Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices

Ward Beullens[1], Shuichi Katsumata[2], Federico Pintore[3]

[1] imec-COSIC, KU Leuven, Belgium
ward.beullens@esat.kuleuven.be
[2] National Institute of Advanced Industrial Science and Technology (AIST), Japan
shuichi.katsumata@aist.go.jp
[3] Mathematical Institute, University of Oxford, United Kingdom
federico.pintore@maths.ox.ac.uk

Abstract We construct efficient ring signatures (RS) from isogeny and lattice assumptions. Our ring signatures are based on a logarithmic OR proof for group actions. We instantiate this group action by either the CSIDH group action or an MLWE-based group action to obtain our isogeny-based or lattice-based RS scheme, respectively. Even though the OR proof has a binary challenge space and therefore requires a number of repetitions which is linear in the security parameter, the sizes of our ring signatures are small and scale better with the ring size N than previously known post-quantum ring signatures. We also construct linkable ring signatures (LRS) that are almost as efficient as the non-linkable variants. The isogeny-based scheme produces signatures whose size is an order of magnitude smaller than all previously known logarithmic post-quantum ring signatures, but it is relatively slow (e.g. 5.5 KB signatures and 79 s signing time for rings with 8 members). In comparison, the lattice-based construction is much faster, but has larger signatures (e.g. 30 KB signatures and 90 ms signing time for the same ring size). For small ring sizes our lattice-based ring signatures are slightly larger than state-of-the-art schemes, but they are smaller for ring sizes larger than $N \approx 1024$.

1 Introduction

Ring signatures (RS), introduced by Rivest, Shamir, and Tauman [29] allow a person to sign a message on behalf of a group of people (called ring), without revealing which person in the ring signed the message. A ring signature is required to be *unforgeable*, meaning that one cannot produce a signature without having the secret key of at least one person in the ring, and *anonymous*, meaning

This work was supported by CyberSecurity Research Flanders with reference number VR20192203 and the Research Council KU Leuven grants C14/18/067 and STG/17/019. Ward Beullens is funded by FWO SB fellowship 1S95620N. Shuichi Katsumata was supported by JST CREST Grant Number JPMJCR19F6 and JSPS KAKENHI Grant Number JP19H01109..

that it is impossible to learn which person produced the signature. The original motivation behind ring signatures is to allow a whistleblower to leak information without revealing their identity, while still adding credibility to the information by proving that it was leaked by one of the people in the ring. Linkable ring signatures (LRS) are an extension where one can publicly verify whether two messages were signed by the same person or not. This variant has found applications in e-voting and privacy-friendly digital currencies. In both cases, to protect users' privacy it is important to have at disposal a (linkable) ring signature scheme that can efficiently support very large ring sizes.

The security of many known (linkable) ring signatures relies on the hardness of factoring integers or computing discrete logarithms in finite cyclic groups. Unfortunately, these problems can be solved in quantum polynomial time [30], and hence all the schemes based on them would be no longer secure in the presence of adversaries with access to a sufficiently powerful quantum computer. To resolve this issue it is necessary to consider hard problems that resist attacks from quantum computers. Post-quantum ring signature schemes scaling polylogarithmically with the ring size have been constructed from symmetric cryptographic primitives [20, 11] and the hardness of lattice problems [15, 16, 4, 31, 23].

1.1 Our contributions

In this paper, we introduce a logarithmic OR proof for group actions and we then use it to construct concretely efficient logarithmic ring signatures and linkable ring signatures from isogeny and lattice assumptions. Our (linkable) ring signature schemes are realized by first constructing a generic (linkable) ring signature scheme based on a group action that satisfies certain cryptographic properties, and then instantiating this group action by either the CSIDH group action [8] or a MLWE-based group action. This is, to the best of our knowledge, the first concrete construction of (linkable) ring signatures from isogeny-assumptions with logarithmic signature size.

An advantage of our schemes is that the signature size scales very well with the ring size N, even compared to other post-quantum logarithmic (linkable) ring signatures, since the only dependence on N is due to the signatures containing a small number of paths (in the clear) in Merkle trees of depth $\log N$. Therefore, the term in the signature size that depends on $\log N$ is independent of the CSIDH or lattice parameters. All previous works that relied on a hidden path in a Merkle tree had to prove the consistency of a Merkle hash in zero-knowledge. Therefore, the multiplicative factor of $\log N$ was much larger than ours. The very mild dependence on $\log N$ of our schemes can be observed in Figure 1, where we see that for our lattice-based ring signature scheme a signature for ring size $N = 2048$ is only 17% larger than a signature for ring size $N = 2$.

For efficiency and convenience we chose to implement our (linkable) ring signature scheme with parameter sets from pre-existing signature schemes: for our iso-

geny instantiations we consider the CSIDH-512 parameter set, used by CSI-FiSh [6], while for our lattice instantiation we use the Dilithium II parameter set. This allows us to reuse large portions of code from CSIDH, CSI-FiSh, and Dilithium implementations. The signature size and signing speed of our implementations are shown in Figure 1. The signature size can be estimated as $\log N + 2.7$ KB for the isogeny-based instantiation and as $0.5 \log N + 29$ KB for the lattice-based instantiation. For ring size $N = 8$ our lattice-based instantiation has a signing time of 90 ms, faster than our isogeny-based instantiation (79 s) by almost 3 orders of magnitude.

Table 1 lists the signature size of our ring signatures and those of some other post-quantum ring signatures[1]. Not surprisingly, the signature size of our isogeny-based (linkable) ring signature is very small compared to the other post-quantum proposals. In particular, it is an order of magnitude smaller. However, we should notice that it is hard to make a meaningful comparison between our schemes and schemes which claim different security levels. For the lattice-based instantiations, we compute the signature size for a parameter set that achieves NIST security level II[2] (see the third row in Table 1) to allow for a fair comparison with the work of Esgin et al. [16]. We observe that for small ring sizes our lattice-based signatures are larger than those of Esgin et al., but for ring sizes larger than $N \approx 1024$ our signatures are the smallest.

Since our isogeny scheme is compact and our lattice scheme is fast, we call our schemes, respectively, the "Compact And Linkable Anonymous Message Authentication fRom Isogenies" (Calamari) and the "Fast Authentication with Linkable Anonymity From Lattices" (Falafl). We give the names "Faafl" and "Camari" to the isogeny and lattice (non-linkable) ring signatures, respectively.

