Collusion Resistant Revocable Ring Signatures and Group Signatures from Hard Homogeneous Spaces

Yi-Fu Lai and Samuel Dobson

The University of Auckland, New Zealand
ylai276@aucklanduni.ac.nz, samuel.dobson.nz@gmail.com

Abstract

Both ring signatures and group signatures are useful privacy tools, allowing signers to hide their identities within a set of other public keys, while allowing their signatures to be validated with respect to the entire set. Group signature schemes and *revocable* ring signature schemes both provide the additional ability for certain authorized members to revoke the anonymity on a signature and reveal the true signer—allowing management of abuse in the scheme. This work consists of two parts. Firstly, we introduce a stronger security notion—*collusion resistance*—for revocable ring signatures and show how to derive a group signature scheme from it, which provides a new approach to obtaining group signatures. This improves on the existing weak security model (e.g. with *selfless* anonymity) which fails to guarantee anonymity of members whose keys are exposed. Our stronger notion requires that the scheme remains secure against full key exposure in the anonymity game, and allows collusion among arbitrary members in the revocability game. Secondly (and more concretely), we construct a practical collusion-resistant revocable ring signature scheme based on hard homogenous spaces (HHS), and thus obtain a group signature scheme based on isogenies. To the best of our knowledge, the schemes given in this work are the first efficient post-quantum (collusion-resistant) revocable ring signature scheme, and the first efficient isogeny-based group signature scheme in the literature.

1 Introduction

Group signature (GS) schemes were first introduced by the seminal work of Chaum and van Heyst [CvH91]. The scheme allows authorized participants to sign on behalf of a "group" of users or keys, while the specific identity of the signer remains anonymous. A highly related notion is that of ring signatures, introduced by Rivest, Shamir, and Tauman [RST01]. Ring signatures also allow a signer to hide their identity within a set of users or keys—a set which is chosen ad-hoc at the time of signing. An important differentiating factor, though, is that a group signature gives a *group manager* the ability to revoke this anonymity and reveal who created a specific signature. Thus, the scheme gives many of the anonymity benefits of ring signatures while providing a mechanism to deal with abuse.

Between the notions of ring and group signatures, there is room for a hybrid scheme. Known in the literature as a revocable ring signature [LLM+07, ZLS+20], this construction takes both the ad-hoc sign-time formation of a ring from the ring signature setting, but also combines it with the power to grant authority to revoke anonymity, as in a group signature schemes.

As we know, due to Shor's algorithm [Sho99], all "classical" cryptographic assumptions such as the hardness of factoring and the discrete logarithm problem fall prey to the possibility of quantum adversaries—hence the surged interest in post-quantum cryptography research over the last ten years. One of the major contenders for post-quantum standardization is isogeny-based cryptography, first presented by Couveignes, Rostovtsev and Stolbunov in 2006 [Cou06, RS06]. The Supersingular Isogeny Key Encapsulation (SIKE) proposal [JAC+17], based on the Supersingular Isogeny Diffie Hellman (SIDH) scheme by De Feo, Jao, and Plût [JD11, DFJP14], is now one of the third-round alternate candidates in the post-quantum cryptography standardization competition led by NIST [NIS20]. Isogeny-based cryptography is important due to it making use of the thorough study of elliptic curves from classical elliptic curve cryptography, and also benefits from relatively short key sizes (although can suffer from slow computation times). Another distinctive and efficient isogeny-based key exchange scheme was devised by Castryck et al. called Commutative SIDH (CSIDH) [CLM+18]. The CSIDH

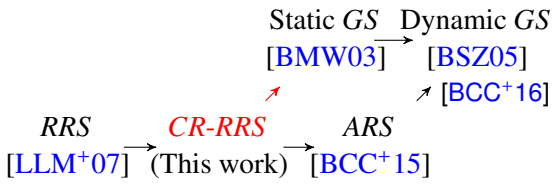

Figure 1: The figure shows the relationships between several cryptographic constructions. $A \leftarrow B$ means a construction for A can be derived from a construction for B. The notion *CR-RRS* we introduce provides a new approach to constructing static group signature schemes.

scheme is conjectured to provide post-quantum security with smaller public keys than the candidates in the NIST competition [NIS20] (although this claim of security is under debate [CLM+18, Pei20]).

Recently, a variety of isogeny-based cryptographic primitives have been proposed, including signature schemes [Sto09, BKV19, DKL+20], a ring signature scheme [BKP20], a threshold scheme [DM20], a UC-secure oblivious transfer scheme [LGdSG20], and miscellaneous theoretical constructions [ADMP20]. Yet, despite these constructions, and though many lattice-based group signature schemes have been proposed [CLRS10, LLLS13, LLNW14, dPLS18], there has been no efficient group signature scheme presented in the literature based upon supersingular isogeny assumptions[1]. There is also currently no post-quantum construction of a revocable ring signature scheme in the literature.

The only isogeny-based "group signature" is briefly mentioned in [BKP20, Remark 5.5] by using the linkable ring signature scheme proposed in that work. The standard security model for static group signature schemes [BMW03] requires two key properties—anonymity and full-traceability. Unfortunately, the idea for a group signature scheme in [BKP20] fails to satisfy both requirements. Specifically, in their scheme, the manager's secret key can be used to forge a signature which opens to an honest party—a breach of the full-traceability property. Their linkable anonymity is *selfless* (in Section 2.3) and, in fact, cannot be secure against full-key exposure.

In this work, we resolve the open problem of constructing an efficient, isogeny-based group signature scheme. We also provide the first construction of an efficient isogeny-based revocable-ring signature scheme—in fact, the first such scheme providing post-quantum security. Furthermore, our revocable ring signature scheme gives a stronger anonymity guarantee (CPA-anonymity) than the existing literature, which ensures only *selfless* anonymity. The efficiency of our scheme is discussed further in Section 3.3.

1.1 Contributions

This work aims to fill the gap in the literature with a new connection between a ring signature and a group signature as shown in Fig 1. We introduce firstly the idea of a **collusion-resistant** (CR-RRS) revocable ring signature, which strengthens the existing definition of revocable ring signatures (RRS) with stronger privacy guarantees in two ways. In particular, this new security notion takes non-selfless anonymity into account and allows arbitrary collusion in the revocability game. This notion establishes a connection from ring signatures to group signatures and provides a straightforward approach to constructing group signatures with provable security. Furthermore, the idea can be practically instantiated from hard homogeneous spaces (HHS) such as CSIDH or CSIDH-512 [BKV19], and therefore provides a new group signature scheme from HHS. To the best of our knowledge, this scheme (when instantiated with the CSIDH group action) is the first efficient, secure isogeny-based group signature scheme and the first efficient post-quantum revocable ring signature scheme[2].

1.2 Technical Review

We briefly summarize the key technical details of this work for the reader familiar with many of the concepts involved. These details will, of course, be explained in more detail in the rest of the paper.

[1] We remark that [BMW03] had shown a theoretical construction for group signature schemes by using a PKE, a signature scheme, and an NIZK for NP statements, but the efficiency of this generic construction is not guaranteed.

[2] Though ARS implies CR-RRS as shown in Figure 1, the existing ARS constructions are only based on pre-quantum assumptions to the best of our knowledge.

To construct our revocable ring signature from an HHS, we start with a sigma protocol for the OR-relation as in [BKP20]. Specifically, the ring public key is of the form $(E, E', E_1, E_2, ..., E_n)$, where the $E_j = s_j \star E$ are the individual public keys in the ring (corresponding to secret keys s_j), and $E' = t \star E$ is the public key of the opening authority. The binary challenge sigma protocol works as follows: The prover P_k commits to $S = (r_{\psi(i)} * E_{\psi(i)})_{i\in[n]}$ where ψ is a permutation in $\mathcal{S}_n$ used to randomize the order of the curves and r'_i are random ephemeral secrets. The prover also commits to the encrypted permutation, using probabilistic CPA-secure encryption $Enc_{E'}(\psi; rnd)$ under the revocation authority's public key E' using random coins rnd.

When the challenge is 1, the prover reveals the ephemeral values $(r_i)_{i\in[n]}$, the permutation ψ and the randomness rnd. The verifier can then check the equality of the curves and the permutation by $S = (r_{\psi(i)} * E_{\psi(i)})_{i\in[n]}$, and deterministically recompute the ciphertext $Enc_{E'}(\psi; rnd)$ using the random coins. When the challenge is 0, the prover P_k only responds with $s_k * r_k$. Then, the verifier simply checks whether $(s_k * r_k) * E$ is in S. The sigma protocol has *2-special soundness* and *computational honest-verifier-zero-knowledge* (see Section 3). The opening authority can extract the index of the signer, k (but not the secret key s_k) when the challenge is 0 due to the knowledge of the decryption secret for $Enc_{E'}(\psi; rnd)$. In particular, the opening authority has $s_k * r_k$ such that $(s_k * r_k) * E \in S$ and the index can be recovered by inverting ψ. Importantly though, under exposure of the secret key s_k, the adversary is still not able to identify the signer because the permutation ψ is unknown, which illustrates some of the intuition of the scheme's anonymity. These proofs are given in detail in Section 3.

To prove security, we use a simple CPA-secure ElGamal-type public key encryption for $Enc_{E'}(\psi; rnd)$, defined to be $(rnd * E \,\|\, H(rnd * E') \oplus \psi)$, where rnd is interpreted as an element in the HHS group (for example, a CSIDH secret key).

Roadmap. The rest of the paper is organized as follows. Section 2 contains some brief preliminaries on the definition of hard homogeneous spaces, (restricted) effective group actions, CSIDH, and introduces collusion-resistant revocable ring signatures. A summarized survey of ring and group signature literature can be found in Appendix A. Section 3 gives an instantiation of a collusion-resistant revocable ring signature scheme based on HHS ((restricted) effective group actions) with an efficiency analysis of the scheme and some possible optimizations of the scheme. A method of obtaining a secure GS scheme from a CR-RRS is provided in Appendix C. Finally, we conclude with a brief summary and identify potential future work in Section 4.

Concurrent Works. Surprisingly, we notice that there are two concurrent works relevant to isogeny-base group signatures published to the Cryptology ePrint Archive [BDK+21, CHH+21] literally around the same time. All of three works attempt to obtain a group signature scheme through a ring signature though the security models for the underlying ring signature schemes differ significantly.

One is an isogeny-based scheme given by Chung et al. [CHH+21]. We remark that though they claim to have an accountable ring signature, they do not follow the notions specified by [BCC+16]. In short, their main focus is similar to this work. Their signature size grows in $O(N^2)$, excluding public keys, which is larger than ours, where N represents the size of the group.

The other one is a framework given by Beullens et al. [BDK+21] provided with an isogeny-based instance and a lattice-based instance. The underlying ring signature scheme follows the security definitions specified by [BCC+16] so that the framework is applicable to a secure dynamic group signature scheme formalized in [BSZ05]. Their signature size grows in $O(\log(N))$. Notably, the additional cost of signature size of their tightly secure variant is constant in N.

Table 1 gives a comparison among three works. Given the diversified security definitions over ring signatures, the security notions in the table are taken from [BMW03] for a static group signature (or see Appendix C.1).

Notions	Signature Size	Anonymity	Unforgeability	Full-traceability
This work	$O(N \log(N))$	CPA	Yes	Yes
[CHH+21]	$O(N^2)$	CPA	Yes	Yes
[BDK+21]	$O(\log(N))$	CCA	Yes	Yes

Table 1: Comparison with concurrent works. The security notions in the first row are taken from [BMW03] for a static group signature or see Appendix C.1. The interger N represents the size of the group.

2 Preliminaries

Notation. We denote the set $\{1, 2, ..., n\}$ by $[n]$. Let $\mathcal{S}_n$ represent the symmetric group of degree n, where the elements can be represented as a string. Let λ represent the security parameter. Finally, we use $\|$ to denote concatenation.

