Signatures with Flexible Public Key: Introducing Equivalence Classes for Public Keys

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

[1] CISPA Helmholtz Center (i.G.) GmbH,
backes@cispa.saarland
[2] CISPA, Saarland University,
{hanzlik, jonas.schneider, kamil.kluczniak}@cispa.saarland
[3] Saarland Informatics Campus
[4] Department of Computing, The Hong Kong Polytechnic University

Abstract. We introduce a new cryptographic primitive called signatures with flexible public key (SFPK). We divide the key space into equivalence classes induced by a relation $\mathcal{R}$. A signer can efficiently change his or her key pair to a different representatives of the same class, but without a trapdoor it is hard to distinguish if two public keys are related. Our primitive is motivated by structure-preserving signatures on equivalence classes ($\mathsf{SPS\text{-}EQ}$), where the partitioning is done on the message space. Therefore, both definitions are complementary and their combination has various applications.

We first show how to efficiently construct static group signatures and self-blindable certificates by combining the two primitives. 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, but is secure in the same setting.

In its own right, our primitive has stand-alone applications in the cryptocurrency domain, where it can be seen as a straightforward formalization of so-called stealth addresses. Finally, it can be used to build the first efficient ring signature scheme in the plain model without trusted setup, where signature size depends only sub-linearly on the number of ring members. Thus, we solve an open problem stated by Malavolta and Schröder at ASIACRYPT'2017.

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 specific applications. 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 redactions of the signed message.

The primitive we introduce in this work allows for a relaxed characterization of authenticity instead. The goal is not complete relaxation, such that an impostor could 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.

The new primitive, called *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 are related via $\mathcal{R}$. Thus, we extend the requirement of unforgeability of signatures to the whole equivalence class of the given key under attack.

Additionally, it should be infeasible, without a trapdoor, to 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 class to which the signature belongs to, i.e., to hide their own identity in the signature (because of class-hiding). This primitive is motivated by (structure-preserving) signatures on equivalence classes [28] (SPS-EQ), where relations are defined for the message space, instead of the key space. Both notions are complementary, in the sense that we can use SPS-EQ to *certify* the public key of an SFPK scheme if the respective equivalence relations are compatible, which immediately gives so called signatures with self-blindable certificates [40].

Signatures with flexible public key are especially useful in applications 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 [17] and to some extent ring signatures [36] 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 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 with n signers, 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, which has a size of $\mathcal{O}(\sqrt{n})$ if we use general proofs and the square matrix idea for membership proofs due to Chandran, Groth and Sahai [14].

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 and their applications. Our new primitive is a natural counterpart of structure-preserving signatures on equivalence classes, but for the public key space. 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 signatures. The resulting group and ring signature schemes have smaller (asymptotic and concrete) signature sizes than the previous state of the art schemes also secure in the strongest attacker model, including schemes with non-standard assumptions.

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 [31] 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 5.3. Our ring signature construction is the first to achieve signature sizes in $\mathcal{O}(\sqrt{N})$ without trusted setup and with security under standard assumptions in the strongest security model by Bender, Katz and Morselli [5]. We also show how to efficiently instantiate the scheme using Groth-Sahai proofs and thereby we solve an open problem stated in the ASIACRYPT'2017 presentation of [33], namely: *Are there efficient ring signature schemes without trusted setup provably secure under falsifiable assumptions?*

Applications of independent interest. We also show that signatures with flexible public key which also implement a key recovery property contribute to the field of cryptocurrencies. In particular, our definitions can be seen as a formalization of the informal requirements for a technique called stealth addresses [39, 34, 37], which allows 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. Moreover, existing schemes implementing stealth addresses are based on a variant of the Diffie-Hellman protocol and inherently bound to cryptography based on the discrete logarithm problem. On the other hand, our definition is generic and SFPK can potentially be instantiated from e.g. lattice assumptions.

1.1 Related Work

At first glance, signatures with flexible public keys are syntactically reminiscent of structure-preserving signatures on equivalence classes [28]. While both primitives are similar in spirit, the former considers equivalence classes of key pairs while the latter only considers equivalence classes on messages.

There exist many primitives that allow for a limited malleability of the signed message. Homomorphic signatures [9] allow to sign any subspace of a vector space. In particular, given a number of signatures σ_i for vectors $\boldsymbol{v}_i$, everyone can compute a signature of $\sum_i \beta_i \cdot \boldsymbol{v}_i$ for scalars β_i.

Chase et al. [15] discussed malleable signatures, which allow any party knowing a signature of message m to construct a signature of message $m' = T(m)$ for some defined transformation T. One can consider malleable signatures as a generalization of quotable [2] and redactable signatures [30].

Signatures on randomized ciphertexts by Blazy et al. [7] allow any party that is given a signature on a ciphertext to randomize the ciphertext and adapt the signature to maintain public verifiability.

Verheul [40] introduces so-called self-blindable certificates. The idea is to use the same scalar to randomize the signature and corresponding message. Verheul proposed that one can view the message as a public key, which allows to preserve the validity of this "certificate" under randomization/blinding. However, the construction does not yield a secure signature scheme. We will show that combining our primitive with signatures on equivalence classes [28] can be used to instantiate self-blindable certificates.

As noted above, all the mentioned works consider malleability of the message space. In our case we consider malleability of the key space. A related primitive are signatures with re-randomizable keys introduced by Fleischhacker et al. [21]. It allows a re-randomization of signing and verification keys such that re-randomized keys share the same distribution as freshly generated keys and a signature created 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 forgery 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 [21] 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 since 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 key pair and use a signature under that key as the forgery.

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 use of random coins 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 from 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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a* bilinear map *(pairing) if it is efficiently computable and the following conditions hold:*

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 $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ 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 [20]:

– 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.*

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 [25].

Definition 4 (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 [32]. We show that this new assumption, which we call the co-Flexible Diffie-Hellman (co-Flex) assumption, holds if the decisional linear assumption holds.

Definition 5 (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.

Proof. Suppose we have an efficient algorithm $\mathcal{A}$ that solves the co-Flexible Diffie-Hellman problem with non-negligible probability. We will show how to build algorithm $\mathcal{R}$ that solves the decision linear problem. Let $(\mathsf{BG}, f_1, f_2, h_1, h_2, f_1^a, f_2^a, h_1^b, h_2^b, g_1^z, g_2^z)$ be an instance of the decision linear problem. The algorithm

$\mathcal{R}$ first runs algorithm $\mathcal{A}$ on input $(\mathsf{BG}, f_1, f_2, g_1^z, g_2^z, f_1^a, f_2^a, h_1^b, h_2^b)$. With non-negligible probability $\mathcal{A}$ outputs a solution to the co-Flexible Diffie-Hellman problem, i.e. it outputs the tuple $((f_1^a)^r, (h_1^b)^r, (f_1^z)^r)$. Then $\mathcal{R}$ computes

$$\begin{aligned} T_1 =& e((f_1^z)^r, h_2) = e(f_1, h_2^r)^z, \\ T_2 =& e((f_1^a)^r, h_2) = e(f_1, h_2^r)^a, \\ T_3 =& e((h_1^b)^r, f_2) = e(h_1, f_2^r)^b = e(f_1^r, h_2)^b, \end{aligned}$$

and outputs 1 if $T_1 = T_2 \cdot T_3$ and 0 otherwise.

2.2 Programmable Hash Functions

Programmable hash functions presented at Crypto'08 by Hofheinz and Kiltz [29] introduce a way to create hash functions with limited programmability. In particular, they show that the function introduced by Waters [41] is a programmable hash function. To formally define such function we first define so called *group hash functions* for a group $\mathbb{G}$, which consists of two polynomial time algorithms $\mathsf{PHF.Gen}$, $\mathsf{PHF.Eval}$ and has an output length of $\ell = \ell(\lambda)$. For a security parameter λ the generation algorithm $\mathsf{PHF.Gen}(\lambda)$ outputs a key K_{PHF}, which can be used in the deterministic algorithm $\mathsf{PHF.Eval}$ to evaluate the hash function via $y \xleftarrow{\$} \mathsf{PHF.Eval}(K_{\mathsf{PHF}}, X) \in \mathbb{G}$. We will use $\mathcal{H}_{K_{\mathsf{PHF}}}(X)$ to denote the evaluation of the function $\mathsf{PHF.Eval}(K_{\mathsf{PHF}}, X)$ on $X \in \{0,1\}^\ell$. We can now recall the definition of programmable has functions.