1.2 Technical overview

Our (linkable) ring signature scheme is based on a generalisation to group actions of the classical sigma protocol for the Graph Isomorphism Problem. Let $\star : G \times \mathcal{X} \to \mathcal{X}$ be a group action and fix $X_0 \in \mathcal{X}$. To prove knowledge of a group element g such that $g \star X_0 = X$, the prover uniformly samples $r \in G$, and sends $R = r \star X$ as commitment. The verifier responds with a random challenge bit c. If $c = 0$ the prover sends $\mathsf{resp} = r + g$, while they send $\mathsf{resp} = r$ if $c = 1$. The verifier checks whether $\mathsf{resp} \star X_0 = R$ when $c = 0$, and whether $\mathsf{resp} \star X = R$ when $c = 1$.

A key observation is that the verification algorithm is independent of X when the challenge bit is 0. This allows us to design the following OR proof for group

[1] We compare only signature sizes since, to the best of our knowledge, ours is the only post-quantum logarithmic (linkable) ring signature with an implementation.

[2] We used the Dilithium III parameters, 168-bit seeds and commitment randomness, and a challenge space of size 2^{168}, which suffices to achieve NIST level II for low MAXDEPTH.

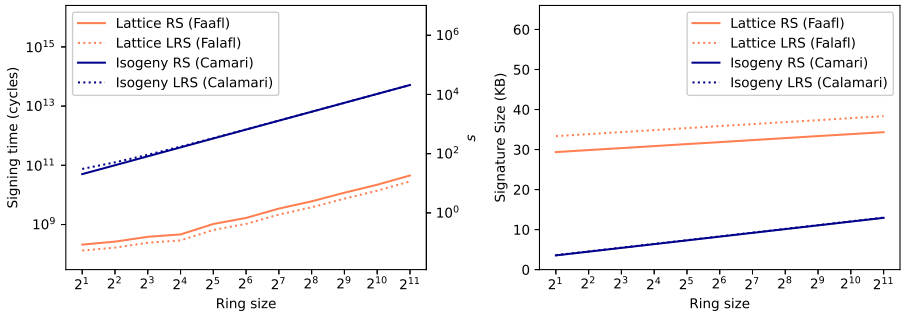

Figure 1. Signing time (left) and signature size (right) of our isogeny-based and lattice-based (linkable) ring signatures. The left and right scales in the figure of signing time correspond to the isogeny-based and lattice-based schemes, respectively. Signing time is measured on an Intel i5-8400H CPU core.

	N					hardness	Security
	2^1	2^3	2^6	2^{12}	2^{21}	assumption	Level
Calamari	3.5	5.4	8.2	14	23	CSIDH-512	*
Falafl	29	30	32	35	39	MSIS, MLWE	NIST 1
Falafl for 2	49	50	52	55	59	MSIS, MLWE	NIST 2
RAPTOR[24]	~ 2.5	~ 10	81	5161	/	NTRU	100 bits
EZSLL[15, 16]	18	19	31	59	148	MSIS, MLWE	NIST 2
KKW[20]	/	/	250	456	/	LowMC	NIST 5

Table 1. Comparison of the signature size (KB) of some concretely efficient post-quantum ring signature schemes.
* 128 bits of classical security and 60 bits of quantum security [27].

actions. For some $X_0, X_1, \cdots, X_N \in \mathcal{X}$, the prover wants to prove knowledge of $g \in G$ such that $g \star X_0 = X_I$ for some $I \in \{1, \cdots N\}$. Then they start by simulating a commitment for each X_i with $i \in \{1, \cdots, N\}$, so that they can respond to the challenge $c = 1$, and send these commitments in a random order to the verifier. If the verifier sends $c = 1$, we let the prover respond for all the commitments (and hence I is not leaked). If the verifier sends the challenge bit $c = 0$, the prover can answer the I-th challenge, but not the other challenges, because they do not know group elements g_i such that $g_i \star X_0 = X_i$ for $i \neq I$. Therefore, we let the prover respond only to the I-th challenge. This does not reveal I, because verification is independent of X_I.

More concretely, the prover sends N elements $R_1 = r_1 \star X_1, \cdots, R_N = r_N \star X_N$ in a random order to the verifier, where the r_i are chosen uniformly at random from G. Then, after the verifier sends a challenge bit c, the prover responds with $\mathsf{resp} = r_I + g$ if the challenge bit c is 0, or responds with $r_1, \cdots, r_N$ in case $c = 1$. The verifier checks whether $\mathsf{resp} \star X_0 \in \{R_1, \cdots, R_N\}$ in case $c = 0$ and

whether $\{r_1 \star X_1, \cdots, r_N \star X_N\} = \{R_1, \cdots, R_N\}$ in case $c = 1$. Here, note that the commitments are sent in a random order, so the response hides the index I in case $c = 0$.

Since the prover sends N elements $R_1, \cdots, R_N$ as commitment and N group elements $r_1, \cdots, r_N$ as response in case $c = 1$, it looks like the proof size is linear in N, and that there is no improvement over the generic OR proof. However, since the r_i are chosen at random, they can be generated from a pseudorandom number generator (PRG) instead, which reduces the communication cost to just sending a seed as the response in case $c = 1$. Moreover, instead of sending all the R_i we can commit to them using a Merkle tree and only send the root as the commitment. To make verification possible, the prover then sends a path in the Merkle tree to the verifier as part of the response in case $c = 0$. This makes the total proof size logarithmic in N, a clear improvement over generic OR proofs. Furthermore, since for some group actions it is more efficient to compute N group actions $r \star X_i$ with the same element $r \in G$ rather than computing N group actions $r_i \star X_i$ with distinct r_i, in our protocol we set $r_1 = \cdots = r_N = r$. Given that this would break the zero-knowledge property of the Sigma protocol, we replace each R_i by a hiding commitment $\mathsf{Com}(R_i, \mathsf{bits}_i)$, and we let the prover include bits_I in the response in case $c = 0$.

To enlarge the challenge space of the OR proof for group actions, we run parallel executions of it and then we obtain a ring signature scheme by applying the Fiat-Shamir transform to the OR proof. To avoid multi-target attacks similar to those of Dinur and Nadler [12] we made a detailed security proof in the random oracle model, with concrete expressions for the security loss in each step of the proof. This led us to include a unique salt value in each signature and to carefully separate the domain of various calls to the random oracles. We notice that in choosing our concrete parameters, as per usual, we ignore the artificial reduction loss incurred by the rewinding argument of Fiat-Shamir (since no attacks that can exploit this loss are known).

We instantiate the group action $\star$ by either the CSIDH group action or the MLWE group action defined as:

$$\star : R_q^{n+m} \times R_q^m : (\mathbf{s}, \mathbf{e}) \star \mathbf{t} \mapsto \mathbf{A} \star \mathbf{s} + \mathbf{e} + \mathbf{t}$$

where $R_q = \mathbb{Z}_q[X]/(X^d + 1)$ and $\mathbf{A}$ is a matrix belonging to $R_q^{m \times n}$. To achieve one-wayness it is necessary to restrict the domain to $S_\eta^{n+m} \times R_q^m$, where S_η is the set of elements of R_q with coefficients bounded in absolute value by η. In this case we need to use the Fiat-Shamir with aborts technique [25] to ensure that the signatures do not leak the secret key.

