RVerify})$ *such that:*

$\mathsf{RCRSGen}(1^\lambda)$**:** *takes as input the security parameter* λ *and outputs a common reference string* ρ*,*

$\mathsf{RKeyGen}(\rho, 1^\lambda)$**:** *takes as input the common reference string* ρ *and outputs a pair* $(\mathrm{SK}, \mathrm{PK})$ *of secret and public keys,*

$\mathsf{RSign}(\rho, m, \mathsf{sk}_{\mathsf{RS}}^{(s)}, \mathtt{Ring})$**:** *takes as input a message* $m \in \{0,1\}^*$*, a signing key* $\mathsf{sk}_{\mathsf{RS}}^{(s)}$ *and an ordered set (a ring) of public keys* $\mathtt{Ring} = \left(\mathsf{pk}_{\mathsf{RS}}^{(1)}, \ldots, \mathsf{pk}_{\mathsf{RS}}^{(n)}\right)$ *with* $\mathsf{pk}_{\mathsf{RS}}^{(s)} \in \mathtt{Ring}$*, and outputs a signature* Σ*,*

$\mathsf{RVerify}(\rho, m, \Sigma, \mathtt{Ring})$**:** *takes as input a message* m*, signature* Σ*, and a ring of public keys* Ring *and outputs either* $\mathtt{accept}(1)$ *or* $\mathtt{reject}(0)$*.*

A ring signature scheme is correct *if for all* $\lambda \in \mathbb{N}, n = \mathsf{poly}(\lambda)$*, all common reference strings* $\rho \xleftarrow{s} \mathsf{RCRSGen}(\lambda)$*, any* $\left\{(\mathsf{sk}_{\mathsf{RS}}^{(i)}, \mathsf{pk}_{\mathsf{RS}}^{(i)})\right\}_{i=1}^{n}$ *generated with* $\mathsf{RKeyGen}(\rho, 1^\lambda)$*, any* $s \in \{1, \ldots, n\}$ *and any message* m*, we have* $\mathsf{RVerify}(\rho, m, \mathsf{RSign}(\rho, m, \mathsf{sk}_{\mathsf{RS}}^{(s)}, \mathtt{Ring}), \mathtt{Ring}) = \mathtt{accept}$*, where* $\mathtt{Ring} = \left(\mathsf{pk}_{\mathsf{RS}}^{(1)}, \ldots, \mathsf{pk}_{\mathsf{RS}}^{(n)}\right)$*.*

In case the scheme does not require a common reference string, we omit the first argument ρ *to* $\mathsf{RKeyGen}, \mathsf{RSign}$ *and* $\mathsf{RVerify}$*.*

Ring signatures should be unforgeable with respect to the specific message that was signed and the ring of public keys that it was signed to, i.e. besides being unable to forge signatures on new messages, an adversary should also be unable to create a new signature for a known message but with a modified ring.

Definition 24 (Unforgeability w.r.t. insider corruption). *For ring signature scheme* RS *and adversary* $\mathcal{A}$ *we define the following experiment:*

$\mathsf{Unforgeability}^{\mathcal{A}}_{\mathsf{RS}}(\lambda)$

$\rho \xleftarrow{\$} \mathsf{RCRSGen}(\lambda); Q := \emptyset, C := \emptyset$
for $i = 1 \ \ldots \ l = \mathsf{poly}(\lambda)$ **do**
  $(\mathsf{sk}^{(i)}_{\mathsf{RS}}, \mathsf{pk}^{(i)}_{\mathsf{RS}}) \xleftarrow{\$} \mathsf{RKeyGen}(\rho, 1^{\lambda})$
$(m^*, \Sigma^*, \mathtt{Ring}^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign},\mathsf{Corrupt}}\left(S := \left\{\mathsf{pk}^{(i)}_{\mathsf{RS}}\right\}_{i=1}^{l}\right)$
return $\mathsf{RVerify}(\rho, m^*, \Sigma^*, \mathtt{Ring}^*) = 1 \ \wedge$
  $(m^*, \mathtt{Ring}^*) \notin Q \ \wedge$
  $\mathtt{Ring}^* \subseteq S \setminus C$

$\mathsf{Sign}(m, s, \mathtt{Ring})$

$Q := Q \cup \{(m, \mathtt{Ring})\}$
if $\mathrm{PK}_s \in \mathtt{Ring}$ **then**
  $\Sigma \xleftarrow{\$} \mathsf{RSign}(\rho, m, \mathsf{sk}^{(s)}_{\mathsf{RS}}, \mathtt{Ring})$
  return Σ
else return $\perp$

$\mathsf{Corrupt}(i)$

$C := C \cup \left\{\mathsf{pk}^{(i)}_{\mathsf{RS}}\right\}$
return $\mathsf{sk}^{(i)}_{\mathsf{RS}}$

A signature scheme RS *is* unforgeable with respect to insider corruption *if for all* PPT *adversaries* $\mathcal{A}$*, their advantage in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathsf{unforgeability}}_{\mathcal{A},\mathsf{RS}}(\lambda) = \Pr\left[\mathsf{Unforgeability}^{\mathcal{A}}_{\mathsf{RS}}(\lambda) = 1\right] = \mathsf{negl}(\lambda)\,.$$

A ring signature scheme should also be anonymous, i.e. it should be infeasible for an attacker, given a signature, to establish which ring member actually created this signature. In its strongest form, this property should hold true, even if the adversary has access to all key material (including the secret keys) of the members of the ring.

Definition 25 (Anonymity against full key exposure). *For ring signature scheme* RS *and adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *we define the following experiment:*

$\mathsf{Anonymity}^{\mathcal{A}}_{\mathsf{RS}}(\lambda)$

$\rho \xleftarrow{\$} \mathsf{RCRSGen}(\lambda)$
for $i = 1 \ \ldots \ l := \mathsf{poly}(\lambda)$ **do**
  $(\mathsf{sk}^{(i)}_{\mathsf{RS}}, \mathsf{pk}^{(i)}_{\mathsf{RS}}) \xleftarrow{\$} \mathsf{RKeyGen}(\rho, 1^{\lambda}; \omega_i)$
$(\mathsf{st}, m, i_0, i_1, \mathtt{Ring}) \xleftarrow{\$} \mathcal{A}_0^{\mathsf{Sign}}\left(\{\omega_i\}_{i=1}^{l}\right)$
if $\mathsf{pk}^{(i_0)}_{\mathsf{RS}} \notin \mathtt{Ring}$ *or* $\mathsf{pk}^{(i_1)}_{\mathsf{RS}} \notin \mathtt{Ring}$ **then**
  $\Sigma := \perp$
else $b \xleftarrow{\$} \{0, 1\}$;
  $\Sigma \xleftarrow{\$} \mathsf{RSign}(\rho, m, \mathsf{sk}^{(i_b)}_{\mathsf{RS}}, \mathtt{Ring})$
$b' \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Sign}}(\mathsf{st}, \Sigma)$
return $b = b'$

A signature scheme RS *provides* anonymity against full key exposure *if for all* PPT *adversaries* $\mathcal{A}$*, their advantage in the above experiment is negligible:*

$$\mathsf{Adv}^{\mathsf{anonymity}}_{\mathcal{A},\mathsf{RS}}(\lambda) = \left|\Pr\left[\mathsf{Anonymity}^{\mathcal{A}}_{\mathsf{RS}}(\lambda) = 1\right] - \frac{1}{2}\right| = \mathsf{negl}(\lambda)\,.$$