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

- *For any $g, h \in \mathbb{G}$ the trapdoor algorithm $(K'_{\mathsf{PHF}}, t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(\lambda, g, h)$ outputs a key K' and trapdoor t. Moreover, for every $X \in \{0,1\}^\ell$ we have $(a_X, b_X) \xleftarrow{\$} \mathsf{PHF.TrapEval}(t, X)$, where $\mathsf{PHF.Eval}(K'_{\mathsf{PHF}}, X) = g^{a_X} h^{b_X}$.*
- *For all $g, h \in \mathbb{G}$ and for $(K'_{\mathsf{PHF}}, t) \xleftarrow{\$} \mathsf{PHF.TrapGen}(\lambda, g, h)$ and $K_{\mathsf{PHF}} \xleftarrow{\$} \mathsf{PHF.Gen}(\lambda)$, the keys K_{PHF} and K'_{PHF} are statistically γ-close.*
- *For all $g, h \in \mathbb{G}$ and all possible keys K'_{PHF} from the range of $\mathsf{PHF.TrapGen}(\lambda, g, h)$, for all $X_1, \ldots, X_m, Z_1, \ldots, Z_n \in \{0,1\}^\ell$ such that $X_i \neq Z_j$ for any i, j and for the corresponding $(a_{X_i}, b_{X_i}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(t, X_i)$ and $(a_{Z_i}, b_{Z_i}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(t, Z_i)$ we have*

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

  where the probability is over trapdoor t that was generated with key K'_{PHF}.

Note that using this definition we can define the Waters hash function, with key $K_{\mathsf{PHF}} = (h_0, \ldots, h_\ell) \in \mathbb{G}^{\ell+1}$ and message $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ as $h_0 \cdot \prod_{i=1}^{\ell} h_i^{x_i}$. Hofheinz and Kiltz prove that for any fixed $q = q(\lambda)$ this is a $(1, q, 0, 1/8 \cdot (\ell+1) \cdot q)$-programmable hash function. Unless mentioned otherwise, we will always instantiate the programmable hash function using the Waters function and use $\ell = \lambda$.

2.3 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 [27]. 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 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 7 (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 case, we omit the first argument to Prove *and* Verify*.*

Definition 8 (Soundness). *A proof system Π is called* sound*, if for all PPT algorithms $\mathcal{A}$ the following probability, denoted by $\mathsf{Adv}^{\mathsf{sound}}_{\Pi,\mathcal{A}}(\lambda)$, is negligible in the security parameter 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 $\mathsf{Adv}^{\mathsf{sound}}_{\Pi,\mathcal{A}}(\lambda) = 0$.

Definition 9 (Witness Indistinguishability (WI)). *A proof system Π is* witness indistinguishable*, if for all PPT algorithms $\mathcal{A}$ we have that the advantage $\mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda)$ computed as:*

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

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 $\mathsf{Adv}^{\mathsf{wi}}_{\Pi,\mathcal{A}}(\lambda) = 0$.

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 [27]. For constants $A_i \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $t_T \in \mathbb{G}_T$, $\gamma_{ij} \in \mathbb{Z}_p$ which are either publicly known or part of the statement, and witnesses $X_i \in \mathbb{G}_1$, $Y_i \in \mathbb{G}_2$ given as commitments, we can prove that:

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

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

$\rho_1 := (f_1, f_2, h_1, h_2, \ldots) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{PPE}}(\lambda); r, s \xleftarrow{\$} \mathbb{Z}_p^*$

$\rho_2 := (f_1, f_2, h_1, h_2, f_1^r, f_2^r, h_1^s, h_2^s, g_1^{r+s}, g_2^{r+s})$

$\pi_{\text{Linear}} \xleftarrow{\$} \mathsf{Prove}_{\text{Linear}}((\rho_1, \rho_2), (r, s))$

$\pi_1 \xleftarrow{\$} \mathsf{Prove}_{\mathsf{PPE}}(\rho_1, x, w);\ \pi_2 \xleftarrow{\$} \mathsf{Prove}_{\mathsf{PPE}}(\rho_2, x, w)$

return $\pi := (\rho_1, \rho_2, \pi_{\text{Linear}}, \pi_1, \pi_2)$

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

parse $\pi = (\rho_1, \rho_2, \pi_{\text{Linear}}, \pi_1, \pi_2)$

return $\mathsf{Verify}_{\mathsf{PPE}}(\rho_1, x, \pi_1) = 1 \wedge$

$\mathsf{Verify}_{\mathsf{PPE}}(\rho_2, x, \pi_2) = 1 \wedge$

$\mathsf{Verify}_{\text{Linear}}((\rho_1, \rho_2), \pi_{\text{Linear}}) = 1$

Scheme 1: Perfectly Sound Proof System for Pairing Product Equations

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 [25].

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 perform the setup. However, our schemes 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 [26]. They 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 shown 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 [26]. The size 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.

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.

2.4 Structure-Preserving Signatures on Equivalence Classes

Hanser and Slamanig introduced a cryptographic primitive called structure-preserving signatures on equivalence classes [28]. Their work was further extended by Fuchsbauer, Hanser and Slamanig in [23] and [24]. 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 $\forall_{i\in[\ell]}(M_i)^r = M'_i$.

The original paper defines two properties of SPS-EQ namely unforgeability under chosen-message attacks and class-hiding. Fuchsbauer and Gay [22] recently introduced a weaker version of unforgeability called unforgeability under chosen-open-message attacks, which restricts the adversary's signing queries to messages where it knows all exponents.

Definition 10 (Signing Oracles). *A* signing oracle *is an oracle* $\mathcal{O}_{\mathsf{SPS}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$ *(resp.* $\mathcal{O}_{\mathsf{op}}(\mathsf{sk}_{\mathsf{SPS}}, \cdot)$*), 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 a 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 11 (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* $\mathcal{A}$ *with 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 advantage (with templates* T_1, T_2 *defined below) is negligible in the security parameter* λ*:*

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

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 [24]. 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 12 (Perfect Adaptation 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 motivating 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. Since there may 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 because of this the challenger needs a way to efficiently ascertain whether the forgery is valid, even if no transformation randomness is given. 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 the former also hold with respect to the latter.

Definition 13 (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:*

$\mathsf{KeyGen}(\lambda, \omega)$**:** *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* $m \in \{0,1\}^\lambda$ *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 14 (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* $i \in \{0,1\}$

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

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

$\hat{b} \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}', \cdot)}(\omega_0, \omega_1, \mathsf{pk}')$

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 15 (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 $(m^*, \cdot) \notin Q \land$

$\mathsf{ChkRep}(\tau, \mathsf{pk}') = 1 \land$

$\mathsf{Verify}(\mathsf{pk}', m^*, \sigma^*) = 1$

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

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

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

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 attack *if for all* PPT *adversaries* $\mathcal{A}$ *the advantage in the above experiment is negligible:*

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

Definition 16 (Strong Existential Unforgeability under Flexible Public Key). *A* SFPK *is* strongly existentially unforgeable with flexible public key under chosen message attack *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.*

In a standard application, the public key and secret key are jointly randomized by the signer using the same randomness in ChgPK and ChgSK. However, the ChgPK algorithm alone can be executed by a third party given only the public key and random coins r. Revealing r to the signer allows them to compute the corresponding secret key. For some applications we want to avoid interaction during this recovery of the secret key. Allowing the user to extract the new secret key only using their old secret key would break class-hiding, since the attacker in this case has access to the pre-transformed secret keys. Fortunately, we can instead use the additional trapdoor returned by the TKeyGen algorithm. More formally, we define this optional property as follows.

Definition 17 (Key Recovery). *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 part of each user's public key is shared with all the other public keys and is precomputed by a trusted third party in a setup phase, e.g. the key used in a programmable hash function. 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 14 and 15). 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 14 the adversary is allowed to generate the string ρ. Similarly, we call a SFPK scheme *unforgeable under malicious reference string* if the unforgeability definition 15 holds if ρ is generated by the adversary.

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 schemes 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 an SFPK scheme such that it improves on the respective state-of-the-art in terms of concrete efficiency, necessary assumptions or both.

4.1 Cryptocurrency Stealth Addresses

In cryptocurrency systems transactions are confirmed through digital signatures from the spending party on, among other things, the public key of the receiving party. Using a technique called stealth addresses [39, 37], it is possible for the sender to create a fresh public key (address) 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, subsequently enabling it to spend any funds send to the fresh unlinkable key. Crucially, there is no interaction necessary between sender and receiver to establish the fresh key and only the receiver can recover the right secret key.

Informally, a sender can take a recipient's public address and transform it to a one-time address such that:

- The new one is unlinkable to the original one and other one-time addresses,
- only the recipient (or a party given the view key) can link all payments,
- only the recipient can derive the spending key for the one-time address.

In existing schemes, stealth addresses are implemented using a variant of the Diffie-Hellman protocol [37, 19]. Let g^a be the public key of the sender and g^b the recipient's public address. The sender computes the secret $s = \mathsf{H}(g^{a \cdot b})$ and to finish the transaction sends the funds to the address g^s. Note that this requires the recipient to immediately spend the coins, because the sender also knows s. To protect against this type of misuse, an asymmetric Diffie-Hellman was introduced, i.e. the funds are sent to the address $g^{s+b} = (g)^s \cdot g^b$. Note that since only the recipient knows both s and b, only he can spend the money.