In order to obtain a linkable ring signature scheme, we expand our OR proof to an OR proof *with tag* where, given two group actions $\star : G \times \mathcal{X} \to \mathcal{X}$, $\bullet : G \times \mathcal{T} \to \mathcal{T}$ and a list of elements $X_0, X_1, \cdots, X_N \in \mathcal{X}$ and $T_0, T \in \mathcal{T}$, the

prover proves knowledge of $g \in G$ such that $g \star X_0 = X_I$ for some $I \in \{1, \cdots, N\}$ and $g \bullet T_0 = T$. This naturally leads to a linkable signature scheme. The signer includes the tag $T = g \bullet T_0$ in the signature and then proves knowledge of g. Two signatures can be linked by checking if the tags are equal (or close with respect to a well defined metric). We require a number of properties from $\star$ and $\bullet$ to make the linkable ring signature secure (see Definition 15). For example, it should not be possible to learn $g \star X_0$ given $g \bullet T_0$, because that would break the anonymity of the linkable ring signature. We give instantiations of $\star$ and $\bullet$ based on the CSIDH group action (where we put $g \bullet X := (2g) \star X$) or based on the hardness of MLWE and MSIS.

Finally, we would like to point out some optimization tricks that allow to further lower the size of the signatures. Since our base protocol (either the OR proof or the OR proof with tag) has a binary challenge, we must execute parallel repetitions to lower the soundness error to make it useable for (linkable) ring signatures. A naive way to accomplish this would be to run the OR proof (with tag) λ-times, where λ is the security parameter. However, since opening to $c = 1$ (which requires communicating only a single seed value) is much cheaper than opening to $c = 0$, we can do much better. Specifically, we choose integers M, K such that $\binom{M}{K} \geq 2^\lambda$ and do $M > \lambda$ executions of the protocol of which exactly K executions are chosen to have challenge bit 0. Setting $K \ll \lambda$, we get a noticeable gain in the signature size. Moreover, since we now only need to open to seed values in most of the parallel runs, we use a *seed tree* to further lower the signature size. Informally, the seed tree generates a number of pseudorandom values and can later disclose an arbitrary subset of them, without revealing information on the remaining values. Further details on our optimization tricks can be found in Section 3.4.

Roadmap. In Section 2 we provide some necessary preliminaries. In Section 3 we first define an *admissible group action*, then we construct a base OR proof for group actions with binary challenge space, which we then extend to a main OR proof with exponential challenge space. Finally we apply the Fiat-Shamir transform to obtain a ring signature scheme. Section 4 follows the same structure: we define an *admissible pair of group actions*, for which we construct an OR proof with tag, which we convert into a linkable ring signature. In Section 5 we instantiate the group actions from isogeny and lattice assumptions. Finally, in Section 6 we discuss our parameter choices and implementation results, and we draw some conclusions.

2 Preliminaries

A note on random oracles. Throughout the paper, we instantiate several standard cryptographic primitives such as pseudorandom number generators (PRG in short, and denoted by Expand) and commitment schemes by hash functions modeled as a random oracle $\mathcal{O}$. We always assume the input domain of the ran-

dom oracle is appropriately separated when instantiating several cryptographic primitives by one random oracle. With abuse of notation, we may occasionally write, for example, $\mathcal{O}(\mathsf{Expand}\|\cdot)$ instead of $\mathsf{Expand}(\cdot)$ to make the usage of the random oracle explicit. Here, we identify Expand with a unique string when inputting it to $\mathcal{O}$. Moreover, we denote by $\mathcal{A}^{\mathcal{O}}$ an algorithm $\mathcal{A}$ that has black-box access to $\mathcal{O}$, and we may occasionally omit the superscript $\mathcal{O}$ when the meaning is clear. Finally, for a precise definition of relaxed Sigma protocol in the Random Oracle Model we refer to [5, Sec. 2.1].

2.1 Ring signatures

In this subsection, we review the definition of ring signatures.

Definition 1 (Ring signature scheme). *A ring signature scheme Π_{RS} consists of four PPT algorithms* $(\mathsf{RS.Setup}, \mathsf{RS.KeyGen}, \mathsf{RS.Sign}, \mathsf{RS.Verify})$ *such that:*

$\mathsf{RS.Setup}(1^\lambda) \to \mathsf{pp}$: *On input a security parameter 1^λ, it returns public parameters* pp *used by the scheme.*

$\mathsf{RS.KeyGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$: *On input the public parameters* pp*, it outputs a pair of public and secret keys* $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{RS.Sign}(\mathsf{sk}, \mathsf{M}, \mathsf{R}) \to \sigma$: *On input a secret key* sk*, a message* M*, and a list of public keys, i.e., a* ring*,* $\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$*, it outputs a signature σ.*

$\mathsf{RS.Verify}(\mathsf{R}, \mathsf{M}, \sigma) \to 1/0$: *On input a ring* $\mathsf{R} = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_N\}$*, a message* M*, and a signature σ, it outputs either* 1 *(accept) or* 0 *(reject).*

We require a ring signature scheme Π_{RS} to satisfy the following properties: correctness, full anonymity, and unforgeability. Informally, correctness means that verifying a correctly generated signature will always succeed. Anonymity means it should not be possible to learn which secret key was used to produce a signature, even for an adversary that knows the secret keys for all the public keys in the ring. Finally, unforgeability means that it should be impossible to forge a valid signature without knowing a secret key that corresponds to one of the public keys in the ring. For formal security definitions we refer to [5, Sec. 2.2].

2.2 Linkable ring signatures

Linkable ring signatures are a variant of ring signatures where anyone can efficiently check if two messages were signed with the same secret key. Below we review the formal definition.

Definition 2 (Linkable ring signature scheme). *A linkable ring signature scheme Π_{LRS} consists of the four PPT algorithms of a ring signature scheme and one additional PPT algorithm* $\mathsf{LRS.Link}$ *such that:*

$\mathsf{LRS.Link}(\sigma_0, \sigma_1) \to 1/0$: *On input two signatures σ_0 and σ_1, it outputs either* 1 *or* 0*, where* 1 *indicates that the signatures were produced with the same secret key.*

In addition to the correctness property, we require a linkable ring signature scheme Π_{LRS} to satisfy linkability, linkable anonymity, and non-frameability. Informally, linkability means that, if an adversary produces more than k signatures with a ring of k (potentially malformed) public keys, then the $\mathsf{LRS.Link}$ algorithm will output 1 on at least one pair of signatures. Linkable-anonymity means that an adversary cannot tell which secret key was used to produce a signature. In contrast to the ring signature case, the adversary is not given all the secret keys, otherwise they could use the linkability property to deanonymize the signer. Finally, the non-frameability property says it should be impossible for an adversary to produce a valid signature that links to a signature produced by an honest party. For formal security definitions we refer to [5, Sec. 2.3].