2.1 Hard Homogeneous Spaces

The idea of hard homogeneous spaces was proposed by Couveignes [Cou06]. The concept generalizes the discrete-logarithm (DLP) and Diffie-Hellman (DH) problems, which are both prolific in cryptographic constructions. Given a finite commutative group $\mathcal{G}$ and a set $\mathcal{X}$ where $\mathcal{G}$ acts freely and transitively on $\mathcal{X}$, we say $\mathcal{G}$ and $\mathcal{X}$ constitute a hard homogeneous space if the following tasks are computationally *feasible* in polynomial-time:

- **Group Operations:** Given $g_1, g_2 \in \mathcal{G}$, to compute $g_1^{-1} g_2$.
- **Uniform Sampling:** To sample an element from $\mathcal{G}$ uniformly (or close to uniformly) at random.
- **Set Membership:** To decide the validity and equality of elements of $\mathcal{X}$.
- **Group Action:** Given $g \in \mathcal{G}$ and $x \in \mathcal{X}$, to compute $g * x$ (the action of g on x).

While the following two problems are required to be computationally *infeasible*:

- **Vectorization:** Given x and $x' \in \mathcal{X}$, to find $g \in \mathcal{G}$ such that $g * x = x'$.
- **Parallelization:** Given x, x', and $y \in \mathcal{X}$, where $x' = g * x$ for some $g \in \mathcal{G}$, to find $y' \in \mathcal{X}$ such that $y' = g * y$.

2.2 CSIDH

Let p be an odd prime. It is well known that the number of $\mathbb{F}_p$ rational points on a supersingular elliptic curve defined over $\mathbb{F}_p$ is $p + 1$. Also, $E(\mathbb{F}_p)$ has a subgroup of order ℓ if $\ell \mid p + 1$. We denote $\mathrm{End}_p(E)$ to be the ring of endomorphisms of E defined over $\mathbb{F}_p$. Again, for a supersingular E, it is known that $\mathrm{End}_p(E)$ is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$.

Let $\mathcal{O}$ be an order in an imaginary quadratic field. Denote by $\mathcal{E}$ the set of all isomorphism classes of elliptic curves E over F_p with $\mathrm{End}_p(E)$ isomorphic to $\mathcal{O}$. The ideal class group $\mathrm{Cl}(\mathcal{O})$ is the quotient of the group of invertible fractional ideals of $\mathcal{O}$ modulo the principal fractional ideals of $\mathcal{O}$. It is known that $\mathrm{Cl}(\mathcal{O})$ acts freely and transitively on the set $\mathcal{E}$ by $([\mathfrak{a}], E) \mapsto E/\mathfrak{a}$.

However, the task of computing an action $[\mathfrak{a}] * E$ for $[\mathfrak{a}] \in \mathrm{Cl}(\mathcal{O}), E \in \mathcal{E}$ is subexponential time in general [BFJ16]. So instead, for ℓ dividing $p + 1$, the ideals $\mathfrak{l} = \langle \ell, \pi \rangle$ are used, where π is the $\mathbb{F}_p$-Frobenius endomorphism. The actions of these ideals (with repeated application) are feasible to compute via Vélu-type formulae [Vél71], and can be used to generate a large (if not the entire) keyspace. This action of $\mathrm{Cl}(\mathcal{O})$ on $\mathcal{E}$ (for a chosen p) is referred to as the CSIDH [CLM+18] action, and is conjectured to be a hard homogenous space (HHS).

More concretely, in CSIDH, the prime p is defined as $p = 4 \cdot \ell_1 \cdots \ell_n - 1$ where each of the ℓ_i are distinct, small, odd primes. The n-tuple string $(e_i)_{i \in [n]}$ is used to represent the ideal $\mathfrak{g} = \prod_i \mathfrak{l}_i^{e_i}$, where the $e_i \in [-m, m]$ are small-ish (in CSIDH-512, $m = 5$). It is a heuristic assumption made in [CLM+18] that the ideals generated in this way are "uniform-enough" over all class group elements in $\mathrm{Cl}(\mathcal{O})$. To date, this assumption has not been invalidated and no flaws in the cryptosystem have resulted from it. Beullens et al. [BKV19] presented CSI-FiShas an efficient signature scheme, and in doing so compute the class group and relation lattice for the CSIDH-512 parameter set. This allows uniform sampling from the class group Cl. For convenience, we will henceforth simplify the class group notation $[a] \in Cl$ into $a \in Cl$.

In this setting these isomorphism classes can be represented uniquely by a coefficient $A \in \mathbb{F}_p$ representing the curve $E_A : y^2 = x^3 + Ax^2 + x$. The **base curve** is chosen to be $E_0 : y^2 = x^3 + x$, which is supersingular. Moreover, the endomorphism $\mathrm{End}_p(E) \cong \mathbb{Z}[\sqrt{-p}]$ and the class number of $\mathbb{Q}(\sqrt{-p})$ is heuristically assumed to be approximately $\sqrt{p}$ in [CLM+18]. Theoretically, Kuperberg's paper gives an algorithm to solve a dihedral

hidden-subgroup problem with query complexity $O(2^{3\sqrt{\log(N)}})$ [Kup05] where N is the order of the underlying dihedral group. Hence, practically, we may also assume $\sqrt{p} > 2^{\lambda}$ when $\lambda \geq 9$.

Modeling. Alamati et al. [ADMP20] defined more detailed frameworks for cryptographic group actions, including an *effective* group action (EGA) and a *restricted* effective group action (REGA). Briefly, an EGA models the standard cryptographic group action, and allows efficient membership testing, sampling, and equality testing within the group; efficient computation of the group operation, inversions, and the group action; and unique representation of elements in the set being acted upon. Alamati define computational hardness assumptions for EGA in a hierarchy of three levels: one-wayness (OW), weak unpredictability (wU), and weak pseudo-randomness (wPR)—corresponding to the vectorization problem, generalized parallelization problem, and the decisional variant, respectively. The REGA framework is a weakening of EGA, and only requires efficient computation of the action of a small (polylogarithmic in $|G|$) generating set of the group. It also no longer requires efficient equality testing of group elements in G among other things. We refer to [ADMP20] for more thorough descriptions of these models.

The group action used in CSIDH is an example of REGA (instantiated with isogenies), as only the actions of the $\mathfrak{l}_i$ (repeated a few times) are efficiently computable. This can be turned into an EGA or HHS (HHS, in this terminology, is the same as wU-EGA) via the improvement given by CSI-FiSh [BKV19] by exploiting the known structure class group and lattice reductions.

Because REGA does not require unique group element representation or equality testing, protocols based on REGA tend to be less efficient to ensure information about the secret is not leaked. For instance, in SeaSign [DG19], the larger interval $[-2m, 2m]$ is used rather than $[-m, m]$. Specifically, if $a + b = 2m$, where $a, b \in [-m, m]$, then we learn that $a = b = m$. This issue is addressed by using rejection sampling techniques to force the output distribution to be independent of the secret and prevent leaking secret information while the approach also makes the scheme extremely slow [DG19].

In short, the construction in Section 3 can be constructed in the framework of either wU-REGA by using SeaSign(resulting in an extremely inefficient scheme), or with HHS (or wU-EGA) by using CSI-FiSh [BKV19] as a more efficient and practical option.

2.3 Revocable Ring Signature Schemes

In this work we introduce a new definition of security for what we call **collusion-resistant** revocable ring signature schemes. Such a scheme has a strengthened security definition over that previously used in the revocable ring signature literature. Recall that a revocable ring signature is a ring signature with the added functionality of granting *revocation authority* to a given public key (or keys) at sign-time, which allows the owner of said key to later reveal the true signer of any signature under their authority. The important difference between this idea and the idea of a group signature is that revocable ring signatures are still ad-hoc and created at sign-time—the revocation authority has no say in defining who can and cannot be part of a ring. Any given RRS can be considered a ring signature too, if the signer generates or provides an arbitrary revocation public key to the signing function and ignores the rest of the revocation functionality. If public keys can be sampled for which the secret is not known, this gives a standard ring signature construction.

Formally, a revocable ring signature scheme consists of six polynomial-time algorithms $\mathcal{RRS}$ = (**RRS.Setup**, **RRS.KeyGen**, **RRS.RevKeyGen**, **RRS.Sign**, **RRS.Verify**, **RRS.Revoke**):

- $\mathsf{pp} \leftarrow$ **RRS.Setup**(1^{λ}) is a PPT algorithm returning a set of public parameters for the scheme (which may be used implicitly in the other functions).
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow$ **RRS.KeyGen**(pp) is a PPT algorithm returning a private key and public key pair for a user of the scheme.
- $(\mathsf{sk}_{\mathsf{rev}}, \mathsf{pk}_{\mathsf{rev}}) \leftarrow$ **RRS.RevKeyGen**(pp) is a PPT algorithm returning a private key and public key pair for a revocation authority.
- $\sigma \leftarrow$ **RRS.Sign**$(\mathsf{sk}_i, m, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{pk}_{\mathsf{rev}})$ is a randomized signing algorithm taking as inputs the secret key sk_i, the message m, a set of public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ (including pk_i), and a public key for the revocation authority. It returns σ as the signature.

- $0 \mid 1 \leftarrow \textbf{RRS.Verify}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, m, \mathsf{pk}_{\mathsf{rev}}, \sigma)$ is a deterministic verification algorithm taking as inputs the set of public keys pk_j of the ring, the message m, the revocation public key $\mathsf{pk}_{\mathsf{rev}}$, and the signature σ. It returns 1 to represent validity of the signature, or 0 otherwise.
- $i \leftarrow \textbf{RRS.Revoke}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{sk}_{\mathsf{rev}}, m, \sigma)$ is a PPT algorithm taking as inputs the public keys pk_j of the ring, the revocation authority's secret key $\mathsf{sk}_{\mathsf{rev}}$, the message m, and the signature σ. It returns an index i to represent the identity of the member in the ring who generated the signature. If the signature is invalid, then $\perp$ is returned.

Important correctness and security properties for ring signatures are as follows:

Correctness. We require that the following two correctness properties hold:

$$\textbf{RRS.Verify}(\mathrm{PK}, m, \mathsf{pk}_{\mathsf{rev}}, \textbf{RRS.Sign}(\mathsf{sk}_i, m, \mathrm{PK}, \mathsf{pk}_{\mathsf{rev}})) = 1$$

with probability 1, where PK is a set of public keys including pk_i.

$$\textbf{RRS.Revoke}(\mathrm{PK}, \mathsf{sk}_{\mathsf{rev}}, m, \sigma) = \perp \iff \textbf{RRS.Verify}(\mathrm{PK}, m, \mathsf{pk}_{\mathsf{rev}}, \sigma)) = 0$$

That is, revocation fails to output an index if and only if the signature is invalid.

The security of a revocable ring signature scheme is based on three properties—unforgeability, anonymity, and full-revocability. For these properties, we shall define four oracles to capture the powers of the adversary:

- $O_{gen}()$. The generation oracle runs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \textbf{RRS.KeyGen}(pp)$, sets $Q_{gen} \leftarrow Q_{gen} \cup \{\mathsf{pk}_i\}$ and $\mathrm{SK}[\mathsf{pk}_i] = \mathsf{sk}_i$, and returns pk_i.
- $O_{cor}(\mathsf{pk}_i)$. The corruption oracle takes a public key $\mathsf{pk}_i \in Q_{gen}$ as the input, sets $Q_{cor} \leftarrow Q_{cor} \cup \{\mathsf{pk}_i\}$, and returns the corresponding secret key $\mathsf{sk}_i = \mathrm{SK}[\mathsf{pk}_i]$.
- $O_{sig}(\mathrm{PK}, m, \mathsf{pk}_i, \mathsf{pk}_{\mathsf{rev}})$. The signing oracle takes a set of public keys $\mathrm{PK} \subseteq Q_{gen}$, a message m, a public key $\mathsf{pk}_i \in \mathrm{PK}$, and a revocation public key $\mathsf{pk}_{\mathsf{rev}} \in Q_{gen}$. It calls $\sigma \leftarrow \textbf{RRS.Sign}(\mathrm{SK}[\mathsf{pk}_i], m, \mathrm{PK}, \mathsf{pk}_{\mathsf{rev}})$, sets $Q_{sig} \leftarrow Q_{sig} \cup \{(m, \sigma)\}$, and returns the signature σ.
- $O_{rev}(\mathrm{PK}, m, \mathsf{pk}_{\mathsf{rev}}, \sigma)$. The revoking oracle takes a set of public keys $\mathrm{PK} \subseteq Q_{gen}$, a message m, a revocation public key $\mathsf{pk}_{\mathsf{rev}} \in Q_{gen}$, and a signature σ. It calls $i \leftarrow \textbf{RRS.Revoke}(\mathrm{PK}, \mathrm{SK}[\mathsf{pk}_{\mathsf{rev}}], m, \sigma)$, sets $Q_{\mathsf{rev}} \leftarrow Q_{\mathsf{rev}} \cup \{\sigma\}$, and returns the index i.