In practice, the sender's public key g^a is ephemeral and unique for each transaction. Moreover, to increase efficiency a 2-key stealth address scheme was introduced. The recipient still holds the key for spending the coin, but gives a view key g^v to a third party for checking incoming transactions. Therefore, the recipient is not required to download all transactions and check if they correspond to their identity. However, the party holding the view key can break the anonymity of the recipient. To enable this feature, the sender also publishes $(g^v)^a$, as part of this transaction.

It is worth noting that the technique was introduced without a formal model and as an add-on for existing cryptocurrencies. In particular, as shown in [19] there exist many security pitfalls, which are exhibited by some of the schemes. Moreover, all existing schemes inherently rely on the Diffie-Hellman protocol, which is defined for groups in which the discrete logarithm is hard.

We will now show that signatures with flexible public keys that additionally implement the Recover algorithm can be seen as a formalization of 2-key stealth addresses. Let us consider the following scenario. A sender wants to send funds to a recipient identified by an address pk, where $(\mathsf{sk}, \mathsf{pk}, \tau) \xleftarrow{\$} \mathsf{TKeyGen}(\lambda, \omega)$. In order to send the coins, the sender first chooses randomness r and computes the one-time address $\mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}, r)$. The trapdoor τ can be used as the view key to identify an incoming transaction using $\mathsf{ChkRep}(\tau, \mathsf{pk}')$. Finally, the recipient can use $\mathsf{Recover}(\mathsf{sk}, \tau, \mathsf{pk}')$ to compute the secret key sk' that can be used to spend funds sent to address pk'.

The main advantage of instantiating 2-key stealth addresses using SFPK is that we can use the security arguments of the latter. In particular, unforgeability of SFPK means that there cannot exist an efficient adversary that can spend the recipient's coins. Note that this holds even if the adversary knows the view key τ. Privacy of the recipient is protected by class-hiding. Since the distributions of TKeyGen and KeyGen are identical, it follows that any adversary breaking privacy would break class-hiding. The party holding the view key τ can distinguish transactions by definition, hence class-hiding does not hold for this party.

It is worth noting, that all previous descriptions of stealth addresses did not consider any formal model and rigorous proofs. As we have argued above, our definition of SFPK with key recovery seems to directly address the requirements set before stealth addresses. Thus, our schemes are provable secure realizations of a stealth address scheme. Moreover, since we do not use a particular group structure, our construction could be instantiated using e.g. lattice-based cryptography. We leave an instantiation of SFPK from lattices as an open problem.

Finally, note that Scheme 4 is an instance of signatures with flexible public key which has the required recovery algorithm. We also show how to extend Schemes 5 and 6 to support it.

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

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

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

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

$\mathsf{Verify}_{\mathsf{GS}}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}})$

parse $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma, \sigma'_{\mathsf{SPS}})$
    $\mathsf{gpk} = (\mathsf{BG}, \mathsf{pk}_{\mathsf{SPS}}, \rho)$
if $\mathsf{Verify}_{\mathsf{SPS}}(\mathsf{pk}', \sigma'_{\mathsf{SPS}}, \mathsf{pk}_{\mathsf{SPS}}) = 0$ **then return** 0
$M := m||\sigma'_{\mathsf{SPS}}||\mathsf{pk}'$
return $\mathsf{Verify}(\mathsf{pk}', M, \sigma)$

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

parse $\sigma_{\mathsf{GS}} = (\mathsf{pk}', \sigma, \sigma'_{\mathsf{SPS}})$
    $\mathsf{gmsk} = ([(\tau^i, \mathsf{pk}^i)]^n_{i=1})$
if $\mathsf{Verify}_{\mathsf{GS}}(\mathsf{gpk}, m, \sigma_{\mathsf{GS}}) = 0$ **then return** $\perp$
if $\nexists i \in [n]$ such that $\mathsf{ChkRep}(\tau^i, \mathsf{pk}^i, \mathsf{pk}') = 1$
 then return $\perp$ **else return** i

Scheme 2: Generic Group Signature Scheme

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 their 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.

To enable subsequent opening, the group manager generates the SFPK keys using TKeyGen and stores their trapdoors. Opening is then performed using the stored trapdoors with the ChkRep algorithm.The group manager can also generate $\rho \xleftarrow{\$} \mathsf{CRSGen}$ 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. If the KeyGen algorithm is used instead of TKeyGen to compute the SFPK key pairs, there is no efficient opening procedure and the combination of SFPK and SPS-EQ signature scheme yields a self-blindable certificate scheme [40].

Theorem 3. *Scheme 2 is fully traceable if the* SPS-EQ *and the* SFPK *signature schemes are existentially unforgeable under chosen-message attack.*

Proof (Sketch). The proof relies on the fact that the only way for an adversary to win the full traceability game is by either creating a new group member (thus directly breaking the unforgeability of the SPS-EQ scheme) or by creating a forged signature for an existing group member (thus breaking the unforgeability of the SFPK scheme).

Theorem 4. *Scheme 2 is fully anonymous if the* SPS-EQ *signature scheme perfectly adapts signatures and is existentially unforgeable under chosen-message attacks, the* SFPK *scheme is class-hiding and strongly existentially unforgeable.*

Proof (Sketch). We first use the perfect adaptation of SPS-EQ signatures to re-sign the public key pk' used in the challenge signature. Then we exclude the case that the adversary issues an open query that cannot be opened. This means that the adversary created a new group member and can be used to break the unforgeability of the SPS-EQ scheme. In the next step we choose one of the users (and abort if he is not part of the query issued by the adversary to the challenge oracle) for which we change the way we generate the secret key. Instead of using TKeyGen, we use the standard key generation algorithm KeyGen. Note that in such a case, the open oracle cannot identify signatures created by this user. However, since signatures cannot be opened by the oracle for this user we can identify such a case and return his identifier. Finally, we replace the SFPK public key and signature in the challenged group signature by a random one (which is indistinguishable by class-hiding). In the end the challenged signature is independent from the bit $\hat{b}$. However, the adversary still has non-zero advantage. This follows from the fact that it can randomize the challenged signature and our oracle will output $i_{\hat{b}}$ (because the SFPK public key is random in the signature, the oracle will fail to open and return the user's identifier). However, if the adversary is able to submit such a query we can break the strong existential unforgeability of the SFPK scheme.

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, to certify ring members we use a membership proof instead of a SPS-EQ signature. This proof 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 $\mathit{\Pi}$ 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 $\mathit{\Pi}$, since it does not choose the random coins for the challenged signature. Thus, 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 $\mathit{\Pi}$ if (A, B, C) is a non-DDH tuple (cf. Scheme 3). We can instantiate this scheme with a membership proof based on the $\mathcal{O}(\sqrt{n})$ size ring signatures by Chandran, Groth, Sahai [14] and the perfectly sound proof system for NP languages by Groth, Ostrovsky, Sahai [26]. 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 ring signature from standard assumptions without a trusted setup.

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

$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KeyGen}(\lambda, \omega)$

$I := (A, B, C) \xleftarrow{\$} \mathbb{G}_1^3$

$\textbf{return } (\mathsf{pk}_{\mathsf{RS}} := (\mathsf{pk}, I), \mathsf{sk}_{\mathsf{RS}} := \mathsf{sk})$

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

$\textbf{parse } \Sigma = (\mathsf{pk}', \sigma, \Pi, \rho_\Pi)$

$\textbf{return } \mathsf{Verify}(x, \Pi) \wedge$

$\qquad \mathsf{Verify}(\mathsf{pk}', m, \sigma)$

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

$r \xleftarrow{\$} \mathbb{Z}_p^*;\ \mathsf{sk}' \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}, r);\ \mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}, r)$

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

$\Pi \xleftarrow{\$}$ $\mathsf{Prove}(x, (\mathsf{pk}, r))$ where x is statement

$\exists_{i,\mathsf{pk},r} \big((i, \mathsf{pk}, \cdot) \in \mathtt{Ring} \ \wedge\ \mathsf{ChgPK}(\mathsf{pk}, r) = \mathsf{pk}'\big) \ \vee$

$\exists_{i,I} \left((i, \cdot, I) \in \mathtt{Ring} \ \wedge\ I \text{ is not a DDH tuple}\right)$

$\textbf{return } \Sigma := (\mathsf{pk}', \sigma, \Pi)$

Scheme 3: Generic Ring Signature Scheme

Theorem 5. *The generic construction of ring signatures presented in Scheme 3 is unforgeable w.r.t. insider corruption assuming the* SFPK *scheme is existentially unforgeable, the proof system used is perfectly sound and the decisional Diffie-Hellman assumption holds.*

Proof (Sketch). We first fix all public keys of honest users to contain only DDH tuples. This ensures that the forgery $\Sigma^* = (\mathsf{pk}^*, \sigma^*, \Pi^*, \rho^*_\Pi)$ includes a perfectly sound proof for the first clause of the statement, i.e. there exists a public key $\mathsf{pk} \in \mathtt{Ring}$, which is in relation to pk^* (all users in $\mathtt{Ring}$ must be honest). This enables us to break existential unforgeability of the SFPK scheme. Note that we have to guess the correct user to execute a successful reduction.