Remark 3 (Unforgeability). We can also require a linkable ring signature to be unforgeable, as defined above for a ring signature. However, it can be shown that unforgeability is implied by linkability and non-frameability.

2.3 Isogenies and ideal class group actions

Let $\mathbb{F}_p$ be a prime field, with $p \geq 5$, and E a supersingular elliptic curve defined over $\mathbb{F}_p$. The ring $\mathrm{End}_p(E)$ of all endomorphisms of E that are defined over $\mathbb{F}_p$ is isomorphic to an order $\mathcal{O}$ of the field $\mathbb{K} = \mathbb{Q}(\sqrt{-p})$ [8]. The invertible fractional ideals of $\mathcal{O}$ form an abelian group whose quotient by the subgroup of principal fractional ideals is finite, called the *ideal class group* of $\mathcal{O}$ and denoted by $\mathcal{C}\ell(\mathcal{O})$. The ideal class group $\mathcal{C}\ell(\mathcal{O})$ acts freely and transitively on the set $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, which contains all supersingular elliptic curves E over $\mathbb{F}_p$ - modulo isomorphisms defined over $\mathbb{F}_p$ - such that there exists an isomorphism between $\mathcal{O}$ and $\mathrm{End}_p(E)$ mapping $\sqrt{-p} \in \mathcal{O}$ into the Frobenius endomorphism $\pi : (x, y) \mapsto (x^p, y^p)$. We denote this action by $*$. Recently, it has been used to design several cryptographic primitives [8, 10, 6], whose security proofs rely on (variations of) the Group Action Inverse Problem (GAIP), defined as follows:

Definition 4 (Group Action Inverse Problem (GAIP)). *Let $[E_0]$ be a an element in $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, where $p \geq 5$ is an odd prime. Given $[E]$ sampled uniformly at random from $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, the GAIP_p problem consists in finding an element $[\mathfrak{a}] \in \mathcal{C}\ell(\mathcal{O})$ such that $[\mathfrak{a}] * [E_0] = [E]$.*

For the security of the isogeny-based instantiations of our (linkable) ring signature schemewe will rely on a newly-introduced hard problem, called the Squaring Decisional CSIDH Problem ($\mathsf{sdCSIDH}$ in short).

Definition 5 (Squaring Decisional CSIDH ($\mathsf{sdCSIDH}$) Problem). *Let $[E_0]$ be an element in $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$, where $p \geq 5$ is an odd prime. Given $[\mathfrak{a}]$ sampled uniformly at random from $\mathcal{C}\ell(\mathcal{O})$, the $\mathsf{sdCSIDH}_p$ problem consists in distinguishing the two distributions $([\mathfrak{a}] * [E_0], [\mathfrak{a}]^2 * [E_0])$ and $([E], [E'])$, where $[E], [E']$ are both sampled uniformly at random from $\mathcal{E}\ell\ell_p(\mathcal{O}, \pi)$.*

In analogy with the classical group-based scenario [3], we assume the above problem is equivalent to the decisional CSIDH problem, recently used in [14, 9].

2.4 Lattices

For positive integers n and q, let R and R_q denote the rings $\mathbb{Z}[X]/(X^n+1)$ and $\mathbb{Z}[X]/(q, X^n+1)$, respectively. Norms over R are defined through the coefficient vectors of the polynomials, which lie over $\mathbb{Z}^n$. Norms over R_q are defined in the conventional way by uniquely representing coefficients of polynomials in R_q by elements in the range $(-q/2, q/2]$ when q is even and $[-(q-1)/2, (q-1)/2]$ when q is odd (see, for example, [13] for more details).

The hard problems that we rely on for our lattice-based schemes are the *module short integer solution* (MSIS) problem and *module learning with errors* (MLWE) problem, first introduced in [22].

Definition 6 (Module short integer solution Problem). *Let n, q, k, ℓ, γ be positive integers. The advantage for the (Hermite normal form)* module short integer solution problem $\mathsf{MSIS}_{n,q,k,\ell,\gamma}$ *for an algorithm $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathsf{MSIS}}_{n,q,k,\ell,\gamma}(\mathcal{A}) = \Pr\left[0 < \|\mathbf{u}\|_\infty \leq \gamma \wedge [\mathbf{A} \mid \mathbf{I}] \cdot \mathbf{u} = \mathbf{0} \;\middle|\; \mathbf{A} \leftarrow R_q^{k\times\ell}; \mathbf{u} \leftarrow \mathcal{A}(\mathbf{A})\right].$$

Definition 7 (Module learning with errors Problem). *Let n, q, k, ℓ be positive integers and D a probability distribution over R_q. The advantage for the* decisional module learning with errors problem $\mathsf{dMLWE}_{n,q,k,\ell,D}$ *for an algorithm $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathsf{dMLWE}}_{n,q,k,\ell,D}(\mathcal{A}) = \left|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{As}+\mathbf{e}) \to 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{v}) \to 1]\right|,$$

where $\mathbf{A} \leftarrow R_q^{k\times\ell}$, $\mathbf{s} \leftarrow D^\ell$, $\mathbf{e} \leftarrow D^k$ and $\mathbf{v} \leftarrow R_q^k$.

The advantage for the search learning with errors problem $\mathsf{sMLWE}_{n,q,k,\ell,D}$ *is defined analogously to above as the probability that $\mathcal{A}(\mathbf{A}, \mathbf{v} := \mathbf{As}+\mathbf{e})$ outputs $(\tilde{\mathbf{s}}, \tilde{\mathbf{e}})$ such that $\mathbf{A}\tilde{\mathbf{s}} + \tilde{\mathbf{e}} = \mathbf{v}$ and $(\tilde{\mathbf{s}}, \tilde{\mathbf{e}}) \in \mathsf{Supp}(D^\ell) \times \mathsf{Supp}(D^k)$.*

When it is clear from the context, we omit the subscript n and q from above for simplicity. The MLWE assumptions are believed to hold even when D is the uniform distribution over ring elements with infinity norm at most B, say $B \approx 5$, for appropriate choices of n, q, k, ℓ [1]. We write $\mathsf{MLWE}_{k,\ell,B}$ when we consider such distribution. For example, the round-2 NIST candidate signature scheme Dilithium [13] uses such parameters. Looking ahead, we will choose our parameters for MSIS and MLWE in accordance with [13].