Unforgeability. This property ensures that an adversary cannot generate an accepting signature for a ring of public keys they are not part of (and do not know any secret keys for). For security parameter λ, the challenger C generates public parameters pp, and, for all $i \in [n]$, keypairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \textbf{RRS.KeyGen}(pp)$, and sets $\mathrm{SK}[\mathsf{pk}_i] = \mathsf{sk}_i$. The challenger also sets $Q_{gen} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, $Q_{cor} = \{\}$ and $Q_{sig} = \{\}$. A revocable ring signature scheme is unforgeable if, for any PPT adversary $\mathcal{A}$, given $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ from C, as well as access to oracles $O_{gen}, O_{cor}, O_{sig}$, there exists a negligible function $\mathsf{negl}()$ such that

$$\mathsf{Adv}_{\mathsf{unf}}(\mathcal{A}) := \Pr\left[\begin{array}{c}(\sigma^*, \mathrm{PK}^*, m^*, \mathsf{pk}^*_{\mathsf{rev}}) \leftarrow \mathcal{A}^{O_{gen}, O_{cor}, O_{sig}}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}) \\ \textbf{RRS.Verify}(\mathrm{PK}^*, m^*, \mathsf{pk}^*_{\mathsf{rev}}, \sigma^*) = 1 \\ \wedge\ \mathrm{PK}^* \subseteq Q_{gen} \\ \wedge\ \mathrm{PK}^* \cap Q_{cor} = \emptyset \\ \wedge\ (m^*, \sigma^*) \notin Q_{sig}\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Note that unforgeability places no restriction on the revocation key, which the adversary generates themselves.

Full-anonymity. This property ensures anonymity for members even in the case of secret key exposure, and even if the identity of the signer of several other signatures has been revealed. The challenger C first generates public parameters pp for security parameter λ, and for all $i \in [n]$, keypairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \textbf{RRS.KeyGen}(pp)$. The challenger also generates a revocation keypair $(\mathsf{sk}_{\mathsf{rev}}, \mathsf{pk}_{\mathsf{rev}}) \leftarrow \textbf{RRS.RevKeyGen}(pp)$. A revocable ring

signature scheme has full-anonymity if, for any PPT adversaries $\mathcal{A}_1, \mathcal{A}_2$ given $\{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_n, \mathsf{pk}_n), \mathsf{pk}_{\mathsf{rev}}\}$ from C, and access to oracle O_{rev}, there exists a negligible function $\mathsf{negl}()$ such that

$$\left| \Pr\left[\begin{array}{c} (m, i, j, \text{state}) \leftarrow \mathcal{A}_1^{O_{rev}}(\{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_n, \mathsf{pk}_n)\}, \mathsf{pk}_{\mathsf{rev}}) \\ b \leftarrow\!\$\ \{i, j\} \\ \sigma \leftarrow \textbf{RRS.Sign}(\mathsf{sk}_b, m, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{pk}_{\mathsf{rev}}) \\ b^* \leftarrow \mathcal{A}_2^{O_{rev}}(\text{state}, \sigma) \\ b^* = b \ \wedge \ \sigma \notin Q_{\mathsf{rev}} \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

In the above definition, if the adversary's access to O_{rev} is removed, then the notion is called *weak anonymity* or sometimes CPA-anonymity [dPLS18].

Remark 1. In the previous literature on revocable ring signatures [LLM+07, ZLS+20], anonymity was defined as guessing the signer without knowing the two secret keys corresponding to $\mathsf{pk}_i, \mathsf{pk}_j$ in the challenge. Our definition is thus stronger and more in-line with that of full-anonymity or CPA-anonymity in the group signature setting, which requires the scheme to be secure against full key exposure. That is, even though the adversary obtains all secret keys except for the revocation secret key, the adversary still cannot tell who the signer is.

Full-revocability. Roughly speaking, full-revocability ensures that the power of the revocation authority to reveal the creator of a particular signature cannot be obstructed even if the opening key is stolen and a number of ring members collude. Formally, in the revocability experiment, the challenger generates public parameters pp for security parameter λ, and runs the key generation algorithm $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \textbf{RRS.KeyGen}(pp)$ for all $i \in [n]$, and $(\mathsf{sk}_{\mathsf{rev}}, \mathsf{pk}_{\mathsf{rev}}) \leftarrow \textbf{RRS.RevKeyGen}(pp)$. A revocable ring signature scheme has full-revocability if, for any PPT adversary $\mathcal{A}$ given $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$ and $(\mathsf{sk}_{\mathsf{rev}}, \mathsf{pk}_{\mathsf{rev}})$ from C, and access to oracles $O_{gen}, O_{sig}, O_{cor}$, there exists a negligible function $\mathsf{negl}()$ such that

$$\mathsf{Adv}_{\mathsf{rev}}(\mathcal{A}) := \Pr\left[\begin{array}{c} (m^*, \sigma^*, \mathrm{PK}^*) \leftarrow \mathcal{A}^{O_{gen}, O_{sig}, O_{cor}}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{sk}_{\mathsf{rev}}, \mathsf{pk}_{\mathsf{rev}}) \\ i \leftarrow \textbf{RRS.Revoke}(\mathrm{PK}^*, \mathsf{sk}_{\mathsf{rev}}, m^*, \sigma^*) \\ \textbf{RRS.Verify}(\mathrm{PK}^*, m^*, \mathsf{pk}_{\mathsf{rev}}, \sigma^*) = 1 \\ \wedge\, i \neq \perp \\ \wedge\, \mathsf{pk}_i \in \mathrm{PK}^* \cap Q_{gen} - Q_{cor} \ \wedge \ (m^*, \sigma^*) \notin Q_{sig} \end{array} \right] \leq \mathsf{negl}(\lambda)$$

Essentially, the goal of the adversary is to generate a valid signature (m, σ) such that revoking the anonymity of the signature does not indicate any identity in the group of corrupted parties (it must either point to an uncorrupted member of the group, or fail to open to any member in the group).

Remark 2. The definition of revocability in previous works [LLM+07, ZLS+20] allows only one honest member's secret key to be corrupted, and does not allow access to the revocation authority's secret key. That is much weaker than the definition of full-revocability we provide.

In order to differentiate schemes satisfying these stronger security notions above, we shall call such schemes **collusion-resistant** revocable ring signature schemes.

Definition 2.1. *A revocable ring signature scheme* $\mathcal{RRS} = (\textbf{\textit{RRS.Setup}}, \textbf{\textit{RRS.KeyGen}}, \textbf{\textit{RRS.RevKeyGen}}, \textbf{\textit{RRS.Sign}}, \textbf{\textit{RRS.Verify}}, \textbf{\textit{RRS.Revoke}})$ *is said to be* collusion-resistant *if the scheme* $\mathcal{RRS}$ *has correctness, unforgeability, CPA-anonymity, and full-revocabililty.*

3 Instantiation from Hard Homogeneous Spaces

This section will present a practical construction of a collusion-resistant revocable ring signature scheme from hard homogeneous spaces (weak-unpredictable-EGA of [ADMP20]). We will start with a sigma protocol, obtain a signature scheme via the Fiat-Shamir transformation, and show the scheme meets the definitions of security for a collusion-resistant revocable ring signature scheme.

3.1 Sigma Protocol

The foundation of our construction is a special sigma protocol for an OR-relation. This sigma protocol is similar to the one presented in [BKP20]. Let the language $R \subset \mathcal{E}^{n+2} \times Cl$ where $R := \{(E, E', E_1, \ldots, E_n, s) \mid s * E = E_j \text{ for some } j \in [n]\}$. Assume $(E, E', E_1, E_2, \ldots, E_n)$ to be the ring public key, each party member P_i has the corresponding secret key s_i such that $s_i * E = E_i$, and $\mathsf{pk}_{\mathsf{rev}} = E' = \mathsf{sk}_{\mathsf{rev}} * E$ is the revocation authority's public key corresponding to secret key $\mathsf{sk}_{\mathsf{rev}}$. Given $(E, E', E_1, \ldots, E_n)$, a sigma protocol to prove the possession of s_k satisfying the relation $(E, E', E_1, \ldots, E_n, s_k) \in R$ is specified as follows:

Protocol 1.

- **Common Input:** Both the prover $\mathcal{P}_k$ and the verifier $\mathcal{V}$ are given group public key $\mathsf{gpk} = (E, E', E_1, \ldots, E_n)$.
- **Private Input:** $\mathcal{P}_k$ has $s_k \in Cl$ such that (gpk, s_k) is in the language R.
- **Specification:**

  1. **(Commitment)** The prover $\mathcal{P}_k$ generates a value $e \leftarrow Cl$, values $r_i \leftarrow Cl$ for $i \in [n]$, and a permutation $\psi \in \mathcal{S}_n$ (represented as a string). The prover then computes $E'_i = r_i * E_i$ for each $i \in [n]$, and then permutes these curves with ψ to obtain a curve array $\mathbf{S} = (S_i)_{i\in[n]}$ where $S_i = E'_{\psi(i)}$. The prover also generates the revocation data with respect to E' by computing $ct = (e*E \parallel H(e*E') \oplus \psi)$. Finally, the prover sends the commitment $comm = (\mathbf{S}, ct)$ to the verifier V.
  2. **(Challenge)** $\mathcal{V}$ randomly generates challenge bit $c \leftarrow \{0, 1\}$ and sends to the prover $\mathcal{P}_k$.
  3. **(Response)** $\mathcal{P}_k$ computes
  $$resp = \begin{cases} s_k r_k & \text{if } c = 0 \\ (r_i)_{i\in[n]} \parallel e \parallel \psi & \text{if } c = 1 \end{cases}$$
  and sends $resp$ to V.
  4. **(Verification)** After parsing $resp$, the verifier $\mathcal{V}$ checks whether
  $$\begin{cases} resp * E \stackrel{?}{\in} \mathbf{S} & \text{if } c = 0, \\ r_{\psi(i)} * E_{\psi(i)} \stackrel{?}{=} S_i \text{ for } i \in [n] \wedge (e * E \parallel H(e * E') \oplus \psi) \stackrel{?}{=} ct & \text{if } c = 1. \end{cases}$$
  If the equations hold, then $\mathcal{V}$ outputs 1. Otherwise, it outputs 0.

The following theorem shows non-abort HVZK for Protocol 1, which will be used in the security proofs. Since the special soundness cannot be directly applied to the security proofs for the scheme, we omit it here to save the space.

Theorem 3.1. *Protocol 1 is correct. Besides, when H is modeled as a random oracle, if the parallelization of HHS is hard, then Protocol 1 is non-abort honest verifier zero-knowledge (HVZK). That is, there exists a simulator given the challenge $c \leftarrow \{0, 1\}$ that generates a transcript $(comm', c, resp')$ which is perfectly indistinguishable from the one $(comm, c, resp)$ generated from the real execution of Protocol 1.*

Proof. See Appendix B □

3.2 Collusion-Resistant Revocable Ring Signatures from Protocol 1

We now illustrate the construction of a revocable ring signature scheme built on top of the aforementioned sigma protocol. We will use the notation of Section 2 regarding hard homogeneous spaces. Given the security parameter λ, n and two random oracles H_1 and H_2, our collusion-resistant revocable ring signature scheme is specified as follows:

Protocol 2. (CR-RRS from Protocol 1)

- $pp \leftarrow$ **RRS.Setup**(1^λ) where $pp = \{\lambda, E, \mathcal{E}, Cl\}$, $E \in \mathcal{E}$ and $(\mathcal{E}, Cl)$ is a hard homogeneous space.
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow$ **RRS.KeyGen**(pp) where the secret key $\mathsf{sk}_i \leftarrow Cl$ and the public key is $\mathsf{pk}_i = \mathsf{sk}_i * E$. The revocation authority's secret key and public key are also generated through this procedure (**RRS.RevKeyGen**() := **RRS.KeyGen**()).