Theorem 6. *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 (Sketch). We first fix all public keys of honest users to contain only non-DDH tuples I. In the next step we randomly choose a fresh bit $\hat{b} \xleftarrow{\$} \{0, 1\}$ and use the witness for the tuple $I_{i_{\hat{b}}}$ in the challenged signature. Note that the proof is valid for both values of $\hat{b}$ but now the proof part is independent from the bit b. Next we change the SFPK scheme public key pk' and signature σ returned as part of the challenged signature $\Sigma = (\mathsf{pk}', \sigma', \Pi)$. Again we choose a fresh bit $\hat{b} \xleftarrow{\$} \{0, 1\}$ and compute them using $\mathsf{pk}' \xleftarrow{\$} \mathsf{ChgPK}(\mathsf{pk}_{i_{\hat{b}}}, r)$, $\mathsf{sk}' \xleftarrow{\$} \mathsf{ChgSK}(\mathsf{sk}_{i_{\hat{b}}}, r)$ and $\sigma \xleftarrow{\$} \mathsf{Sign}(\mathsf{sk}', m||\mathtt{Ring})$. Any adversary distinguishing this change can be used to break the class-hiding property of the SFPK scheme. Finally, all elements of Σ are independent from b and the adversary's advantage is zero.

5 Efficient Instantiation from Standard Assumptions

In this section we present two 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 in the plain model scheme (i.e. without a common reference string) the public key contains the implicit security parameter λ and parameters BG. Since the bilinear-group generation algorithm $\mathsf{BGGen}(\lambda)$ is deterministic, it follows that this does not influence the class-hiding property or the unforgeability property. Therefore, for readability we omit those parameters.

The first instantiation is based on a modified version of Waters signatures [41] for type-2 and type-3 pairings due to Chatterjee and Menezes [16]. The scheme has the key recovery property and can hence be used to implement stealth addresses and instantiate our ring signature construction.

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. A strongly unforgeable variant of this scheme is ideal for instantiating the group signature scheme presented in Section 4. In combination with the $\mathsf{SPS\text{-}EQ}$ from [22] it results in the shortest static group signature scheme under standard assumptions. Further, using type-2 pairing and the random oracle model allows to use this scheme without a trusted party.

5.1 Warm-up Scheme

Theorem 7. *Scheme 4 is existentially unforgeable under flexible public key, assuming the decisional linear assumption holds and that* PHF *is* $(1, \mathsf{poly}(\lambda))$

Proof. In this particular proof we assume that we can re-run $\mathsf{PHF.TrapGen}$ using the same random coins on a different group, i.e. that we can generate key $K_{\mathsf{PHF}} = (g_1^{\mu_0}, \ldots, g_1^{\mu_\lambda}) \in \mathbb{G}_1^{\lambda+1}$ and a corresponding key $K'_{\mathsf{PHF}} = (g_2^{\mu_0}, \ldots, g_2^{\mu_\lambda}) \in \mathbb{G}_2^{\lambda+1}$. Note that this means that we make non-blackbox use of the underlying programmable hash function, but this re-running is possible for the hash function we use, i.e. the Waters hash function.

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 that has non-negligible advantage $\mathsf{Adv}_{\mathcal{A},\mathsf{SFPK}}^{\mathsf{euf-cma}}(\lambda)$. 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 $\mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, K_{\mathsf{PHF}})$ as follows. It sets:

$$X = g_1^\gamma \qquad A = f_1^\alpha \qquad B = h_1^\beta$$

$$C = h_1 \qquad t = e(X, f_2) = e(X^\phi, g_2) \qquad D = X^d$$

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

$K_{\mathsf{PHF}} \xleftarrow{\$} \mathsf{PHF.Gen}(\lambda) \in \mathbb{G}_1^{\lambda+1}$
$A, B, C, D, X \xleftarrow{\$} \mathbb{G}_1 \; y \xleftarrow{\$} \mathbb{Z}_p^*$
$t \leftarrow e(X^y, g_2)$
$\textbf{return } (\mathsf{pk}_{\mathsf{FW}} := (A, B, C, D, t, K_{\mathsf{PHF}}),$
$\quad \mathsf{sk}_{\mathsf{FW}} := (y, X, \mathsf{pk}_{\mathsf{FW}}))$

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

$K_{\mathsf{PHF}} \xleftarrow{\$} \left(g_1^{\mu_i} \mid i \in \{0, \ldots, \lambda\}, \mu_i \xleftarrow{\$} \mathbb{Z}_p\right)$
$a, b, c, d, x \xleftarrow{\$} \mathbb{Z}_p^* \; y \xleftarrow{\$} \mathbb{Z}_p^*$
$t \leftarrow e(g_1^{x \cdot y}, g_2)$
$\textbf{return } (\mathsf{pk}_{\mathsf{FW}} := (g_1^a, g_1^b, g_1^c, g_1^{x \cdot d}, t, K_{\mathsf{PHF}})$
$\quad \mathsf{sk}_{\mathsf{FW}} := (y, g_1^x, \mathsf{pk}_{\mathsf{FW}}),$
$\quad \tau := (d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\mu_0}, g_2^{\mu_1}, \ldots, g_2^{\mu_\lambda}))$

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

$\textbf{parse } \mathsf{sk}_{\mathsf{FW}} = (y, X, \mathsf{pk}_{\mathsf{FW}})$
$r \xleftarrow{\$} \mathbb{Z}_p^*$
$\textbf{return}$
$\quad \sigma_{\mathsf{FW}} := \left(X^y \cdot (\mathcal{H}_{K_{\mathsf{PHF}}}(m))^r, g_1^r, g_2^r\right)$

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

$\textbf{parse } \sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$
$\quad \mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, K_{\mathsf{PHF}})$
$\textbf{return } e(\sigma_{\mathsf{W}}^2, g_2) = e(g_1, \sigma_{\mathsf{W}}^3) \wedge$
$\quad e(\sigma_{\mathsf{FW}}^1, g_2) = t \cdot e\left(\mathcal{H}_{K_{\mathsf{PHF}}}(m), \sigma_{\mathsf{FW}}^3\right)$

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

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

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

$\textbf{parse } \mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, K_{\mathsf{PHF}})$
$\textbf{return } \mathsf{pk}_{\mathsf{FW}}' := (A^r, B^r, C^r, D^r, t^r, (K_{\mathsf{PHF}})^r)$

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

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

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

$\textbf{parse } \mathsf{sk} = (y, g_1^x, \mathsf{pk})$
$\quad \tau = (d, g_2^y, g_2^a, g_2^b, g_2^c, g_2^{\mu_0}, \ldots g_2^{\mu_\lambda})$
$\quad \mathsf{pk}' = (A^r, B^r, C^r, D^r, t^r, (K_{\mathsf{PHF}})^r)$
$X' \leftarrow (D^r)^{1/d}$
$\textbf{return } \mathsf{sk}' := (y, X', \mathsf{pk}')$

Scheme 4: Warm-up Scheme for Waters Signatures

and $(K_{\mathsf{PHF}}, \tau_{\mathsf{PHF}}) \xleftarrow{\$} \mathsf{PHF.TrapGen}(\lambda, g_1^\gamma, g_1)$. The reduction also prepares the trapdoor $\tau = (d, f_2, f_2^\alpha, h_2^\beta, h_2, K'_{\mathsf{PHF}})$, where to generate K'_{PHF} we re-run the algorithm $\mathsf{PHF.TrapGen}(\lambda, g_2^\gamma, g_2)$.

Let (m, l) be one of $\mathcal{A}'s$ signing queries. To answer it, $\mathcal{R}$

- chooses random values $t \xleftarrow{\$} \mathbb{Z}_p^*$,
- it computes $(a_m, b_m) \xleftarrow{\$} \mathsf{PHF.TrapEval}(\tau_{\mathsf{PHF}}, m)$ and aborts if $a_m = 0$,
- it computes $\mathsf{pk}_{\mathsf{FW}}' \xleftarrow{\$} \mathsf{ChgPK}_{\mathsf{SFPK}}(\mathsf{pk}_{\mathsf{SFPK}}, l)$,
- it computes:

$$\sigma_{\mathsf{FW}}^1 = (g_1^\gamma)^{t \cdot l \cdot a_m} \cdot ((f_1)^{(-a_m^{-1})} \cdot g_1^t))^{l \cdot b_m},$$
$$\sigma_{\mathsf{FW}}^2 = (f_1)^{-a_m^{-1}} \cdot g_1^t, \qquad \sigma_{\mathsf{FW}}^3 = (f_1)^{-a_m^{-1}} \cdot g_2^t,$$

- it returns the signature $\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$.