2.5 Index-hiding Merkle trees

Merkle trees [26] allow to hash a list of elements $A = (a_1, \cdots, a_N)$ into one hash value (often called root). At a later point, one can efficiently prove to a third party that an element a_i was included at a certain position in the list A. In the following, we consider a slight modification of the standard Merkle tree construction, such that one can prove that a single element a_i was included in the tree without revealing its position in the list. Security proofs for the binding

and index-hiding properties of our Merkle tree construction can be found in [5, Sec. 2.6].

Formally, the index-hiding Merkle tree technique consists of three algorithms ($\mathsf{MerkleTree}$, $\mathsf{getMerklePath}$, $\mathsf{ReconstructRoot}$) with access to a common collision-resistant hash function $\mathcal{H}_{\mathsf{Coll}} : \{0,1\}^\star \to \{0,1\}^{2\lambda}$ (with λ being the security parameter):

- $\mathsf{MerkleTree}(A) \to (\mathsf{root}, \mathsf{tree})$: On input a list of 2^k elements $A = (a_1, \cdots, a_{2^k})$, with $k \in \mathbb{N}$, it constructs a binary tree of height k with $\{l_i = \mathcal{H}_{\mathsf{Coll}}(a_i)\}_{i\in[2^k]}$ as its leaves, and where every internal node h, with children h_{left} and h_{right}, equals the hash of a concatenation of its two children. While it is standard to consider the concatenation $h_{\mathsf{left}}||h_{\mathsf{right}}$, for index-hiding Merkle trees we consider a variation which consists in ordering the two children according to the lexicographical order (or any other total order on binary strings). We denote by $(h_{\mathsf{left}}, h_{\mathsf{right}})_{\mathsf{lex}}$ this concatenation. The algorithm then outputs the root root of the Merkle tree, as well as a description of the entire tree tree.
- $\mathsf{getMerklePath}(\mathsf{tree}, i) \to \mathsf{path}$: On input the description of a Merkle tree tree and an index $i \in [2^k]$, it outputs the list path, which contains the sibling of l_i (i.e. a node, different from l_i, that has the same parent as l_i), as well as the sibling of any ancestor of l_i, ordered by decreasing height.
- $\mathsf{ReconstructRoot}(a, \mathsf{path}) \to \mathsf{root}$: On input an element a in the list of elements $A = (a_1, \cdots, a_{2^k})$ and $\mathsf{path} = (n_1, \cdots, n_{k-1})$, it outputs a reconstructed root $\mathsf{root}' = h_k$, which is calculated by putting $h_0 = \mathcal{H}_{\mathsf{Coll}}(a)$ and defining h_i for $i \in [k]$ recursively as $h_i = \mathcal{H}_{\mathsf{Coll}}((h_{i-1}, n_i)_{\mathsf{lex}})$.

2.6 Seed tree

We formalize a primitive called *seed tree*, whose purpose is to first generate a number of pseudorandom values and later disclose an arbitrary subset of them, without revealing information on the remaining values. A seed tree is a complete binary tree[3] of λ-bit seed values such that the left (resp. right) child of a seed seed_h is the left (resp. right) half of $\mathsf{Expand}(\mathsf{seed}||h)$, where Expand is a pseudorandom number generator (PRG). The unique identifier h of the parent seed is appended to separate the input domains of the different calls to the PRG. A sender can efficiently reveal the seed values of a subset of the set of leaves by revealing the appropriate set of internal seeds in the tree. We detail the formal construction of a seed tree below, where $\mathsf{Expand} : \{0,1\}^{\lambda+\lceil\log_2(M-1)\rceil} \to \{0,1\}^{2\lambda}$ is a PRG for any $\lambda, M \in \mathbb{N}$, instantiated by a random oracle $\mathcal{O}$. Then, a seed tree consists of the following four oracle-calling algorithms:

- $\mathsf{SeedTree}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, M) \to \{\mathsf{leaf}_i\}_{i\in[M]}$: On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^\lambda$ and an integer $M \in \mathbb{N}$, it constructs a complete binary tree with M

[3] A *complete* binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.

leaves by recursively expanding each seed to obtain its children seeds. Calls are of the form $\mathcal{O}(\mathsf{Expand}\|\mathsf{seed}\|h)$, where $h \in [M-1]$ is a unique identifier for the position of seed in the binary tree.

- $\mathsf{ReleaseSeeds}^{\mathcal{O}}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}) \to \mathsf{seeds}_{\mathsf{internal}}$: On input a root seed $\mathsf{seed}_{\mathsf{root}} \in \{0,1\}^{\lambda}$, and a challenge $\mathbf{c} \in \{0,1\}^{M}$, it outputs the list of seeds $\mathsf{seeds}_{\mathsf{internal}}$ that covers all the leaves with index i such that $c_i = 1$. Here, we say that a set of nodes F *covers* a set of leaves S if the union of the leaves of the subtrees rooted at each node $v \in F$ is exactly the set S.
- $\mathsf{RecoverLeaves}^{\mathcal{O}}(\mathsf{seeds}_{\mathsf{internal}}, \mathbf{c}) \to \{\overline{\mathsf{leaf}_i}\}_{i \text{ s.t. } c_i=1}$: On input a set $\mathsf{seeds}_{\mathsf{internal}}$ and a challenge $\mathbf{c} \in \{0,1\}^{M}$, it computes and outputs all the leaves of the subtrees rooted at the seeds in $\mathsf{seeds}_{\mathsf{internal}}$.
- $\mathsf{SimulateSeeds}^{\mathcal{O}}(\mathbf{c}) \to \mathsf{seeds}_{\mathsf{internal}}$: On input a challenge $\mathbf{c} \in \{0,1\}^{M}$, it identifies the set of nodes covering the leaves with index i such that $c_i = 1$. It then randomly samples a seed from $\{0,1\}^{\lambda}$ for each of these nodes, and finally outputs the set of these seeds as $\mathsf{seeds}_{\mathsf{internal}}$.

By construction, the leaves $\{\mathsf{leaf}_i\}_{i \text{ s.t. } c_i=1}$ output by $\mathsf{SeedTree}(\mathsf{seed}_{\mathsf{root}}, M)$ are the same as those output by $\mathsf{RecoverLeaves}(\mathsf{ReleaseSeeds}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c}), \mathbf{c})$ for any $\mathbf{c} \in \{0,1\}^{M}$. The last algorithm $\mathsf{SimulateSeeds}$ can be used to argue that the seeds associated with all the leaves with index i such that $c_i = 0$ are indistinguishable from uniformly random values for a recipient that is only given $\mathsf{seeds}_{\mathsf{internal}}$ and $\mathbf{c}$. For a formal proof we refer to [5, Lemma 2.11].