- $\sigma \leftarrow$ **RRS.Sign**$(\mathsf{sk}_k, m, \{\mathsf{pk}_1 = E_1, \ldots, \mathsf{pk}_n = E_n\}, \mathsf{pk}_{\mathsf{rev}} = E_{\mathsf{rev}})$: Generate $e, r_i \leftarrow Cl$ for $i \in [n]$ and $\psi \in \mathcal{S}_n$, compute $comm = ((E'_{\psi(i)})_i, e * E \parallel H_2(e * E_{\mathsf{rev}}) \oplus \psi)$ as the commitment (Item 1 of Protocol 1). Repeat the process $n\lambda$ times to get $\mathbf{comm} = (comm_j)_{j\in[n\lambda]}$. Compute $\mathbf{c} = (c_j)_{j\in[n\lambda]} = H_1(\mathbf{comm} \parallel m) \in \{0,1\}^{n\lambda}$ as the challenge. Depending on whether c_j is 0 or 1, set $resp_j$ to be either $s_k r_k$ or $((r_i)_{i\in[n]} \parallel e \parallel \psi)$ respectively as the j^{th} response (Item 3 of Protocol 1). Let $\mathbf{resp} = (resp_j)_{j\in[n\lambda]}$. Return $\sigma = (\mathbf{comm}, \mathbf{c}, \mathbf{resp})$ as a group signature on m.
- $0 \mid 1 \leftarrow$ **RRS.Verify**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, m, \mathsf{pk}_{\mathsf{rev}}, \sigma)$: For each $j \in [n\lambda]$, only if $\mathbf{c} = H_1(\mathbf{comm} \parallel m)$ holds, proceed as the verifier of the sigma protocol described above (Item 4 of Protocol 1) component-wise, and output 1 if all subroutines return 1. Otherwise, output 0.
- $i \leftarrow$ **RRS.Revoke**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{sk}_{\mathsf{rev}}, m, \sigma)$ proceeds as follows:
  1. Execute **RRS.Verify**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, m, \mathsf{pk}_{\mathsf{rev}}, \sigma)$. Continue only if it outputs 1, otherwise return $\perp$.
  2. Repeat the following steps 3-5 for each $j \in [n\lambda]$ such that $c_j = 0$:
  3. Extract the response $resp_j$ and parse the commitment as $comm_j = (\mathbf{S}, ct)$ (where $\mathbf{S} \subset \mathcal{E}$) corresponding to the challenge bit $c_j = 0$ from σ.
  4. Extract the string ψ_j from the string ct in $comm_j$ with $\mathsf{sk}_{\mathsf{rev}}$ by recomputing $H_2(\mathsf{sk}_{\mathsf{rev}} * e * E) \oplus ct$.
  5. Use $resp_j$ to identify a member index. Say, during verification, the curve $resp_j * E = E'_k$ is in the array $\mathbf{S}$ for some $k \in [n]$. Then one can recover k by looking up the permutation ψ_j. Specifically, $resp_j * E = S_{\psi^{-1}(k)}$.
  6. After executing steps 3-5 for each 0 bit in the challenge, we will obtain a set of indices, one from each loop. Output the majority index (the most commonly occurring index in the set). If there is a tie for most common, randomly select one of the majority indices.

Remark 3. For readability, throughout most following proofs, we will use specific index letters with the following convention. The index $j \in [n\lambda]$ will be used for the j^{th} component of the sigma protocol, such as $comm_j \in \mathbf{comm}$ and $resp_j \in \mathbf{resp}$. We will use $i \in [n]$ to label the multiple sub-contents within each component, for example the individual commitment curves E'_{ji} (single per-round components, for example the permutations ψ_j, will simply be identified with the iteration index j). The index $k \in [n]$ will be used to indicate a specific member (for example, the one who signs or will be revealed).

Theorem 3.2. *Assuming H_1, H_2 are modeled by random oracles, if the parallelization problem of HHS is hard, then the scheme Protocol 2 has CPA-anonymity.*

Proof. In the CPA-anonymity experiment, the adversary is given all public keys, private keys, and a ring signature generated according to the protocol specification. The adversary's task is to identify the index of the signer after providing two indices and obtaining a signature from one of them.

Theorem 3.1 shows that the underlying sigma protocol is HVZK if the parallelization problem is hard. The proof is similar here. We will sketch the proof that the signature is simulatable by using the programmability of the random oracle H_1.

Firstly, the simulator randomly generates the challenge string $\mathbf{c} \in \{0,1\}^{n\lambda}$. Secondly, the simulator computes the corresponding $comm_j$ and $resp_j$ for each c_i, as in the proof of Theorem 3.1. Specifically, if $c_j = 1$, then the simulator generates e, r_i for $i \in [n]$ and $\psi \in \mathcal{S}_n$, and computes $comm_j$ and sets the response $resp_j$ accordingly. If $c_j = 0$, the simulator randomly picks $k \leftarrow [n]$, generates $e, r_i \leftarrow Cl$, $\psi \in \mathcal{S}_n$ and computes $E'_i = r_i * E_i$ for $i \in [n] - \{k\}$ and $E'_k = r_k * E$. The simulator computes $comm_j = (S, ct)$ where S follows the protocol specification and ct is a string generated uniformly at random of the correct length. The response $resp_j$ is set to r_k. Thirdly, the simulator simulates the oracle H_1 by assigning $\mathbf{c}$ to be the oracle value of $H_1(\mathbf{comm} \parallel m)$. The simulator ends the simulation by outputting $(\mathbf{comm}, \mathbf{c}, \mathbf{resp})$ as the signature. Note that if the adversary makes the oracle queries for H_1, then the simulator simulates the oracle in an on-the-fly manner and returns the corresponding values.

The simulated signature contains no information about the signer since the information about the permutation ψ is removed from the $comm_j$ for $c_j = 0$, which was supposed to be contained in ct. Also we have shown in the proof of Theorem 3.1, the simulation is perfectly indistinguishable for $c = 1$. The simulation is computationally indistinguishable for $c = 0$ by assuming the hardness of the parallelization problem for $(E, E', e * E)$ where E' is the revocation authority's public key. Therefore, when H_1, H_2 are modeled by random oracles, if the parallelization problem is hard, Protocol 2 has CPA-anonymity. □ □

Theorem 3.3. *Assuming that the vectorization problem of the hard homogeneous space is hard and H_1, H_2 are modelled as random oracles, Protocol 2 is unforgeable. Precisely, let $\mathcal{A}$ be an adversary with advantage $\mathsf{Adv}_{\mathsf{unf}}(\mathcal{A})$ against the unforgeability experiment, making n queries to the generation oracle O_{gen}, and Q queries to H_1 and the signing oracle O_{sig}. Then there exists an adversary $\mathcal{B}$ with advantage $\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})$ against the vectorization problem, such that*

$$\mathsf{Adv}_{\mathsf{unf}}(\mathcal{A}) \leq \frac{Q + \sqrt{Q^2 + 4\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})}}{2} + \frac{Q}{2^\lambda}.$$

Proof. The adversary $\mathcal{A}$ is given access to the oracles O_{gen}, O_{cor}, and O_{sig} and is required to output (σ, PK, m) where **RRS.Verify**$(\mathrm{PK}, m, \sigma) = 1$ while none of the corresponding secret keys for PK are given to the adversary by O_{cor} and (m, σ) is not listed in the past queries of the signing oracle, Q_{sig}.

We claim that signatures from O_{sig} are simulatable. We again rely on the programmability of the random oracle H_1. Say the adversary queries O_{sig} for signatures of an party P_k. The simulation for the oracle O_{sig} proceeds as follows.

1. The simulator randomly generates the challenge string $\mathbf{c} \in \{0,1\}^{n\lambda}$.
2. The simulator computes the corresponding $comm_j$ for each c_i as in the proof of Theorem 3.1. Specifically, if $c_j = 1$, then the simulator generates e, r_i for $i \in [n]$ and $\psi \in \mathcal{S}_n$, and computes $comm_j$. If $c_j = 0$, then the simulator generates $e, r_i \leftarrow Cl$, $\psi \in \mathcal{S}_n$ and computes $E'_i = r_i * E_i$ for $i \in [n] - \{k\}$ and $E'_k = r_k * E$.
3. The simulator simulates the oracle H_1 by assigning $\mathbf{c}$ to be the oracle value of $H_1(\mathbf{comm} \parallel m)$. The simulator ends the simulation by outputting the corresponding $(\mathbf{comm}, \mathbf{c}, \mathbf{resp})$ as the signature. Note that if the adversary makes oracle queries to H_1, then the simulator simulates the oracle in an on-the-fly manner and returns the corresponding values.

Note that, compared to the simulator in Theorem 3.1, the simulator of the oracle here doesn't need to randomly select the index k since it is provided by the adversary. In contrast to the real execution of signing by $\mathcal{P}_k$, the only difference lies in Step 2—that for the case $c_j = 0$ the simulator generates E'_k through $r_k * E$ with $r_k \leftarrow Cl$ while it is computed as $r'_k * E_k$ with $r'_k \leftarrow Cl$ in the real execution. Because the sampling of the group Cl is uniform, the distributions of generating E'_k via these two methods are identical. Hence, these parts are perfectly indistinguishable. It follows that the simulation is perfectly indistinguishable not only for the case $c_j = 1$ but also for $c_j = 0$.

We will prove unforgeability by using a standard hybrid argument with the following series of games. Let $\mathsf{Adv}_i(\mathcal{A})$ denote the advantage of the adversary $\mathcal{A}$ in Game_i for $i \in \{0, \ldots, 4\}$.

- Set Game_0 to be the original unforgeability experiment where the adversary $\mathcal{A}$ has access to oracles $O_{gen}, O_{sig}, O_{cor}$.
- Let Game_1 to be identical to Game_0 except that the signing oracle is replaced by the simulator as described above. If $H_1(\mathbf{comm} \parallel m)$ has been queried before in Step 3, then abort. Since for any $E \in \mathcal{E}$ the distribution of $a * E$ is uniform if a uniformly sampled from Cl, the min-entropy of the distribution is λ. (Recall that $|\mathcal{E}| \approx \sqrt{p} \geq 2^\lambda$) It follows that **comm** contains $n\lambda$ bits entropy. Hence, we have $\mathsf{Adv}_{\mathsf{unf}}(\mathcal{A}) := \mathsf{Adv}_0(\mathcal{A}) \leq \mathsf{Adv}_1(\mathcal{A}) + Q/2^{n\lambda} \leq \mathsf{Adv}_1(\mathcal{A}) + Q/2^\lambda$.
- Let Game_2 to be identical to Game_1 except that the adversary does not obtains keys from the oracle O_{gen}. Instead, a simulator samples $\mathsf{sk} \leftarrow Cl$, computes $\mathsf{pk} = \mathsf{sk} * E$, and returns $(\mathsf{sk}, \mathsf{pk})$ to the query. Hence, we have $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}_2(\mathcal{A})$.
- Let Game_3 to be identical to Game_2 except that we restrict the success condition: the challenger guesses an index $k \in \{1, \ldots, n\}$ uniformly at random. The adversary wins if the forged signature not only satisfies the original condition but pk_k is also an element of PK output by the adversary. Hence, we have $\mathsf{Adv}_2(\mathcal{A}) \leq n \cdot \mathsf{Adv}_3(\mathcal{A})$.