Let $f_1 = g_1^\phi$. We will now show that this is a valid signature. Note that the a valid signature is of the form $(f_1^{\gamma \cdot l} \cdot ((g_1^\gamma)^{a_m} \cdot g_1^{b_m})^{l \cdot r}, g_1^r, g_2^r)$. In this case, the reduction has set $r = -a_m^{-1} \cdot \phi + t$ and this means that the $f_1^{\gamma \cdot l}$ cancels out and the reduction does not need to compute f_1^γ.

Finally, $\mathcal{A}$ will output a valid signature under message m^*: $\hat{\sigma_{\mathsf{FW}}} = (\hat{\sigma_{\mathsf{FW}}}^1, \hat{\sigma_{\mathsf{FW}}}^2, \hat{\sigma_{\mathsf{FW}}}^3) = ((g_1^{\gamma\cdot\phi}\mathcal{H}_{K_{\mathsf{PHF}}}(m^*)^{r^*})^{l^*}, g_1^{r^*}, g_2^{r^*})$, for which we hope that $a_{m^*} = 0$, where $(a_{m^*}, b_{m^*}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(\tau_{\mathsf{PHF}}, m^*)$. Moreover, since this should be a valid forgery then we have that this signature is under 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 $\hat{\sigma_{\mathsf{FW}}} = ((f_1^{\gamma}(g_1^{r^*})^{b_{m^*}})^{l^*}, g_1^{r^*}, g_2^{r^*})$, for some unknown r^* but known b_{m^*}. Since $(\mathsf{pk}_{\mathsf{FW}}, \hat{\mathsf{pk}_{\mathsf{FW}}}) \in \mathcal{R}$. This means that $\hat{\mathsf{pk}_{\mathsf{FW}}} = (A^{l^*}, B^{l^*}, C^{l^*}, D^{l^*}, t^{l^*}, K_{\mathsf{PHF}}^{l^*}) = ((f_1^{\alpha})^{l^*}, (h_1^{\beta})^{l^*}, (h_1)^{l^*}, (g_1^{\gamma\cdot d})^{l^*}, t^{l^*}, K_{\mathsf{PHF}}^{l^*})$. We now compute

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

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 $a_{m^*} = 0$). However, since we assume that the used hash function is a $(1, \mathsf{poly}(\lambda))$-programmable hash function, it follows that $\mathcal{R}$ has a non-negligible advantage in solving the decisional linear problem.

Theorem 8. *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 $\mathbf{GAME}_i$. We will also use the vector $\boldsymbol{u}$ to denote the key for the programmable hash function K_{PHF}. Let $\mathsf{pk}_{\mathsf{FW}}' = (A', B', C', D', t', \boldsymbol{u'})$ 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.

$\mathbf{GAME}_1$: 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 $\boldsymbol{u} = (g_1^{\nu_0}, \ldots, g_1^{\nu_\lambda})$. Moreover, we change the way $\mathsf{sk}_{\mathsf{FW}}'$ and $\mathsf{pk}_{\mathsf{FW}}'$ are computed from $\mathsf{sk}_{\mathsf{FW}_{\hat{b}}}$ $\mathsf{pk}_{\mathsf{FW}_{\hat{b}}}$, i.e. $\mathsf{pk}_{\mathsf{FW}}' = (Q^a, Q^b, Q^c, Q^d, e(Q^x, g_2^y), (Q^{\nu_0}, \ldots, Q^{\nu_\lambda}))$, and $\mathsf{sk}_{\mathsf{FW}}' = (y, Q^x, \mathsf{pk}_{\mathsf{FW}}')$. 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 $\Pr[S_1] = \Pr[S_0]$. Note that since the secret key $\mathsf{sk}_{\mathsf{FW}}'$ is known, the signing oracle $\mathsf{Sign}(\mathsf{sk_{FW}}', \cdot)$ can be properly simulated for any adversary.

GAME$_2$: In this game instead of computing $\mathsf{pk_{FW}}' = (Q^a, Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_0}, \ldots, Q^{\nu_\lambda}))$ as in **GAME**$_1$, we sample $A' \xleftarrow{\$} \mathbb{G}_1$ set $\mathsf{pk_{FW}}' = (A', Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_0}, \ldots, Q^{\nu_\lambda}))$.

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

$$\mathsf{pk_{FW}}' = ((g_1^\gamma)^{r_{\hat{b},A}}, Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_0}, \ldots, Q^{\nu_\lambda})).$$

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

GAME$_3$ (series of sub-games): In this game instead of computing $\mathsf{pk_{FW}}' = (A', Q^b, Q^c, Q^d, e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (Q^{\nu_1}, \ldots, Q^{\nu_\lambda}))$ as in **GAME**$_2$, we sample $B', C', D', u'_0, \ldots, u'_\lambda \xleftarrow{\$} \mathbb{G}_1$ and set $\mathsf{pk_{FW}}' = (A', B', C', D', e(Q^{x_{\hat{b}}}, g_2^{y_{\hat{b}}}), (u'_0, \ldots, u'_\lambda))$.

This transition is composed of a number of sub-games, in which we change each element of the public key $\mathsf{pk_{FW}}'$ separately. Obviously, we can use the same reduction as above and show that each change lowers the adversary's advantage by at most $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$. It is worth noting, that the reduction can always create a valid signature, since the secret key $\mathsf{sk_{FW}}' = (y_{\hat{b}}, Q^{x_{\hat{b}}}, \mathsf{pk_{FW}}')$ can be computed by $\mathcal{R}$. Thus, we have $|\Pr[S_3] - \Pr[S_2]| \leq (4 + \lambda) \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$.

Let us now take a look at the randomized public key and signature given to the adversary. Because of all the changes, we have: $\mathsf{pk_{FW}}' = (A', B', C' e(Q^{x_{\hat{b}} \cdot y_{\hat{b}}}, g_2), \boldsymbol{u'})$ and signatures from the oracle are of the form $(Q^{x_{\hat{b}} \cdot y_{\hat{b}}}(\mathcal{H}_{K_{\mathsf{PHF}}}(m))^r, g_1^r, g_2^r)$ for some $r \in \mathbb{Z}_p^*$ and $A', B', C', \boldsymbol{u'}(= K_{\mathsf{PHF}}), Q$, 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 $\mathsf{Adv}_{\mathcal{A},\mathsf{SFPK}}^{\mathsf{c\text{-}h}}(\lambda) = \Pr[S_0] \leq (5 + \lambda) \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$.

5.2 Flexible Public Key Scheme in the Multi-user Setting

Theorem 9. *Scheme 5 is existentially unforgeable under flexible public key in the common reference string model, assuming the co-Flexible Diffie-Hellman assumption holds and that* PHF *is a* $(1, \mathsf{poly}(\lambda))$*-programmable hash function.*

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

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

$K_{\mathsf{PHF}} \xleftarrow{\$} \mathsf{PHF.Gen}(\lambda) \in \mathbb{G}_1^{\lambda+1}$

$y \xleftarrow{\$} \mathbb{Z}_p^*; Y_1 \leftarrow g_1^y; Y_2 \leftarrow g_2^y$

$\textbf{return } \rho := (\mathsf{BG}, Y_1, Y_2, K_{\mathsf{PHF}})$

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

$A, B \xleftarrow{\$} \mathbb{G}_1; x \xleftarrow{\$} \mathbb{Z}_p^*$

$\textbf{return } (\mathsf{pk}_{\mathsf{FW}} := (A, B, g_1^x)$

$\mathsf{sk}_{\mathsf{FW}} := (Y_1^x, \mathsf{pk}_{\mathsf{FW}}))$

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

$a, b, x \xleftarrow{\$} \mathbb{Z}_p^*$

$\textbf{return } (\mathsf{pk}_{\mathsf{FW}} := (g_1^a, g_1^b, g_1^x),$

$\mathsf{sk}_{\mathsf{FW}} := (Y_1^x, \mathsf{pk}_{\mathsf{FW}}),$

$\tau := (g_2^a, g_2^b, g_2^x))$

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

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

$r \xleftarrow{\$} \mathbb{Z}_p^*$

$\textbf{return}$

$\sigma_{\mathsf{FW}} := (Z \cdot (\mathcal{H}_{K_{\mathsf{PHF}}}(m))^r, g_1^r, g_2^r)$