3 From group actions to ring signatures

In this section, our main result consists in showing an efficient OR sigma protocol for group actions. Unlike generic OR sigma protocols, whose proof size grows linearly in N, the proof size of our construction will only grow logarithmically in N. Moreover, the multiplicative overhead in $\log N$ is much smaller (i.e., only the size of a single hash) compared to previous works. To obtain ring signatures, we apply the Fiat-Shamir transform (with aborts) to our OR sigma protocol.

3.1 Admissible group actions

Definition 8 (Admissible group action). *Let G be an additive group, S_1, S_2 two symmetric subsets of G, $\mathcal{X}$ a finite set, δ in $[0,1]$ and $D_{\mathcal{X}}$ a distribution over a set of group actions $\star : G \times \mathcal{X} \to \mathcal{X}$. We say that $\mathsf{AdmGA} = (G, \mathcal{X}, S_1, S_2, D_{\mathcal{X}})$ is a* δ-admissible group action *with respect to $X_0 \in \mathcal{X}$ if the following holds:*

1. *One can efficiently compute $g \star X$ for all $g \in S_1 \cup S_2$ and all $X \in \mathcal{X}$, sample uniformly from S_1, S_2 and $D_{\mathcal{X}}$, and represent elements of G and $\mathcal{X}$ uniquely.*

2. *The intersection of the sets $S_2 + g$, for $g \in S_1$, is sufficiently large. More formally, let $S_3 = \bigcap_{g \in S_1} S_2 + g$, then*

$$|S_3| = \delta\,|S_2|\,.$$

Furthermore, it is efficient to check whether an element $g \in G$ belongs to S_3 and to compute $g \star X$ for all $g \in S_3$, $X \in \mathcal{X}$.

3. *It is difficult to output $g' \in S_2 + S_3$ such that $g' \star X_0 = X$ with non-negligible probability, given $X = g \star X_0$ for some g sampled uniformly from S_1. That is, for any efficient adversary $\mathcal{A}$ we have*

$$\Pr\left[\begin{array}{c|c} g' \in S_2 + S_3, & \star \leftarrow D_{\mathcal{X}} \\ g' \star X_0 = X & g \leftarrow S_1 \\ & X \leftarrow g \star X_0 \\ & g' \leftarrow \mathcal{A}(\star, X) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Hereafter, when the context is clear, we omit the description of the group action $\star$ provided to the adversary and implicitly assume the probabilities are taken over the random choice of $\star$.

3.2 From an admissible group action to base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$.

Before presenting the main OR sigma protocol used for our ring signature, we present an intermediate *base* OR sigma protocol with a binary challenge space. Looking ahead, our main OR sigma protocol will run the base OR sigma protocol several times to amplify the soundness.

Let $\mathsf{AdmGA} = (G, \mathcal{X}, S_1, S_2, D_{\mathcal{X}})$ be an admissible group action with respect to $X_0 \in \mathcal{X}$, and suppose that $X_1 = s_1 \star X_0, \cdots, X_N = s_N \star X_0$ are N public keys, where the corresponding secret keys $s_1, \cdots, s_N$ are drawn uniformly from S_1. In this section, we give an efficient binary-challenge OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}} = (P' = (P'_1, P'_2), V' = (V'_1, V'_2))$ proving knowledge of $(s_I, I) \in S_1 \times [N]$, such that $s_I \star X_0 = X_I$.[4]

We sketch the description of our base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$. First, the prover samples an element r uniformly from S_2, and computes $R_i = r \star X_i$ for all $i \in [N]$. The prover further samples random bit strings $\{\mathsf{bits}_i\}_{i \in [N]}$ uniformly from $\{0,1\}^{\lambda}$, and commits to R_i with the random oracle as $\mathsf{C}_i \leftarrow \mathcal{O}(\mathsf{Com} \| R_i \| \mathsf{bits}_i)$. Then, the prover builds a index-hiding Merkle tree with $\mathsf{C}_1, \cdots, \mathsf{C}_N$ as its leaves.[5] Note that this procedure can be done deterministically, by generating randomness by a pseudorandom number generator (PRG) Expand from a short seed seed. The prover sends the root root of the Merkle tree to the verifier, who responds with a uniformly random bit c.

[4] To be accurate, we prove knowledge of $s_I \in S_2 + S_3$, as we consider "relaxed" special soundness.

[5] For simplicity, we will assume that N is a power of 2. If this is not the case we add additional dummy commitments to make the number of leaves a power of 2.

If the challenge bit c is 0, then the prover computes $z = r + s_I$. If $z \notin S_3$, then the prover aborts (this happens with probability $1 - \delta$). Otherwise the prover sends z, the path in the Merkle tree that connects C_I to the root of the tree and the opening bits bits_I for the commitment C_I. The verifier then computes $\widetilde{R} = z \star X_0$ and $\tilde{\mathsf{C}} := \mathsf{Com}(\tilde{R}, \mathsf{bits}_I)$, and uses the path to reconstruct the root $\widetilde{\mathsf{root}}$ of the index-hiding Merkle tree. They finally check if $z \in S_3$ and $\widetilde{\mathsf{root}} = \mathsf{root}$.

If the challenge bit c is 1 then the prover reveals r to the verifier, as well as the opening bits bits_i for all $i \in [N]$. This allows the verifier to recompute the index-hiding Merkle tree and to check if its root matches the value of root that they received earlier. Note that in this case, it suffices for the prover to just send seed, since r and the bits_i are derived pseudorandomly from this seed.

A toy protocol is displayed in Figure 2 and the full protocol is detailed in Figure 3. In the full protocol, we assume the PRG Expand and the commitment scheme to be instantiated by a random oracle $\mathcal{O}$. We further assume w.l.o.g. that the output length of the random oracle is adjusted appropriately.

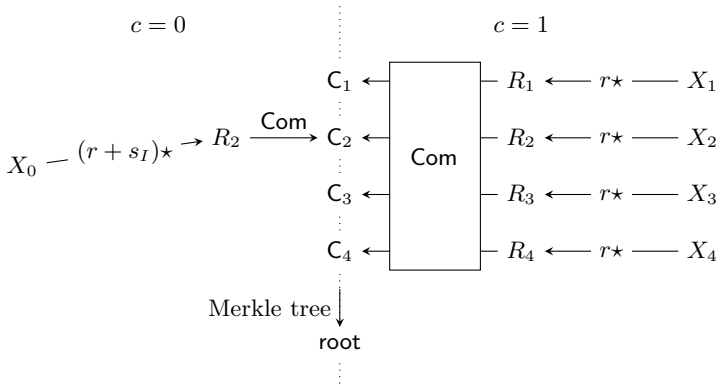

Figure 2. The base sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ to prove knowledge of (s_I, I) such that $s_I \star X_0 = X_I$ (In the drawing $N = 4$ and $I = 2$). If the challenge bit c is 0, then the left side of the picture is revealed, otherwise the right side of the picture is revealed.