From an adversary $\mathcal{A}$ of Game_3, we can construct an adversary against the HHS vectorization problem by using the rewinding technique. Given $\mathcal{A}$ and the vectorization problem instance (E, E_{cha}), the reduction $\mathcal{B}$ can be made as follows,

1. Start the experiment and invoke the adversary $\mathcal{A}$. During the execution, the reduction also simulates the oracles H_1 and H_2 in an on-the-fly manner and returns the corresponding values. Also, the reduction simulates the signing oracle as described in Game_1,

2. Pick an index k from $[n]$ uniformly at random corresponding to the index that the adversary needs to guess (the additional requirement made in Game_3). To simulate the key generation oracle O_{gen}, the reduction follows the simulation in Game_2 and stores the secret keys, except assigning the instance E_{cha} as k-th public key. Note that the reduction has no secret keys for the k-th public key.

3. If the adversary inquires secret keys through the corruption oracle O_{cor}, the reduction $\mathcal{B}$ returns the corresponding stored secret key but aborts if $\mathcal{A}$ inquires the k-th secret key.

4. Execute $\mathcal{A}$ and obtain (m, σ, PK). We may assume $\mathsf{pk}_k \in \mathrm{PK}$. Parse σ as $(\mathbf{comm}, \mathbf{c}, \mathbf{resp})$.

5. Rewind $\mathcal{A}$ and obtain a second output $(m', \sigma', \mathrm{PK})$ such that the commitment components are the same (and so is PK). That is, we can parse $\sigma = (\mathbf{comm}, \mathbf{c}', \mathbf{resp}')$.

6. If $\mathbf{c} = \mathbf{c}'$, then abort. Otherwise, choose an index $j \in [n\lambda]$ such that $\mathbf{c}_j \neq \mathbf{c}'_j$. We have two elements $a, b \in Cl$ such that $a * E_{\mathsf{cha}} = E_j$ and $b * E = E_j$ for some E_j taken from $\mathbf{comm}$. Then we have $a^{-1}b \in Cl$ such that $(a^{-1}b) * E = E_{\mathsf{cha}}$.

We have shown the correctness. By the Forking lemma [BN06], if the success probability of $\mathcal{A}$ is $\mathsf{Adv}_3(\mathcal{A})$ with less than Q random oracle queries of H_1, then the advantage of this method to recover a secret key (i.e. to solve the vectorization problem) is greater than $\mathsf{Adv}_3(\mathcal{A}) * (\frac{\mathsf{Adv}_3(\mathcal{A})}{Q} - \frac{1}{2^{n\lambda}})$. Therefore, we have

$$\mathsf{Adv}_{\mathsf{unf}}(\mathcal{A}) \leq \frac{Q + \sqrt{Q^2 + 4\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})}}{2} + \frac{Q}{2^{\lambda}}.$$

□ □

Next, we provide two lemmas that we will use to show revocability of our RRS scheme.

Definition 3.1. *Let M be a binary matrix where the rows represent the possible commitments from the adversary and the columns represent the possible challenges given by the random oracle. A matrix entry is 1 if the adversary will succeed in the experiment, otherwise the entry is 0. Say the advantage of the adversary is ϵ. A row of M is said to be **heavy** if the fraction of 1's is not less than $\epsilon/2$.*

Lemma 3.4. *(Heavy Row Lemma [MR02]) Given an entry of 1 in the binary matrix M, the probability that the given entry lies in a heavy row is at least 1/2.* □

Lemma 3.5. *Let X be a random variable defined as $X = \sum_1^{n\lambda} B_i$ where B_i are from independent Bernoulli distributions of $p = 1/2$. Then, $Pr[X < n\lambda/4] \leq e^{-n\lambda/14}$.*

Proof. Since the expectation is $E[X] = n\lambda/2$, by applying the Chernoff bound, we have

$$Pr[X < (1 - 1/2)n\lambda/2] < (2/e)^{E[X]/2} < e^{-n\lambda/14}.$$

□

Theorem 3.6. *Assuming that the vectorization of hard homogeneous spaces is hard and H_1, H_2 can be modeled as random oracles, Protocol 2 has full-revocability. Precisely, let $\mathcal{A}$ be an adversary with advantage $\mathsf{Adv}_{\mathsf{rev}}(\mathcal{A})$ against the full-revocability experiment, making n queries to the generation oracle O_{gen} and Q queries to H_1 and the signing oracle O_{sig}. Then there exists an algorithm $\mathcal{B}$ with advantage $\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})$ against the HHS vectorization problem, such that*

$$\mathsf{Adv}_{\mathsf{rev}}(\mathcal{A}) \leq 2n\sqrt{\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})} + Q/2^{\lambda} + e^{-n\lambda/14}.$$

Proof. Recall that in the full-revocability experiment, the adversary is given three oracles $O_{gen}, O_{sig}, O_{cor}$ for key generation, signing, and corrupting, respectively. Further, the adversary possesses the revocation authority's public key $\mathsf{pk}_{\mathsf{rev}}$ and secret key $\mathsf{sk}_{\mathsf{rev}}$. The goal of the adversary is to forge a signature that will be opened to an index of an honest (uncorrupted) member by the revocation algorithm **RRS.Revoke**.

We will prove full-revocability by using the rewinding technique and a standard hybrid argument with the following series of games. Let $\mathsf{Adv}_i(\mathcal{A})$ denote the advantage of the adversary $\mathcal{A}$ in Game_i for $i \in \{0, \ldots, 4\}$.

- Set Game_0 to be the original revocability experiment where the adversary $\mathcal{A}$ has access to oracles $O_{gen}, O_{sig}, O_{cor}$.
- Let Game_1 to be identical to Game_0 except that the signing oracle is replaced by the simulator as described in Theorem 3.3. If $H_1(\mathbf{comm} \parallel m)$ has been queried before in Step 3, then abort. Since for any $E' \in \mathcal{E}$ the distribution of $a * E'$ is uniform if a is uniformly sampled from Cl, the min-entropy of the distribution is λ (recall that $|\mathcal{E}| \approx \sqrt{p} \geq 2^{\lambda}$). It follows that **comm** contains $n\lambda$ bits entropy. Hence, we have $\mathsf{Adv}_{\mathsf{rev}}(\mathcal{A}) := \mathsf{Adv}_0(\mathcal{A}) \leq \mathsf{Adv}_1(\mathcal{A}) + Q/2^{n\lambda} \leq \mathsf{Adv}_1(\mathcal{A}) + Q/2^{\lambda}$.
- Let Game_2 to be identical to Game_1 except that the adversary does not obtain keys from the oracle O_{gen}. Instead, a simulator samples $\mathsf{sk} \leftarrow Cl$, computes $\mathsf{pk} = \mathsf{sk} * E$, and returns $(\mathsf{sk}, \mathsf{pk})$ to the query. This behavior is indistinguishable, so we have $\mathsf{Adv}_1(\mathcal{A}) = \mathsf{Adv}_2(\mathcal{A})$.
- Let Game_3 be identical to Game_2 except that we restrict the success condition: the adversary wins if the forged signature not only satisfies the original condition, but also the Hamming weight of the challenge string $\mathbf{c}$ is not greater than $3n\lambda/4$. By using the Chernoff bound (Lemma 3.5), we have $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + e^{-n\lambda/14}$.
- Let Game_4 be identical to Game_3 except that we restrict the success condition again: the challenger guesses an index $k \in \{1, \ldots, n\}$ uniformly at random. The adversary wins if the forged signature not only satisfies the conditions of Game_3, but also is opened to the k-th member. Hence, we have $\mathsf{Adv}_3(\mathcal{A}) = n \cdot \mathsf{Adv}_4(\mathcal{A})$.

From an adversary $\mathcal{A}$ against Game_4, we construct a reduction to an adversary against the vectorization problem by using the rewinding technique. Given $\mathcal{A}$ and the vectorization problem instance (E, E_{cha}), the reduction $\mathcal{B}$ can be made as follows,

1. Start the experiment and invoke the adversary $\mathcal{A}$. During the execution, the reduction also simulates the oracles H_1 and H_2 in an on-the-fly manner and returns the corresponding values. Also, the reduction simulates the signing oracle as described in Game_1.
2. Pick an index $k \in [n]$ uniformly at random, as the index that the adversary needs to guess (the additional requirement made in Game_4). To simulate the key generation oracle O_{gen}, the reduction follows the simulation in Game_2 and stores the secret keys, except assigning the instance E_{cha} as the k-th public key. Note therefore that the reduction has no secret keys for the k-th public key.
3. If the adversary queries secret keys through the corruption oracle O_{cor}, the reduction returns the corresponding stored secret key, but aborts if $\mathcal{A}$ inquires after the k-th secret key.
4. Execute $\mathcal{A}$ and obtain (m, σ, PK). Let $I \subseteq [n\lambda]$ be a set of indices such that the responses $resp_{j \in I}$ corresponding to $c_{j \in I} = 0$ are individually opened (by **RRS.Revoke**) to the k-th honest party. Parse σ as $(\mathbf{comm}, \mathbf{c}, \mathbf{resp})$.
5. Rewind $\mathcal{A}$ and, obtain a second output $(m', \sigma', \mathrm{PK})$ such that the commitment components are the same. That is, we can parse $\sigma = (\mathbf{comm}, \mathbf{c}', \mathbf{resp}')$. Say $I' \subseteq [n\lambda]$ such that the responses $resp'_{j \in I'}$ corresponding to the challenges $c'_{j \in I'} = 0$ are individually opened (by **RRS.Revoke**) to an honest party, say $\mathcal{P}_k$ whose $\mathsf{pk}_k \in \mathrm{PK}$ and $\mathsf{pk}_k \notin Q_{cor}$.
6. If $\mathbf{c} = \mathbf{c}'$, then abort. Otherwise, let $j \in I'$ and say c'_j is flipped. That is, $c_j = 1$ and $c'_j = 0$ (or vice versa). Parse $resp_j = ((r_{ji})_{i \in [n]} \in Cl \parallel e \parallel \psi)$ and $resp'_j \in Cl$, then the secret key s_k for the member P_k is $(resp'_j) * (r_{jk})^{-1}$.

Correctness. For the case where $j \in I'$ and c'_j is flipped, from the procedure (Step 5) of **RRS.Revoke** that opens to the honest member P_k, we know $resp'_j * E = S_{\psi^{-1}(k)}$ where S and ψ are obtained from the corresponding $comm_j$ (we omit the subscript j for ease of notation). Also, we have $S_i = r_{j\psi(i)} * E_{\psi(i)}$ for any $i \in [n]$. Hence, we have

$$resp'_j * E = S_{\psi^{-1}(k)} = r_{jk} * E_k,$$

and therefore $(resp'_j) * (r_{jk})^{-1} * E = E_k = E_{\mathsf{cha}}$. That is, $(resp'_j) * (r_{jk})$ is the secret key for the member P_k and the solution for the vectorization problem.

Probability Analysis.

By the heavy row lemma 3.4, with probability $1/2$, the commitment **comm** lies in a heavy row. The advantage of the reduction, $\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})$, is not less than $\frac{\mathsf{Adv}_4^2}{4}$. Therefore, we have $\mathsf{Adv}_{\mathsf{rev}}(\mathcal{A}) \leq 2n * \sqrt{\mathsf{Adv}_{\mathsf{vec}}(\mathcal{B})} + Q/2^{\lambda} + e^{-n\lambda/14}$.

□

We end this subsection with the following result.

Theorem 3.7. *Protocol 2 is a collusion-resistant revocable ring signature scheme (and therefore also a group signature scheme—see Appendix C for details). That is, it satisfies correctness, unforgeability, CPA-anonymity, and full-revocability.*

Remark 4. We use the looser $Q/2^{\lambda}$ instead of $Q/2^{n\lambda}$ in Game_1 of both the proofs of unforgeability and full-revocability so that the numbers in the statement of the theorem will not change after the optimization in Section 3.3.

3.3 Cost Analysis and Optimization

In this section, we analyze the cost of the scheme and propose three methods to improve the efficiency. The summary is made in Table 2 by comparing the cost before and after the improvement. The techniques in this section consist make use of a computational binding commitment scheme, Merkle trees, and a pseudo-random number generator (PRNG). All of these can be instantiated by SHA-3, for example. As long as all the components have the same strength of security (regarding the computational binding, second preimage resistance, and pseudo-randomness, respectively), the changes will not downgrade the security strength of the signature scheme (see Remark 4). Thus all the results of the previous section apply to the scheme with these improvements applied.