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

$\textbf{parse } \mathsf{pk}_{\mathsf{FW}} = (A, B, X)$

$\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$

$\textbf{return } e(\sigma_{\mathsf{W}}^2, \hat{g}_2) = e(\hat{g}_1, \sigma_{\mathsf{W}}^3) \wedge$

$e(\sigma_{\mathsf{FW}}^1, \hat{g}_2) = e(X, Y_2) \cdot e(\mathcal{H}_{K_{\mathsf{PHF}}}(m), \sigma_{\mathsf{FW}}^3)$

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

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

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

$\textbf{return}$

$\mathsf{sk}_{\mathsf{FW}}' := ((Z)^r, \mathsf{pk}_{\mathsf{FW}}')$

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

$\textbf{parse } \mathsf{pk}_{\mathsf{FW}} = (A, B, X)$

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

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

$\textbf{parse } \tau = (\tau_1, \tau_2, \tau_3)$

$\mathsf{pk}_{\mathsf{FW}}' = (\mathsf{pk}_1, \mathsf{pk}_2, \mathsf{pk}_3)$

$\textbf{return}$

$\bigwedge_{i \in [3]} \bigwedge_{j \in [3]} e(\mathsf{pk}_i, \tau_j) = e(\mathsf{pk}_j, \tau_i)$

Scheme 5: Multi-user Flexible Public Key

Proof (Sketch). The proof follows the same idea as the proof of Theorem 7. The only difference is that in this case we will use a reduction directly to the co-Flexible Diffie-Hellman assumption. 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 this problem. The reduction $\mathcal{R}$ prepares the common reference string $\rho = (\mathsf{BG}, Y_1, Y_2, K_{\mathsf{PHF}})$ and the public key $\mathsf{pk}_{\mathsf{FW}} = (A, B, X)$ as follows. It sets $X = g_1^\beta$, $Y_1 = g_1^\alpha$, $Y_2 = g_2^\alpha$, $A = g_1^\gamma$, $B = g_1^\theta$ and $(K_{\mathsf{PHF}}, \tau_{\mathsf{PHF}}) \xleftarrow{\$} \mathsf{PHF.TrapGen}(\lambda, g_1^\beta, g_1)$. Moreover, $\mathcal{R}$ sets $\tau = (g_2^\gamma, g_2^\theta, g_2^\beta)$. Finally, the adversary $\mathcal{A}$ will output a public key $\hat{\mathsf{pk}_{\mathsf{FW}}} = (A^{l^*}, B^{l^*}, X^{l^*})$ and a valid signature under message m^*: $\hat{\sigma_{\mathsf{FW}}} = ((g_1^{\alpha \cdot \beta})^{l^*} (g_1^{r^*})^{b_{m^*}}, g_1^{r^*}, g_2^{r^*})$, for some unknown r^* but known b_{m^*}. The reduction can compute $S = (g_1^{\alpha \cdot \beta})^{l^*}$ and return (A^{l^*}, B^{l^*}, S) as a solution to the co-Flexible Diffie-Hellman problem.

Theorem 10. *Scheme 5 is class-hiding under the DDH assumption in $\mathbb{G}_1$.*

Proof (Sketch). The proof is analogous to the proof of Theorem 8.

Remark 1 (Key Recovery). To support key recovery, the public key must be extended to the form (A, B, C, X) for $C = Y_1^c$. The value c is then part of τ and can be used to restore the value Y_1^r, where r is the randomness 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 $\mathsf{sk}_{\mathsf{FW}} = (x, Y_1^x) = (x, Z)$.

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

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

$r \xleftarrow{\$} \mathbb{Z}_p^*; s \xleftarrow{\$} \mathbb{Z}_p^*$

$v \leftarrow \mathsf{H}(m, g_1^r, g_2^r, \mathsf{pk}_{\mathsf{FW}}) \in \mathbb{Z}_p^*$

$M \leftarrow g_1^v h^s$

$\textbf{return } \sigma_{\mathsf{FW}} := (Z \cdot (\mathcal{H}_{K_{\mathsf{PHF}}}(M))^r, g_1^r, g_2^r, s)$

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

$\textbf{parse } \mathsf{pk}_{\mathsf{FW}} = (A, B, X)$

$\sigma_{\mathsf{FW}} = (\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3, s)$

$v \leftarrow \mathsf{H}(m, g_1^r, g_2^r, \mathsf{pk}_{\mathsf{FW}})$

$M \leftarrow g_1^v h^s$

$\textbf{return } e(\sigma_{\mathsf{W}}^2, \hat{g}_2) = e(\hat{g}_1, \sigma_{\mathsf{W}}^3) \wedge$

$e(\sigma_{\mathsf{FW}}^1, \hat{g}_2) = e(X, Y_2) \cdot e(\mathcal{H}_{K_{\mathsf{PHF}}}(M), \sigma_{\mathsf{FW}}^3)$

Scheme 6: Strong Existential Unforgeable Variant of Scheme 5

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

A generic transformation from existentially unforgeable to strongly unforgeable signatures was proposed by Boneh, Shen and Waters [10]. In particular, they use Waters signatures as a case study. It works for all schemes 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 these properties hold for standard Waters signatures and for Scheme 5, since we can compute $\sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3$ in algorithm F_2 and σ_{FW}^1 in F_1. What is more, once the random value r is set, there exists exactly one value σ_{FW}^1, for which $(\sigma_{\mathsf{FW}}^1, \sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$ 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 the output is signed. In a scenario where we consider a given public key, this means that the signature cannot be randomized. Any manipulation of the values $(\sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$ would result in a different signed message, which would lead to an attack against existential unforgeability of the underlying scheme. Fixing $(\sigma_{\mathsf{FW}}^2, \sigma_{\mathsf{FW}}^3)$ fixes σ_{FW}^1, 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_{FW}^1 by randomizing the public key. We can overcome this by simply including the public key in the hash computation.

This 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 equivocation of the commitment. 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 an element $h \xleftarrow{\$} \mathbb{G}_1$. This element is part of the commitment key for the Pedersen scheme. More details are given in Scheme 6.

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	$(\lambda + 5)$	1	2	1	-	-	-	-	DLIN + DDH	✓
5	3	-	2	1	-	-	$(\lambda + 2)$	1	co-Flex (or DLIN) + DDH	✗/✓[†]
6	3	-	2	1	-	1	$(\lambda + 3)$	1	co-Flex + DDH + CRHF	✗/✓[†]

[†] Can support key recovery at an expense of a larger public key (one element in $\mathbb{G}_1$).

Fig. 1. Comparison of Presented Instantiations

Theorem 11. *Scheme 6 is strongly existentially 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 [10].

Theorem 12. *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 10.

5.3 Discussion

In this we instantiate the generic group signature Scheme 2 and the generic ring signature Scheme 3 with our SFPK instantiations.

Note that in the case of group signatures we can use a SFPK scheme that is strongly existentially unforgeable in the multi-user setting, since the group manager can be trusted to perform a proper setup of public parameters. Thus, a natural candidate is Scheme 6. We also require a SPS-EQ signature scheme, which we instantiate using the scheme presented in [22]. A caveat to this scheme is that it only supports a one-time adaptation of signatures to a different representative. This does not impact our use of 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, hence we require the following lemma.

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 3 and 4 still hold if the* SPS-EQ *scheme is only existential unforgeable under adaptive chosen-open-message attacks.*

Proof (Sketch). In the proof of Theorem 3, instead of excluding the case where the adversary creates a new user, we can toss a coin and chose the adversary's

	Scheme	Signature size* [$bits$]	Key size* [$bits$ gpk + $bits$ gmsk]	Anonymity	Assumptions
Random Oracle	Camenisch-Groth [13]	13 568	26 112 + $\mathcal{O}(\lambda)$	full	standard
	Boneh-Boyen-Shacham [8]	2 304	2 048 + 512	CPA-full	q-type
	Bichsel et al. [6]†	1 280	1 024 + 512	no key exposure	interactive
No Random Oracle	Boyen-Waters [12]	18 432	$\mathcal{O}(\lambda) + 6144$	CPA-full	q-type
	Boyen-Waters [12]‡	6 656	$\mathcal{O}(\lambda) + 512$	CPA-full	q-type
	Libert-Peters-Yung [31]	8 448	18 688¶ + 256	full	standard
	Ours with [23]	3 072	$\mathcal{O}(\lambda) + \mathcal{O}(n)$	full	interactive
	Ours with [22]	7 680	$\mathcal{O}(\lambda) + \mathcal{O}(n)$	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 [31]; ¶ A chameleon hash key excluded.

Fig. 2. Comparison of Static Group Signature Schemes

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 4 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, 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-blackbox 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 in [31].