3.3 Security proof for the base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$

The following Theorems 9 and 10 provide the security of $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$. For their proofs we refer to [5, Sec. 3.3]

Theorem 9. *Let $\mathcal{O}$ be a random oracle. Define the relation*

$$R = \{((X_1, \cdots, X_N), (s, I)) \,|\, s \in S_1, X_i \in \mathcal{X}, I \in [N], X_I = s \star X_0\}$$

round 1: $P_1'^{\mathcal{O}}((X_1, \cdots, X_N), (s_I, I))$

1: $\mathsf{seed} \leftarrow \{0,1\}^\lambda$ ▷ The only randomness used by the Prover
2: $(r, \mathsf{bits}_1, \cdots, \mathsf{bits}_N) \leftarrow \mathcal{O}(\mathsf{Expand}||\mathsf{seed})$ ▷ Sample $r \in S_2$ and $\mathsf{bits}_i \in \{0,1\}^\lambda$
3: **for** i from 1 to N **do**
4: $\quad R_i \leftarrow r \star X_i$
5: $\quad \mathsf{C}_i \leftarrow \mathcal{O}(\mathsf{Com}||R_i||\mathsf{bits}_i)$ ▷ Create commitment $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
6: $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \cdots, \mathsf{C}_N)$ ▷ Index-hiding Merkle tree
7: Prover sends $\mathsf{com} \leftarrow \mathsf{root}$ to Verifier.

round 2: $V_1'(\mathsf{com})$

1: $c \leftarrow \{0,1\}$
2: Verifier sends $\mathsf{chall} \leftarrow c$ to Prover.

round 3: $P_2'((s_I, I), \mathsf{chall})$

1: $c \leftarrow \mathsf{chall}$
2: **if** $c = 0$ **then**
3: $\quad z \leftarrow r + s_I$
4: $\quad$ **if** $z \notin S_3$ **then**
5: $\quad\quad P$ aborts the protocol.
6: $\quad \mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I)$
7: $\quad \mathsf{rsp} \leftarrow (z, \mathsf{path}, \mathsf{bits}_I)$
8: **else**
9: $\quad \mathsf{rsp} \leftarrow \mathsf{seed}$
10: Prover sends rsp to Verifier

Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{rsp})$

1: $(\mathsf{root}, c) \leftarrow (\mathsf{com}, \mathsf{chall})$
2: **if** $c = 0$ **then**
3: $\quad (z, \mathsf{path}, \mathsf{bits}) \leftarrow \mathsf{rsp}$
4: $\quad \widetilde{R} \leftarrow z \star X_0$
5: $\quad \widetilde{\mathsf{C}} \leftarrow \mathcal{O}(\mathsf{Com}\|\widetilde{R}\|\mathsf{bits})$
6: $\quad \widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
7: $\quad$ Verifier outputs accept if $z \in S_3$ and $\widetilde{\mathsf{root}} = \mathsf{root}$, and otherwise outputs reject
8: **else**
9: $\quad$ Verifier repeats the computation of **round 1** with $\mathsf{seed} \leftarrow \mathsf{rsp}$
10: $\quad$ Verifier outputs accept if the computation results in root, and otherwise outputs reject

Figure 3. Construction of the base OR sigma protocol $\Pi_\Sigma^{\mathsf{RS\text{-}base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$, given an admissible group action $\mathsf{AdmGA} = (G, \mathcal{X}, S_1, S_2, D_\mathcal{X})$ with respect to $X_0 \in \mathcal{X}$ together with a random group action $\star \leftarrow D_\mathcal{X}$. Above, the PRG Expand and the commitment scheme Com are modeled by a random oracle $\mathcal{O}$.

and the relaxed relation

$$\tilde{R} = \left\{ ((X_1, \cdots, X_N), w) \;\middle|\; \begin{array}{c} X_i \in \mathcal{X} \text{ and} \\ w = (s, I) : s \in S_2 + S_3, I \in [N], X_I = s \star X_0 \text{ or} \\ w = (x, x') : \quad x \neq x', \mathcal{H}_{\mathsf{Coll}}(x) = \mathcal{H}_{\mathsf{Coll}}(x') \text{ or} \\ \mathcal{O}(\mathsf{Com}||x) = \mathcal{O}(\mathsf{Com}||x') \end{array} \right\}.$$

Then the OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ of Figure 3 has correctness with probability of aborting $(1-\delta)/2$ and relaxed special soundness for the relations $(R, \tilde{R})$.[6]

Theorem 10. *The OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ of Figure 3 is non-abort honest-verifier zero-knowledge. More concretely, there exists a simulator* Sim *such that for any $(\mathsf{X}, \mathsf{W}) \in R$,* $\mathsf{chall} \in \mathsf{ChSet}$ *and any (computationally unbounded) adversary $\mathcal{A}$ that makes Q queries to the random oracle $\mathcal{O}$, we have*

$$\left| \Pr[\mathcal{A}^{\mathcal{O}}(\widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) \to 1] \right| \leq \frac{2Q}{2^{\lambda}}.$$

Here $\widetilde{P}$ denotes a non-aborting prover $P' = (P'_1, P'_2)$ run on (X, W) with challenge fixed as chall*. In other words,* Sim *simulates to $\mathcal{A}$ the view of an honest non-aborting execution of the sigma protocol without using the witness.*

3.4 From base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ to main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$

To have an OR sigma protocol where a prover cannot cheat with more than negligible probability, we have to enlarge the challenge space. In this section, we show how to obtain our main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$, with a large challenge space, from our base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ with a binary challenge space. Below, we also incorporate three optimization techniques that lead to a much more efficient protocol compared to simply running $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ in parallel λ-times.

Unbalanced challenge space $C_{M,K}$. Notice that in $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$, responding to a challenge with challenge bit $c = 0$ is more costly than responding to the challenge bit $c = 1$ (which requires communicating only a single seed value). Therefore, rather than λ independent executions of $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$, it is more convenient to choose positive integers M, K such that $\binom{M}{K} \geq 2^{\lambda}$ and do $M > \lambda$ executions of the protocol, of which exactly K are chosen to have challenge bit 0. For example, when targeting 128 bits of security, we can do $M = 250$ executions, out of which $K = 30$ correspond to the challenge bit $c = 0$ (so $M - K = 220$ correspond to $c = 1$). Assuming the cost of responding to the challenge $c = 1$ is negligible, this reduces the response size by roughly a factor 2. Moreover, this optimization makes the response size constant and reduces the probability that the prover needs to abort and restart (which allows for better parameter choices). Below, we denote $C_{M,K}$ as the set of strings in $\{0,1\}^M$ such that exactly K-bits are 0.