Cost. The signature of a message m consists of (**comm**, **c**, **resp**), and the group public key is $(\mathrm{PK}, \mathsf{pk}_{\mathsf{rev}})$. Let λ represent the security parameter, $n = |\mathrm{PK}|$ be the number of members in the ring, and let the outputs of H_1, H_2 be of length $n\lambda$ and $N = max\{2\lambda, nlog(n)\}$ respectively. We represent the permutation element from $\mathcal{S}_n$ with $n\log(n)$ bits by using cycle notation.

The commitment component **comm** $= ((S_{ji})_{i\in[n]}, ct_j)_{j\in[n\lambda]}$ is of $n^2\lambda + n\lambda$ elements in the set $\mathcal{E}$ and $n\lambda$ binary strings of length N. The challenge component **c** is a binary string of length $n\lambda$. The response component is **resp** $= (resp_j)_{j\in n\lambda}$ where each $resp_j$ is either (r_j) or $((r_{ji})_{i\in[n]} \parallel e_j \parallel \psi_j)$. On average, it contains $n^2\lambda/2 + n\lambda$ elements of the group Cl and $n\lambda$ binary strings of length N.

The public key set PK is composed of n elements of the set $\mathcal{E}$, plus one element for the revocation key. In total, apart from the message, the ring signature for security parameter λ and n participants on average consists of $n^2\lambda + n + 1$ elements in $\mathcal{E}$, $n^2\lambda/2 + n\lambda$ elements in Cl and $3n\lambda^2 + n\lambda$ bits.

Optimization. Here we list several ways to compress the signature size. Firstly, we can apply the technique used by Beullens et al. [BKP20] to optimize their signature. Instead of generating n random elements $r_{ji} \in Cl$ for $r_{ji} * E_i$ for the j^{th} commitment, the prover uses the same $r_j \in Cl$ and commits to $Com(r * E_i, str_i)_{i\in[n]}$ where Com is a binding commitment scheme and str_i are random strings. The string str_i is of length 3λ to secure against the quantum collision algorithm by Brassard et al. [BHT98].

Furthermore, we can apply standard Merkle trees to the commitment component. By committing to $Com(r * E_{\psi(i)}, str_{\psi(i)})_{i\in[n]}$, the prover P_k reveals r and ψ for the challenge 1 or reveals rs_k and the path for challenge 0. Note that we are not using the index-hiding Merkle trees in [BKP20], since we need to extract the permutation element—the index-hiding Merkle trees are used to replace the permutation by using the lexicographical ordering of the hash digests, while we will simply use order-preserving concatenation. Additionally, we can use a pseudo-random number generator taking as input seeds sd_1 and sd_2 of 3λ bits to generate the $(str_i)_{i\in[n]}$ and the permutation element ψ, respectively. For the case of the challenge 1, then the prover sends to the verifier the seeds sd_1, sd_2 instead of the entire string $(str_i)_{i\in[n]}$ and the permutation element ψ. As a result, **comm** consists of $n\lambda$ elements in $\mathcal{E}$ and $6n\lambda^2$ bits.

Components	$\mathcal{E}$ (Set elements)	Cl (Group elements)	Bits
Commitment	$n^2\lambda + n\lambda$; $n\lambda$	-	$2nN\lambda$; $6n\lambda^2$
Challenge	-	-	$n\lambda$
Response	-	$n^2\lambda/2 + n\lambda$; $n\lambda$	$nN\lambda$; $n\lambda(3\lambda\log(n) + 6\lambda)/2$
Public Key	$n + 1$	-	-

Table 2: The signature size analysis of the construction where $N = max\{\lambda, n\log(n)\}$. The notation $a; b$ represents the original size of a and the improved size of b.

4 Conclusion

This work presents the following contributions. It introduces a better security notion for revocable ring signatures, establishes another connection between a ring signature and a group signature, and provides an approach to obtaining a group signature (as shown in Fig 1). Also, it provides a construction from hard homogeneous spaces. This work also presents the first efficient construction of an efficient group signature scheme from isogenies, and also the first construction of a post-quantum (collusion-resistant) revocable ring signature scheme.

As a limitation, we remark that the non-tight proof in Section 3 is in the classical random oracle model instead of the quantum random oracle model (QROM), a common obstacle for Σ-protocol-based schemes. Classically, after rewinding an adversary who can forge a signature, an extractor can get two responses for distinct challenges but for the same commitment by programming and extracting the random oracle (*programmability* and *extractability* of ROM). As a result, an extractor is able to extract a witness (a secret) of a computationally hard problem, by 2-special soundness. However, in QROM, this tactic will be detected by the adversary since observing the oracle query in a quantum state will make it collapse. A state-of-the-art work on Σ-protocols is [DFMS19]. However, the technique does not apply to this work since the signature scheme does not satisfy *quantum computationally unique responses*. In particular, when the challenge is 0, two distinct responses cannot be used to construct a solution for a hard computational assumption. Also, we admit that efficiency is not as competitive as the start-of-the-art lattice instances like [dPLS18]. We look forward to an improvement of our construction, developments in quantum security proof techniques or in isogeny cryptography to have a more efficient or secure isogeny-based group signature scheme in QROM.

Acknowledgement

We would like to acknowledge Shuichi Katsumata, Federico Pintore and anonymous reviewers for valuable discussion and feedback on this work. Their suggestions substantially improved the presentation of this paper. This research is funded by the Ministry for Business, Innovation and Employment in New Zealand.

References

[ACJT00] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, pages 255–270, 2000.

[ADMP20] Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In *ASIACRYPT 2020*, pages 411–439, 2020.

[AOS02] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In *ASIACRYPT 2002*, pages 415–432, Berlin, Heidelberg, 2002.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *International conference on the theory and applications of cryptographic techniques*, pages 56–73. Springer, 2004.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, pages 41–55, 2004.

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. In *European Symposium on Research in Computer Security*, pages 243–265. Springer, 2015.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve A. Schneider, editors, *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 117–136. Springer, 2016.

[BDK+21] Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. Cryptology ePrint Archive, Report 2021/1366, 2021. `https://ia.cr/2021/1366`.

[BFJ16] Jean-François Biasse, Claus Fieker, and Michael J. Jacobson. Fast heuristic algorithms for computing relations in the class group of a quadratic order, with applications to isogeny evaluation. *LMS Journal of Computation and Mathematics*, 19(A):371–390, 2016.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. *Lecture Notes in Computer Science*, page 163–169, 1998.

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYPT 2020*, pages 464–492, 2020.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *ASIACRYPT 2019*, pages 227–247, 2019.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of cryptology*, 17(4):297–319, 2004.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, pages 614–629, 2003.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 390–399. ACM, 2006.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 136–153, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[CHH+21] Kai-Min Chung, Yao-Ching Hsieh, Mi-Ying Huang, Yu-Hsuan Huang, Tanja Lange, and Bo-Yin Yang. Group signatures and accountable ring signatures from isogeny-based assumptions. Cryptology ePrint Archive, Report 2021/1368, 2021. `https://ia.cr/2021/1368`.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Annual International Cryptology Conference*, pages 61–76. Springer, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pages 56–72, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *ASIACRYPT 2018*, pages 395–427, 2018.

[CLRS10] Pierre-Louis Cayrel, Richard Lindner, Markus Rückert, and Rosemberg Silva. A lattice-based threshold ring signature scheme. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *Progress in Cryptology – LATINCRYPT 2010*, pages 255–272, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Cou06] Jean Marc Couveignes. Hard homogeneous spaces., 1997. *IACR Cryptology ePrint Archive*, 2006:291, 2006.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT '91*, pages 257–265, 1991.

[CWLY06] Sherman SM Chow, Victor K Wei, Joseph K Liu, and Tsz Hon Yuen. Ring signatures without random oracles. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 297–302, 2006.

[DFJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In *CRYPTO 2019*, pages 356–383, 2019.

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 759–789. Springer, 2019.

[DKL+20] Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: compact post-quantum signatures from quaternions and isogenies. In *ASIACRYPT 2020*, pages 64–93, 2020.

[DM20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020*, volume 12111 of *Lecture Notes in Computer Science*, pages 187–212. Springer, 2020.

[dPLS18] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 574–591. ACM, 2018.

[ELL+15] Martianus Frederic Ezerman, Hyung Tae Lee, San Ling, Khoa Nguyen, and Huaxiong Wang. A provably secure group signature scheme from code-based assumptions. In *ASIACRYPT 2015*, pages 260–285, 2015.

[EM18] Rachid El Bansarkhani and Rafael Misoczki. G-Merkle: A hash-based group signature scheme from standard assumptions. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 441–463, Cham, 2018. Springer International Publishing.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007*, pages 181–200, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Fuj11] Eiichiro Fujisaki. Sub-linear size traceable ring signatures without random oracles. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 393–415, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[GKV10] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In *ASIACRYPT 2010*, pages 395–412, 2010.

[JAC+17] David Jao, Reza Azarderakhsh, Matt Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalili, Brian Koziel, Brian Lamacchia, Patrick Longa, et al. SIKE: supersingular isogeny key encapsulation, 2017. `https://sike.org/`.

[JD11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer, 2011.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 525–537, 2018.

[KP17] Sudhakar Kumawat and Souradyuti Paul. A new constant-size accountable ring signature scheme without random oracles. In *International Conference on Information Security and Cryptology*, pages 157–179. Springer, 2017.

[Kup05] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput.*, 35(1):170–188, 2005.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *EUROCRYPT 2005*, pages 198–214, 2005.

[KY19] Shuichi Katsumata and Shota Yamada. Group signatures without NIZK: From lattices in the standard model. Cryptology ePrint Archive, Report 2019/221, 2019. `https://eprint.iacr.org/2019/221`.

[LGdSG20] Yi-Fu Lai, Steven D. Galbraith, and Cyprien Delpech de Saint Guilhem. Compact, efficient and UC-secure isogeny-based oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2020:1012, 2020.

[LLLS13] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013*, pages 41–61, 2013.

[LLM+07] Dennis Y W Liu, Kai Sui Liu, Yi Mu, Willy Susilo, and Duncan Shek Wong. Revocable ring signature. *Journal of Computer Science and Technology*, 22(6):785 – 794, 2007.

[LLNW14] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In *International workshop on public key cryptography*, pages 345–361. Springer, 2014.

[LNW15] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: Simpler, tighter, shorter, ring-based. In *PKC 2015*, pages 427–449, 2015.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *International Workshop on Selected Areas in Cryptography*, pages 184–199. Springer, 1999.

[LWW04] Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In Huaxiong Wang, Josef Pieprzyk, and Vijay Varadharajan, editors, *Information Security and Privacy*, pages 325–335, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[MR02] Silvio Micali and Leonid Reyzin. Improving the exact security of digital signature schemes. *J. Cryptol.*, 15(1):1–18, 2002.

[NIS20] NIST. National institute of standards and technology, 2020. `https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions`.

[NZZ15] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In Jonathan Katz, editor, *Public-Key Cryptography – PKC 2015*, pages 401–426, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Pei20] Chris Peikert. He gives c-sieves on the CSIDH. In *EUROCRYPT 2020*, pages 463–492, 2020.

[RS06] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptology ePrint Archive*, 2006:145, 2006.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT 2001*, pages 552–565, 2001.

[Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Sto09] Anton Stolbunov. Reductionist security arguments for public-key cryptographic schemes based on group action. *Norsk informasjonssikkerhetskonferanse (NISK)*, pages 97–109, 2009.

[Vél71] Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris, Séries A*, 273:305–347, 1971.

[XY04] Shouhuai Xu and Moti Yung. Accountable ring signatures: A smart card approach. In *Smart Card Research and Advanced Applications VI*, pages 271–286. Springer, 2004.