Even shorter signatures can be achieved at the expense of introducing stronger assumptions without relying on Lemma 2, by using the scheme found in [23], which is unforgeable in the generic group model and has signatures of size 2 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$. More details are given in Figure 2.

We now focus on instantiating our ring signatures construction. Combining any scheme from Section 5 with a generic perfectly sound proof system would result in a ring signature scheme that is unlikely to be of interest, as there are already more efficient schemes with/without a trusted setup (see Figure 3 for a comprehensive comparison). However, using the results presented by Chandran, Groth and Sahai [14] we can make the membership proof efficient. They propose a perfectly sound proof of size $\mathcal{O}(\sqrt{n})$ that a public key $\mathsf{pk} \in \mathbb{G}_1$ (or $\mathsf{pk} \in \mathbb{G}_2$), is in a $\mathtt{Ring}$ of size n. This 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. Thus, we must use a compatible SFPK instantiation, leaving as the only scheme without a trusted party assumption Scheme 4. A public key of Scheme 4 contains an element in $\mathbb{G}_T$ and therefore cannot be used with the proof system from Subsection 2.3, which is based on the efficient Groth-Sahai proofs for pairing product equations. We solve this problem in the following way:

Lemma 3. *Scheme 4 is unforgeable and class-hiding even if $X = g_1^x$, $Y = g_2^y$ are publicly known, where $t = e(X^y, g_2) = e(X, Y)$ is part of the signer's public key. Moreover, knowing the secret key one can compute such values.*

Proof. Class-hiding still holds, because the adversary is given the secret keys sk_i for $i \in \{0, 1\}$, which contain X_i and y_i so it can compute X_i and Y_i by itself already. To show that unforgeability still holds, we first have to note that Y is part of the trapdoor τ and does not provide new information for the adversary. Finally, in the proof of unforgeability of Scheme 4 X is set to be g_1^γ, where g_1^γ is part of the decisional linear problem instance. This element is not given to the adversary directly but the same proof works if this value would be given to the adversary.

The idea is that instead of putting the public key $\mathsf{pk}_{\mathsf{FW}} = (A, B, C, D, t, K_{\mathsf{PHF}})$ into the ring, we put $(A, B, C, D, X, Y, K_{\mathsf{PHF}})$. Finally, we modify the first part of the statement proven during signing, i.e. we use

$$\begin{aligned}\exists_{A,B,C,D,X,X',Y,K_{\mathsf{PHF}},r}\ & (i, (A, B, C, D, X, Y, K_{\mathsf{PHF}}), \cdot) \in \mathtt{Ring}\ \wedge\ e(X, g_2^r) = e(X', g_2) \wedge \\ & e(X', Y) = t'\ \wedge\ e(A, g_2^r) = e(A', g_2) \wedge \\ & e(B, g_2^r) = e(B', g_2)\ \wedge\ e(C, g_2^r) = e(C', g_2) \wedge \\ & e(D, g_2^r) = e(D', g_2)\ \wedge\ e(K_{\mathsf{PHF}}, g_2^r) = e(K'_{\mathsf{PHF}}, g_2),\end{aligned}$$

instead of $\exists_{\mathsf{pk},r} \left((i, \mathsf{pk}, \cdot) \in \mathtt{Ring}\ \wedge\ \mathsf{ChgPK}(\mathsf{pk}, r) = \mathsf{pk}'\right)$, where ${\mathsf{pk}_{\mathsf{FW}}}' = (A', B', C', D', t', K'_{\mathsf{PHF}})$ is the randomized SFPK public key used as part of the ring signature. Since all elements in the ring are now elements in $\mathbb{G}_1$ or $\mathbb{G}_2$, we can use the proof system from Subsection 2.3 to efficiently instantiate the proof used in our ring signature construction. What is more, we can also apply the trick from [14] and create a membership proof of length only $\mathcal{O}(\sqrt{n})$. The resulting 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. This solves the open problem stated by Malavolta and Schröder [33].

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

	Scheme	Signature size	Assumptions
Trusted Setup	Shacham-Waters [38]	$\mathcal{O}(n)$	standard
	Boyen [11]	$\mathcal{O}(n)$	q-type
	Chandran-Groth-Sahai [14]	$\mathcal{O}(\sqrt{n})$	q-type
	Malavolta-Schröder [33]	$\mathcal{O}(1)$	q-type + GGM
No Trusted Setup	Chow et al. [18]	$\mathcal{O}(n)$	q-type
	Bender-Katz-Morselli [5]	$\mathcal{O}(n)$	ENC + ZAP
	Malavolta-Schröder [33]	$\mathcal{O}(n)$	q-type + knowledge
	Our scheme	$\mathcal{O}(n)$	standard
	Our scheme with [14]	$\mathcal{O}(\sqrt{n})$	standard

Fig. 3. Comparison of Ring Signature Schemes without Random Oracles and Secure in the Strongest Model from [5]

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] Nuttapong Attrapadung, Benoît Libert, and Thomas Peters. "Efficient Completely Context-Hiding Quotable and Linearly Homomorphic Signatures". In: *PKC 2013*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Springer, 2013.

[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. Springer, 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] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. "Signatures on Randomizable Ciphertexts". In: *PKC 2011*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Springer, 2011.

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

[9] Dan Boneh, David Mandell Freeman, Jonathan Katz, and Brent Waters. “Signing a Linear Subspace: Signature Schemes for Network Coding”. In: *PKC 2009*. Ed. by Stanislaw Jarecki and Gene Tsudik. Springer, 2009.

[10] 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.

[11] Xavier Boyen. “Mesh Signatures”. In: *EUROCRYPT 2007*. Ed. by Moni Naor. Springer, 2007.

[12] 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.

[13] 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.

[14] 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.

[15] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. “Malleable Signatures: New Definitions and Delegatable Anonymous Credentials”. In: *CSF 2014*. IEEE Computer Society, 2014.

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

[17] David Chaum and Eugène van Heyst. “Group Signatures”. In: *EUROCRYPT ’91*. Ed. by Donald W. Davies. Springer, 1991.

[18] Sherman S. M. Chow, Victor K.-W. Wei, Joseph K. Liu, and Tsz Hon Yuen. “Ring signatures without random oracles”. In: *ASIACCS 2006*. Ed. by Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Paul Lin, Shiuhpyng Shieh, and Sushil Jajodia. ACM, 2006.

[19] Nicolas T. Courtois and Rebekah Mercer. “Stealth Address and Key Management Techniques in Blockchain Systems”. In: *ICISSP 2017*. Ed. by Paolo Mori, Steven Furnell, and Olivier Camp. SciTePress, 2017.

[20] 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.

[21] 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*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Springer, 2016.

[22] Georg Fuchsbauer and Romain Gay. *Weakly Secure Equivalence-Class Signatures from Standard Assumptions*. Cryptology ePrint Archive, Report 2018/037. 2018.
[23] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. *EUF-CMA-Secure Structure-Preserving Signatures on Equivalence Classes*. Cryptology ePrint Archive, Report 2014/944. 2014.
[24] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. "Practical Round-Optimal Blind Signatures in the Standard Model". In: *CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Springer, 2015.
[25] 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.
[26] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Non-interactive Zaps and New Techniques for NIZK". In: *CRYPTO 2006*. Ed. by Cynthia Dwork. Springer, 2006.
[27] Jens Groth and Amit Sahai. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *EUROCRYPT 2008*. Ed. by Nigel P. Smart. Springer, 2008.
[28] 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, 2014.
[29] Dennis Hofheinz and Eike Kiltz. "Programmable Hash Functions and Their Applications". In: *CRYPTO 2008*. Ed. by David A. Wagner. Springer, 2008.
[30] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David A. Wagner. "Homomorphic Signature Schemes". In: *CT-RSA 2002*. Ed. by Bart Preneel. Springer, 2002.
[31] Benoît Libert, Thomas Peters, and Moti Yung. "Short Group Signatures via Structure-Preserving Signatures: Standard Model Security from Simple Assumptions". In: *CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Springer, 2015.
[32] Benoît Libert and Damien Vergnaud. "Multi-use unidirectional proxy re-signatures". In: *CCS 2008*. Ed. by Peng Ning, Paul F. Syverson, and Somesh Jha. ACM, 2008.
[33] Giulio Malavolta and Dominique Schröder. "Efficient Ring Signatures in the Standard Model". In: *ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Springer, 2017.
[34] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. http://bitcoin.org/bitcoin.pdf.
[35] Tatsuaki Okamoto and Xiaoyun Wang, eds. *Public Key Cryptography - PKC 2007*. Springer, 2007.
[36] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to Leak a Secret". In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Springer, 2001.

[37] Nicolas van Saberhagen. *CryptoNote v 2.0*. https://cryptonote.org/whitepaper.pdf. Oct. 2013.

[38] Hovav Shacham and Brent Waters. “Efficient Ring Signatures Without Random Oracles”. In: *PKC 2007*. Ed. by Tatsuaki Okamoto and Xiaoyun Wang. Springer, 2007.

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

[40] Eric R. Verheul. “Self-Blindable Credential Certificates from the Weil Pairing”. In: *ASIACRYPT 2001*. Ed. by Colin Boyd. Springer, 2001.

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

Appendix

Full Security Proofs - Group and Ring Signature Constructions

Proof of Theorem 3

Proof (Theorem 3). 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 $\mathsf{gmsk} = ([(\tau^i, \mathsf{pk}^i)]_{i=1}^n)$. 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

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

Proof of Theorem 4

Proof (Theorem 4). 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 $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}^{\ell,\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{SPS\text{-}EQ}}(\lambda)$.

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 **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 **GAME**$_3$) and we have $\Pr[S_4] = \Pr[S_3]$.

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 **GAME**$_4$ and **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 **GAME**$_4$. If it was created by the second user, then we are in **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 SPS-EQ signature on it. However, an adversary $\mathcal{A}$ can still use the way we implemented the $\mathsf{Open}_{\mathsf{GS}}$ in **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

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

Proof of Theorem 5

Proof (Theorem 5). 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

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

Proof of Theorem 6

Proof (Theorem 6). 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 $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$.

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 $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{wi}}(\lambda)$.

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

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

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 **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:

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

Full Security Proofs - SFPK Scheme

Proof of Theorem 9

Proof (Theorem 9).

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 $\rho = (\mathsf{BG}, Y_1, Y_2, K_{\mathsf{PHF}})$ and the public key $\mathsf{pk}_{\mathsf{FW}} = (A, B, X)$ as follows. It sets:

$$X = g_1^\beta \qquad Y_1 = g_1^\alpha \qquad Y_2 = g_2^\alpha$$
$$A = g_1^\gamma \qquad B = g_1^\theta$$

and $(K_{\mathsf{PHF}}, \tau_{\mathsf{PHF}}) \xleftarrow{\$} \mathsf{PHF.TrapGen}(\lambda, g_1^\beta, g_1)$. 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. The reduction $\mathcal{R}$ follows the following steps:

- it chooses random values $t \xleftarrow{\$} \mathbb{Z}_p^*$,
- it computes $(a_m, b_m) \xleftarrow{\$} \mathsf{PHF.TrapEval}(\tau_{\mathsf{PHF}}, m)$ and aborts if $a_m = 0$,
- it computes $\mathsf{pk_{FW}}' \xleftarrow{\$} \mathsf{ChgPK_{SFPK}}(\mathsf{pk_{SFPK}}, l)$,
- it computes:

$$\begin{aligned}
\sigma^1_{\mathsf{FW}} &= (g_1^\beta)^{t \cdot a_m} \cdot ((g_1^\alpha)^{-a_m^{-1} \cdot l} \cdot g_1^t)^{b_m},\\
\sigma^2_{\mathsf{FW}} &= (g_1^\alpha)^{-a_m^{-1} \cdot l} \cdot g_1^t,\\
\sigma^3_{\mathsf{FW}} &= (g_1^\alpha)^{-a_m^{-1} \cdot l} \cdot g_2^t,
\end{aligned}$$

- set the signature $\sigma_{\mathsf{FW}} = (\sigma^1_{\mathsf{FW}}, \sigma^2_{\mathsf{FW}}, \sigma^3_{\mathsf{FW}})$.

We will now show that this is a valid signature. Note that the a valid signature is of the form $(g_1^{\alpha \cdot \beta \cdot l} \cdot ((g_1^\beta)^{a_m} \cdot g_1^{b_m})^r, g_1^r, g_2^r)$. In this case, the reduction has set $r = -a_m^{-1} \cdot \alpha \cdot l + t$ and this means that the $g_1^{\alpha \cdot \beta \cdot l}$ cancels out and the reduction does not need to compute $g_1^{\alpha \cdot \beta}$.

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

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

for which we hope that $a_{m^*} = 0$, where $(a_{m^*}, b_{m^*}) \xleftarrow{\$} \mathsf{PHF.TrapEval}(\tau_{\mathsf{PHF}}, m^*)$. Moreover, since this should be a valid forgery then we have that this signature is under a public key $\hat{\mathsf{pk}_{\mathsf{FW}}}$ for which $(\mathsf{pk_{FW}}, \hat{\mathsf{pk}_{\mathsf{FW}}}) \in \mathcal{R}$. Thus, we have

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

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

Finally, the reduction $\mathcal{R}$ returns $(A^{l^*}, B^{l^*}, \hat{\sigma_{\mathsf{FW}}}^1 \cdot (\hat{\sigma_{\mathsf{FW}}}^2)^{-b_{m^*}})$, 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 (i.e. for which we must have $a_{m^*} = 0$). However, since we assume that the used hash function is a $(1, \mathsf{poly}(\lambda))$-programmable hash function, it follows that $\mathcal{R}$ has a non-negligible advantage in solving the co-Flexible Diffie-Hellman assumption if $\mathcal{A}$'s advantage is non-negligible.

Proof of Theorem 10

Proof (Theorem 10). 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 $\mathsf{pk}_{\mathsf{FW}}' = (A', B', X')$ be the public key given to the adversary. Moreover, let $\mathsf{pk}_{\mathsf{FW}_0} = (A_0, B_0, X_0)$ and $\mathsf{pk}_{\mathsf{FW}_1} = (A_1, B_1, X_1)$ 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 $\mathsf{pk}_{\mathsf{FW}_0}$ and $\mathsf{pk}_{\mathsf{FW}_1}$ 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 $\mathsf{sk}_{\mathsf{FW}}'$ and $\mathsf{pk}_{\mathsf{FW}}'$ but compute them as $\mathsf{pk}_{\mathsf{FW}}' = (Q^{a_{\hat{b}}}, Q^{b_{\hat{b}}}, Q^{x_{\hat{b}}})$, and $\mathsf{sk}_{\mathsf{FW}}' = (Q^{x_{\hat{b}}\cdot y}, \mathsf{pk}_{\mathsf{FW}}')$, 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 valid signatures using $\mathsf{sk}_{\mathsf{FW}}'$.

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

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

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

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

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 knows the secret key $\mathsf{sk_{FW}}'$ and can answer signing queries.

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

$\mathbf{GAME}_3$: 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 $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$.

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 valid signatures are of the form $\sigma_{\mathsf{FW}} = ((Q^{x_{\hat{b}}})^y (\mathcal{H}_{K_{\mathsf{PHF}}}(m))^r, g_1^r, g_2^r)$ 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 $\mathsf{Adv}_{\mathcal{A},\mathsf{SFPK}}^{\mathsf{c\text{-}h}}(\lambda) = \Pr[S_0] \leq 2 \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ddh}}(\lambda)$.

Group Signature Definitions

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

Definition 18 (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 19. *For group signature scheme* GS *and adversary* $\mathcal{A}$ *we define the following experiment:*

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

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

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

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

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

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*

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

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 20. *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:*

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

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 21 (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* $\mathtt{Ring} = \left(\mathsf{pk}_{\mathsf{RS}}^{(1)}, \dots, \mathsf{pk}_{\mathsf{RS}}^{(n)}\right)$ *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* $\mathtt{Ring}$ *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* $\rho \xleftarrow{\$} \mathsf{RCRSGen}(\lambda)$*, any* $\left\{(\mathsf{sk}_{\mathsf{RS}}^{(i)}, \mathsf{pk}_{\mathsf{RS}}^{(i)})\right\}_{i=1}^{n}$ *generated with* $\mathsf{RKeyGen}(\rho, 1^\lambda)$*, any* $s \in \{1, \dots, n\}$ *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* $\mathtt{Ring} = \left(\mathsf{pk}_{\mathsf{RS}}^{(1)}, \dots, \mathsf{pk}_{\mathsf{RS}}^{(n)}\right)$*.*

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

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

$\rho \xleftarrow{\$} \mathsf{RCRSGen}(\lambda); Q := \emptyset, C := \emptyset$
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 $\mathsf{RVerify}(\rho, m^*, \Sigma^*, \mathtt{Ring}^*) = 1 \wedge$
  $(m^*, \mathtt{Ring}^*) \notin Q \wedge$
  $\mathtt{Ring}^* \subseteq S \setminus C$

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

$Q := Q \cup \{(m, \mathtt{Ring})\}$
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 $\perp$

$\underline{\mathsf{Corrupt}(i)}$

$C := C \cup \left\{\mathsf{pk}^{(i)}_{\mathsf{RS}}\right\}$
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 23 (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:*

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

$\rho \xleftarrow{\$} \mathsf{RCRSGen}(\lambda)$
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:*

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