[ZLS+20] Xinyu Zhang, Joseph K. Liu, Ron Steinfeld, Veronika Kuchta, and Jiangshan Yu. Revocable and linkable ring signature. In Zhe Liu and Moti Yung, editors, *Information Security and Cryptology*, pages 3–27, Cham, 2020. Springer International Publishing.

A A Brief Survey of Ring Signature and Group Signature

A.1 Ring Signatures

Ring signature and group signature schemes share a lot of commonality, but differ in important ways. We begin with some history and definitions in the ring signature setting. Ring signatures [RST01, AOS02, CWLY06] were proposed later than group signatures, and are designed to provide signers with a level of anonymity while still retaining credibility. Specifically, a signer will choose an ad-hoc "ring" of other users' public keys, and create a signature with respect to that ring, such that anyone can publicly verify that the signature came from one of the keys in said ring, but cannot ascertain which. Unlike group signatures we will see below, there is no concept of "opening" or revoking anonymity in this plain setting since no trusted third party is assumed.

Further constructions in this vein resulted in variations such as

- Linkable ring signatures [LWW04, BKP20] - in which it can be publicly checked whether two signatures were generated by the same signer (still without revealing who that signer was).
- Traceable ring signatures [FS07, Fuj11] - similar to linkable ring signatures, but where two signatures by the same signer publicly identify who the signer was.

In this work we are interested in another extension of ring signatures, where the signer may grant a chosen authority the power to reveal who the signer was. Previous work on this idea has labelled it under two different names:

- Revocable ring signatures [LLM+07, ZLS+20]- there exists special keys granted authority to revoke anonymity and reveal the signer of a particular signature. This idea was introduced by [LLM+07] using bilinear pairings and a proof-of-knowledge.
- Accountable ring signatures [XY04, BCC+15, KP17] - similar to revocable ring signatures but with a stronger security model. The revocation authority is required to provide a publicly verifiable proof of the signers identity upon opening - not just return the identity itself.

There have been a number of ring signature constructions based on lattice assumptions, the first of these being [CLRS10]. Recently, a group-action based linkable ring signature was introduced by Beullens, Katsumata, and Pintore [BKP20], which can be instantiated using isogenies or lattices - both resulting in post-quantum constructions. This work is discussed further below. Our work is (to the best of our knowledge) the first to introduce a secure post-quantum (strong) revocable ring signature scheme and an isogeny-based group signature scheme. We also remark that there are no post-quantum accountable ring signature schemes in the literature.

A.1.1 Group Signatures

Group signatures have had a long and rich history of development since their initial introduction by Chaum and van Heyst [CvH91]. In the early stages of their development, most group signature schemes proposed had no formal model or definition of security and thus no security proof. A large step forward in efficiency was taken in [ACJT00], which also gave a proof of security and adaptive-coalition-resistance in the random oracle model under the strong RSA assumption. Further works [CL02, CL04] based on the LRSW assumption [LRSW99] followed, as well as a formalization of definitions and notions of security for (static) group signatures in [BMW03]. This formal treatment was extended to the case of dynamic groups in [BSZ05]. Kiayias and Yung [KY05] subsequently improved the joining mechanism for new members of the group. However, all these preceding works required either a trusted third party to generate the keys, or a very complicated multi-round interaction with the group manager.

Boneh, Boyen, and Shacham [BBS04] made large progress in terms of efficiency, with their presentation of short group signatures based on bilinear maps. The security of their scheme is based on the Strong Diffie-Hellman (SDH) assumption and Decisional Linear Diffie-Hellman assumption in groups with such a map, and is proven in the random oracle model. It builds on the previous work of Boneh, Lynn, and Shacham [BLS04] and Boneh and Boyen [BB04] which developed short signature schemes based on similar assumptions.

A number of works have aimed at bringing quantum resistance to the realm of group signatures. Notably, [ELL+15] introduced a code based group signature, while there has been a rich history of works with lattice based group signature schemes [GKV10, LLLS13, NZZ15, LNW15, KY19]. In 2018, Bansarkhani and

Misoczki [EM18] introduced the first hash-based group signature scheme. Being hash-based with standard assumptions - namely the existence of one-way functions - this scheme also claims post-quantum security. Finally, [KKW18], working in the MPC-in-the-head paradigm, introduced new zero-knowledge proofs based on symmetric primitives alone, which also result in quantum-resistant group and ring signature schemes.

B Proof for Theorem 3.1

Theorem B.1. *Protocol 1 is correct. Besides, when H is modeled as a random oracle, if the parallelization of HHS is hard, then Protocol 1 is non-abort honest verifier zero-knowledge (HVZK). That is, there exists a simulator given the challenge $c \leftarrow \{0,1\}$ that generates a transcript $(comm', c, resp')$ which is computationally indistinguishable from the one $(comm, c, resp)$ generated from the real execution of Protocol 1.*

Proof. **(Correctness)** When the challenge $c = 1$, the response is $((r_i)_{i\in[n]} \parallel e \parallel \psi)$. Since $S_i = E'_{\psi(i)} = r_{\psi(i)} * E_{\psi(i)}$, we have $S_i = r_{\psi(i)} * E_{\psi(i)}$ for any $i \in [n]$. Also, $(e * E \parallel H(e * E') \oplus \psi) = ct$ holds directly. When the challenge $c = 0$, the response is $s_k r_k$. In the verification phase $(s_k r_k) * E = r_k * E_k = E'_k \in S$ holds.

(HVZK) (Challenge c = 1) Given the challenge $c \in \{0,1\}$, if $c = 1$, then the simulator generates $e, r_i \leftarrow Cl$ for $i \in [n]$ and $\psi \leftarrow \mathcal{S}_n$ and computes $(E'_i)_i = (r_i * E_i)_i$ for each $i \in [n]$, the curve array $S = (E'_{\psi(i)})_{i\in[n]}$, and the string $ct = (e * E \parallel H(e * E') \oplus \psi)$. The simulator outputs $comm' = (S, ct)$, c and $resp' = ((r_i)_{i\in[n]} \parallel e \parallel \psi)$ and completes simulation.

Note that for an honest prover P_k, the secret s_k will not be used if the challenge c is 1. It follows that when $c = 1$, the simulation is perfectly indistinguishable. That is, the distribution of the transcripts generated by the simulator is identical to the distribution of real transcripts from the execution of the protocol.

(Challenge c = 0) If $c = 0$, then the simulator $\mathcal{S}_1$ randomly picks $k \leftarrow [n]$, generates $e, r_i \leftarrow Cl$ and computes $E'_i = r_i * E_i$ for $i \in [n] - \{k\}$ and $E'_k = r_k * E$. Additionally, the simulator generates $\psi \leftarrow \mathcal{S}_n$ and computes the curve array $S = (E'_{\psi(i)})_{i\in[n]}$, and the string $ct = (e * E \parallel H(e * E') \oplus \psi)$. After replacing ct by a string of the same length generated uniformly at random, the simulator outputs $(comm' = (S, ct'), c = 0, resp' = r_k)$ and completes simulation.

The situation is slightly different from the case of $c = 1$. The first difference is that the simulator generates E'_k through $r_k * E$ with $r_k \leftarrow Cl$, while it is generated as $r'_k * E_k$ with $r_k \leftarrow Cl$ in the real execution. Because the sampling of the group Cl is uniform, the distributions of these two methods of generating E'_k are identical. Hence, these parts are perfectly indistinguishable. The second difference, roughly speaking, is that $\mathcal{S}_1$ simulates P_k for a randomly chosen $k \in [n]$ while the real transcript is generated by a specific prover.

We claim that the output of $\mathcal{S}_1$ is computationally indistinguishable from real transcripts if the parallelization of HHS is hard. To see this, we apply the hybrid argument. Let $V(0)$ represent the algorithm that outputs the real transcript with a fixed input $c = 0$ after interacting with a prover. Let $V'(0)$ to be the same algorithm as $V(0)$ except that it replaces ct by a string of the same length generated uniformly at random. Specifically, while $V(0)$ generates $((S, ct), 0, resp)$, the algorithm $V'(0)$ will output $((S, ct'), 0, resp)$ where ct' is generated uniformly at random.

The distributions of the outputs of $V'(0)$ and $\mathcal{S}_1$ are identical (perfectly indistinguishable). Due to the assumption of the parallelization problem, it is infeasible to compute $e*E'$ when given $(E, E', e*E)$. Furthermore, since H is modeled by a random oracle of which the output is uniform, $V(0)$ and $V'(0)$ are computationally indistinguishable. Therefore, the transcripts generated by $\mathcal{S}_1$ are computationally indistinguishable from the real transcripts.

□

C Collusion-Resistant Revocable Ring Signature to Group Signature

In this section we will show how to construct a group signature from a collusion-resistant revocable ring signature, as defined in Section 2. We start with the definition of a group signature scheme.

C.1 Group Signature Schemes

Group signature schemes are similar to revocable ring signatures as outlined above. The key difference is that membership in the group is controlled strictly by the group manager (who also acts as the revocation authority

/ opener), rather than allowing ad-hoc collections of members for each signature.

Formally, a group signature scheme consists of six polynomial-time algorithms $\mathcal{GS}$ = (**GS.Setup**, **GS.KeyGen**, **GS.MKe**
GS.Sign, **GS.Verify**, **GS.Open**):

- $(pp) \leftarrow$ **GS.Setup**(1^λ) is a PPT algorithm returning a set of public parameters for the scheme (which will be implicitly used in the other functions).
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow$ **GS.KeyGen**(pp) is a PPT algorithm returning a private key and public key pair for a user in the scheme.
- $(\mathsf{gpk} = (\mathsf{pk}_{open}, \mathsf{pk}_{issue}), \mathsf{gmsk} = (\mathsf{sk}_{open}, \mathsf{sk}_{issue})) \leftarrow$ **GS.MKeyGen**(pp) is a PPT algorithm returning a private key and public key pair for the group manager in the scheme.
- $\sigma \leftarrow$ **GS.Sign**$(\mathsf{sk}_i, m, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk})$ is a randomized signing algorithm taking as inputs the secret key s_i, the message m, a set of public keys of the group $\{\mathsf{pk}_j\}$ (including pk_i), proof of membership π of all pk_i in the group, and a group public key gpk. It returns σ as the signature.
- $0 \mid 1 \leftarrow$ **GS.Verify**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk}, m, \sigma)$ is a deterministic verification algorithm taking as inputs the set of public keys pk_j of the group, proof π of membership of all the public keys in the group, the group public key gpk, the message m, and the signature σ. It returns 0 or 1 to represent validity of the signature.
- $i \leftarrow$ **GS.Open**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk}, \mathsf{sk}_{open}, m, \sigma)$ is a PPT algorithm taking as inputs the public keys pk_j of the group, proof of membership of these keys π, the group public key gpk, the revocation authority's secret key sk_{open}, the message m, and the signature σ. It returns an index i to represent the identity of the member who generated the signature. If the signature is invalid, $\perp$ is returned.

One can omit the membership proof π and identify gpk as $\{\mathsf{pk}_1, \cdots, \mathsf{pk}_n, \mathsf{pk}_{open}\}$ and obtain a static group signature scheme. For clarity and simplicity of exposition, we omit the details of members joining the group dynamically from our formal treatment. It is easy to add this functionality in by simply having the new member send their public key to the group manager, who will update the certificate π with their secret issuing key, to include said member. This certificate is then enough to verify that the new member is indeed part of the group. Removal of group members could additionally be accomplished with a revocation list or other similar methods, which are studied in the literature. The details of the certificate π are given in Section C.2

Note that we pass the set of public keys to functions separately from the group public key for this reason - in the case of a fixed group, the group public key may contain this set, but in the case of a dynamic group, in order for the group public key not to change upon each join, it shouldn't contain the list.

Remark 5. We emphasize the unfortunate difference in terminology between the GS and RRS settings - revocation in a ring signature scheme equates to opening in a group signature scheme (revealing which member created a given signature or revoking anonymity), while revocation in a group signature scheme denotes the removal of a member from the group. As our treatment of group and ring signatures is fairly separate, and the meaning should be clear from context, this hope this will not cause undue confusion.

The two key security notions for group signature schemes are *anonymity* and *traceability* [BMW03]. To prove security with respect to these two properties, we shall define four oracles to capture the powers of the adversary:

- $O_{gen}()$. The generation oracle, runs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow$ **GS.KeyGen**(pp), sets $Q_{gen} \leftarrow Q_{gen} \cup \{\mathsf{pk}_i\}$ and $\mathrm{SK}[\mathsf{pk}_i] = \mathsf{sk}_i$, and returns pk_i.
- $O_{cor}(\mathsf{pk}_i)$. The corruption oracle, takes a public key $\mathsf{pk}_i \in Q_{gen}$ as the input, sets $Q_{cor} \leftarrow Q_{cor} \cup \{\mathsf{pk}_i\}$, and returns the corresponding secret key $\mathsf{sk}_i = \mathrm{SK}[\mathsf{pk}_i]$.
- $O_{sig}(\mathsf{gpk}, \mathrm{PK}, \pi, m, \mathsf{pk}_i)$. The signing oracle, takes a group public key, a set of public keys $\mathrm{PK} \subseteq Q_{gen}$, a membership certificate π of PK under gpk, a message m, and a public key $\mathsf{pk}_i \in \mathrm{PK}$. It calls $\sigma \leftarrow$ **GS.Sign**$(\mathrm{SK}[\mathsf{pk}_i], \pi, m, \mathrm{PK}, \mathsf{gpk})$, sets $Q_{sig} \leftarrow Q_{sig} \cup \{(m, \sigma)\}$, and returns the signature σ.
- $O_{open}(\mathrm{PK}, \pi, m, \mathsf{gpk}, \sigma)$. The opening oracle, takes a set of public keys $\mathrm{PK} \subseteq Q_{gen}$, a message m, a group public key gpk, and a signature σ. It calls $i \leftarrow$ **GS.Open**$(\mathrm{PK}, \pi, \mathsf{gpk}, \mathrm{SK}[\mathsf{pk}_{open}], m, \sigma)$, sets $Q_{rev} \leftarrow Q_{rev} \cup \{\sigma\}$, and returns the index i.

Correctness and Unforgeability. These properties are identical to the case of RRS above - correctness ensures that valid signatures are always accepted and that Open() only fails if the signature is invalid, while unforgeability ensures that an adversary cannot generate an accepting signature for a group they do not know any secret keys for. We omit the formal definition as it is identical to the RRS one (replacing RRS with GS).

Full-anonymity. (CCA-anonymity) This property ensures anonymity for members even in the case of secret key exposure, and even if the identity of the signer of several other signatures has been revealed. The challenger C first generates public parameters pp for security parameter λ, and for all $i \in [n]$, keypairs $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \textbf{GS.KeyGen}(pp)$. The challenger also generates an opening keypair and issuing keypair $(\mathsf{gpk} = (\mathsf{pk}_{open}, \mathsf{pk}_{issue}), \mathsf{gmsk} = (\mathsf{sk}_{open}, \mathsf{sk}_{issue})) \leftarrow \textbf{GS.MKeyGen}(pp)$. A group signature scheme has full-anonymity if, for any PPT adversaries $\mathcal{A}_1, \mathcal{A}_2$ given $\{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_n, \mathsf{pk}_n), \mathsf{gpk}\}$ from C, and access to oracle O_{open}, there exists a negligible function $\mathsf{negl}()$ such that

$$\left| \Pr\left[\begin{array}{c} (m, i, j, \text{state}) \leftarrow \mathcal{A}_1^{O_{open}}(\{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_n, \mathsf{pk}_n)\}, \mathsf{gpk}) \\ b \leftarrow_R \{i, j\} \\ \sigma \leftarrow \textbf{GS.Sign}(\mathsf{sk}_b, m, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk}) \\ b^* \leftarrow \mathcal{A}_2^{O_{open}}(\text{state}, \sigma) \\ b^* = b \ \wedge \ \sigma \notin Q_{open} \end{array} \right] - \frac{1}{2} \right| < \mathsf{negl}(\lambda)$$

In the above definition, if the adversary's access to O_{open} is removed, then the notion is called *weak anonymity* or sometimes CPA-anonymity [dPLS18].

Full-traceability. Roughly speaking, full-traceability ensures that the power of the opening authority to reveal the creator of a particular signature cannot be obstructed even if the opening key is stolen and a number of group members collude. Formally, in the traceability experiment, the challenger executes generates public parameters pp for security argument λ, and runs the key generation algorithm $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \textbf{GS.KeyGen}(pp)$ for all $i \in [n]$. The challenger also generates an opening keypair and issuing keypair $(\mathsf{gpk} = (\mathsf{pk}_{open}, \mathsf{pk}_{issue}), \mathsf{gmsk} = (\mathsf{sk}_{open}, \mathsf{sk}_{issue})) \leftarrow \textbf{GS.MKeyGen}(pp)$. A group signature scheme has full-traceability if, for any PPT adversary $\mathcal{A}$ given $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{gpk}$ and gmsk from C, and access to oracles $O_{gen}, O_{sig}, O_{cor}$, there exists a negligible function $\mathsf{negl}()$ such that

$$\Pr\left[\begin{array}{c} (m, \sigma, \text{PK}, \pi) \leftarrow \mathcal{A}^{O_{gen}, O_{sig}, O_{cor}}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{gpk}, \mathsf{gmsk}) \\ i \leftarrow \textbf{GS.Open}(\text{PK}, \pi, \mathsf{sk}_{open}, m, \sigma) \\ \textbf{GS.Verify}(\text{PK}, \pi, \mathsf{gpk}, m, \sigma) = 1 \\ \wedge\, i \neq \perp \wedge\ \mathsf{pk}_i \in \text{PK} \cap Q_{gen} - Q_{cor} \ \wedge\ (m, \sigma) \notin Q_{sig} \end{array} \right] < \mathsf{negl}(\lambda)$$

Essentially, the goal of the adversary is to generate a valid signature (m, σ) such that the signature is opened to an uncorrupted member of the group or none of the members in the group.

C.2 Construction

Given a **collusion-resistant** revocable ring signature scheme, we demonstrate the construction of a secure group signature scheme.

Protocol 3.

- $\textbf{GS.Setup}(1^\lambda)$ will call $\textbf{RRS.Setup}(1^\lambda)$ to obtain pp.
- $\textbf{GS.MKeyGen}(pp)$ runs $\textbf{RRS.RevKeyGen}(pp)$ twice to obtain keypairs $(\mathsf{pk}_{open}, \mathsf{sk}_{open})$ and $(\mathsf{pk}_{issue}, \mathsf{sk}_{issue})$, and returns $(\mathsf{gpk} = (\mathsf{pk}_{open}, \mathsf{pk}_{issue}), \mathsf{gmsk} = (\mathsf{sk}_{open}, \mathsf{sk}_{issue}))$.
- We define $\textbf{GS.KeyGen}(pp)$ to call $\textbf{RRS.KeyGen}(pp)$ and obtain a keypair $(\mathsf{sk}_i, \mathsf{pk}_i)$.

  For a member to join a group, that member must then obtain a group membership certificate from the group manager. Formally, for a group with public key $\mathsf{gpk} = (\mathsf{pk}_{issue}, \mathsf{pk}_{open})$, and current users $Y =$

$\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$, user pk_{n+1} joins by setting $Y' = Y \cup \{\mathsf{pk}_{n+1}\}$ and setting the group certificate π to be a signature on (gpk, Y') with sk_{issue}. π defines the group members and is a proof of group membership for all users in Y'.

- **GS.Sign**$(\mathsf{sk}_i, m, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk})$ - The sign function takes as input a message m, the full set of public keys of the group, the certificate for the set of public keys π, a secret key sk_i corresponding to pk_i for some $1 \leq i \leq n$, and the group public key gpk. It generates a signature as follows: First verify that the certificate on the set of public keys is valid for gpk (i.e. it is a valid signature under pk_{issue}), and fail if not. Let pk_{open} be the opening key from gpk, and call **RRS.Sign**$(\mathsf{sk}_i, m, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{pk}_{open})$ to generate a signature σ. Return σ.

- **GS.Verify**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk}, m, \sigma)$ First verify that π is a valid certificate for all the public keys pk_j for the group gpk (under pk_{issue}), and return 0 if not. Then call and return **RRS.Verify**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, m, \sigma, \mathsf{pk}_{open})$.

- **GS.Open**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \pi, \mathsf{gpk}, \mathsf{sk}_{open}, m, \sigma)$ should check signature validity under **GS.Verify** (returning $\perp$ if invalid) and then simply call and return **RRS.Revoke**$(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}, \mathsf{sk}_{open}, m, \sigma)$.

Note that functionality to remove group members can also be added by again generating a new group membership certificate π with all but the revoked members' public keys, and publishing a revocation list with the old certificate on it, so that signatures under the old group become invalid.

Theorem C.1. *A group signature scheme constructed as in Protocol 3 is correct and unforgeable, and satisfies full-anonymity (CPA-anonymity, resp) and full-traceability, assuming the underlying revocable ring signature scheme has correctness, unforgeability, full-anonymity (CPA-anonymity, resp) and revocability.*

Proof. Here we prove that if the RRS scheme is secure, then the GS scheme constructed from it as above will also be secure. We prove each property sequentially:

Correctness and unforgeability Correctness and unforgeability directly follow from the correctness and unforgeability of the underlying RRS scheme - the group membership certificate checks ensure the group is well formed during signing and verification, and the underlying signature is generated and checked with the RRS scheme functions.

Full-Anonymity/CPA-anonymity Recall that the adversary is given a group $\mathrm{PK} = \{(\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_n, \mathsf{pk}_n)\}$, π, gpk and an opening oracle O_{open}. For CPA-anonymity, access to O_{open} is removed. The adversary outputs a message m and two indices i, j, and obtains a signature σ generated by either sk_i or sk_j on message m. If the adversary has non-negligible advantage in guessing which of the two keys was used to generate the message, the same adversary can beat the full-anonymity game of the underlying ring signature:

Given an instance of the RRS full-anonymity game, the adversary will generate a keypair $(\mathsf{pk}_{issue}, \mathsf{sk}_{issue})$, create a group membership certificate π on the set of public keys $\{\mathsf{pk}_i\}$ using sk_{issue}, and then pass PK, π and $\mathsf{gpk} = (\mathsf{pk}_{rev}, \mathsf{pk}_{issue})$ to the GS full-anonymity adversary. Queries to the GS opening oracle O_{open} can simply be forwarded to the RRS revocation oracle O_{rev}. The output of this adversary is a valid output to win the RRS full-anonymity game.

Thus if the RRS scheme has full-anonymity (CPA-anonymity), so too does the GS scheme built on it.

Full-Traceability The adversary is given here the set of public keys of the group PK, the membership certificate π, the group public key gpk, the group manager's secret key's $\mathsf{gmsk} = (\mathsf{sk}_{open}, \mathsf{sk}_{issue})$, and access to a corruption oracle O_{cor}, generation oracle O_{gen}, and a signing oracle O_{sig}. The adversary outputs a tuple $(m, \sigma, \mathrm{PK}, \pi)$. If the adversary has non-negligible advantage in winning the full-traceability game - that is, outputting a valid signature, message, and set of public keys for the group, such that Open() does not point to any corrupted member - then we can also use this adversary to win the full-revocability game on the base RRS:

Given an instance of the RRS full-revocability game, the adversary will generate a keypair $(\mathsf{pk}_{issue}, \mathsf{sk}_{issue})$, and then pass PK, $\mathsf{gpk} = (\mathsf{pk}_{rev}, \mathsf{pk}_{issue})$ and $\mathsf{gmsk} = (\mathsf{sk}_{rev}, \mathsf{sk}_{issue})$ to the GS full-traceability adversary. The adversary will return a tuple $(m, \sigma, \mathrm{PK}, \pi)$. Then (m, σ, PK) is, with non-negligible advantage, a successful output for the RRS full-revocability game.

Thus if the RRS scheme has full-revocability, the GS scheme built on it has full-traceability.

□