[6] We note that the notion of collision in $\mathcal{O}$ may seem non-standard at this point since the truth table of $\mathcal{O}$ is typically filled in one at a time when queried so it is not clear who is querying the $\mathcal{O}$ right now. However, we observe that this non-standard notion suffices for our (linkable) ring signature application w.l.o.g.

Using seed tree. Using the unbalanced challenge space, we now run our base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ in parallel M times, and in $(M-K) \approx M$ of the runs, we simply output the random seed sampled by $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$. Here, we use the seed tree (introduced in Section 2.6) to optimize this step. In particular, instead of choosing independent seeds for each of the M instances of $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$, we generate the M seeds using a seed tree. Furthermore, instead of responding with $(M-K)$ seeds, the prover outputs $\mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{ReleaseSeeds}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c})$, where $\mathbf{c}$ is the challenge sampled from $C_{M,K}$. The verifier can then use $\mathsf{seeds}_{\mathsf{internal}}$ along with $\mathbf{c}$ to recover the $(M-K)$ seeds by running $\mathsf{RecoverLeaves}$. This reduces the response length.

Adding salt. As a final tweak to the standard parallel repetition of sigma protocols, the prover P_1 of the main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$ picks a 2λ bit salt and runs the i-th ($i \in [M]$) instance of $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ with the random oracle $\mathcal{O}_i(\cdot) := \mathcal{O}(\mathsf{salt} \| i \| \cdot)$. The prover also salts the seed tree construction. This tweak allows us to prove a tighter security proof for the zero-knowledge property. In practice, this modification does not affect the efficiency of the protocol by much, but it avoids multi-target attacks such as those by Dinur and Nadler [12].

The description of our main OR sigma protocol which incorporates all the above optimizations is depicted in Figure 4.

Remark 11 (Commitment recoverable). Notice that the underlying base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ is *commitment recoverable*. That is, given the statement X, the challenge chall and the response rsp, there is an efficient deterministic algorithm $\mathsf{RecoverCom}(\mathsf{X}, \mathsf{chall}, \mathsf{rsp})$ that recovers the unique commitment com that leads the verifier to accept. This property allows the signer of a Fiat-Shamir type signature to include the challenge rather than the commitment in a signature, which shortens the signature size. Our main sigma protocol is "almost" commitment recoverable, since one can recover the entire commitment except for the random salt. We use this property in Section 3.6.

3.5 Security proof for the main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$

The following Theorems 12 and 13 provide the security of $\Pi_{\Sigma}^{\mathsf{RS}}$. Their proofs can be found in [5, Sec. 3.5]

Theorem 12. *Define the relation R and the relaxed relation $\tilde{R}$ as in Theorem 9. Then the OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$ has correctness with probability of aborting $1-\delta^K$, high min-entropy and relaxed special soundness for the relations $(R, \tilde{R})$.*

Theorem 13. *The OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$ is non-abort special zero-knowledge. More concretely, there exists a simulator Sim such that, for any $(\mathsf{X}, \mathsf{W}) \in R$, $\mathsf{chall} \in \mathsf{ChSet}$ and any (computationally unbounded) adversary $\mathcal{A}$ that makes Q*

round 1: $P_1^{\mathcal{O}}((X_1, \cdots, X_N), (s_I, I))$

1: $\mathsf{seed}_{\mathsf{root}} \leftarrow \{0,1\}^{\lambda}$
2: $\mathsf{salt} \leftarrow \{0,1\}^{2\lambda}$
3: $\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt}||\cdot)$
4: $(\mathsf{seed}_1, \cdots, \mathsf{seed}_M) \leftarrow \mathsf{SeedTree}^{\mathcal{O}'}(\mathsf{seed}_{\mathsf{root}}, M)$
5: **for** i from 1 to M **do**
6: $\quad \mathcal{O}_i(\cdot) := \mathcal{O}(\mathsf{salt}||i||\cdot)$
7: $\quad \mathsf{com}_i \leftarrow P_1'^{\mathcal{O}_i}((X_1, \cdots, X_N), (s_I, I); \mathsf{seed}_i)$ ▷ Run P_1' on randomness seed_i
8: Prover sends $\mathsf{com} \leftarrow (\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M)$ to Verifier.

round 2: $V_1(\mathsf{com})$

1: $\mathbf{c} \leftarrow C_{M,K}$
2: Verifier sends $\mathsf{chall} \leftarrow \mathbf{c}$ to Prover.

round 3: $P_2^{\mathcal{O}}((s_I, I), \mathsf{chall})$

1: $\mathbf{c} = (c_1, \cdots, c_M) \leftarrow \mathsf{chall}$
2: **for** i s.t. $c_i = 0$ **do**
3: $\quad \mathsf{rsp}_i \leftarrow P_2'((s_I, I), c_i; \mathsf{seed}_i)$ ▷ Run P_2' on randomness seed_i
4: $\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt}||\cdot)$
5: $\mathsf{seeds}_{\mathsf{internal}} \leftarrow \mathsf{ReleaseSeeds}^{\mathcal{O}'}(\mathsf{seed}_{\mathsf{root}}, \mathbf{c})$
6: Prover sends $\mathsf{rsp} \leftarrow (\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{rsp}_i\}_{i \text{ s.t. } c_i=0})$ to Verifier

Verification: $V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{rsp})$

1: $((\mathsf{salt}, \mathsf{com}_1, \cdots, \mathsf{com}_M), \mathbf{c} = (c_1, \cdots, c_M)) \leftarrow (\mathsf{com}, \mathsf{chall})$
2: $(\mathsf{seeds}_{\mathsf{internal}}, \{\mathsf{rsp}_i\}_{i \text{ s.t. } c_i=0}) \leftarrow \mathsf{rsp}$
3: $\mathcal{O}'(\cdot) := \mathcal{O}(\mathsf{salt}||\cdot)$
4: $\{\mathsf{resp}_i\}_{i \text{ s.t. } c_i=1} \leftarrow \mathsf{RecoverLeaves}^{\mathcal{O}'}(\mathsf{seeds}_{\mathsf{internal}}, \mathbf{c})$
5: **for** i from 1 to M **do**
6: $\quad \mathcal{O}_i(\cdot) := \mathcal{O}(\mathsf{salt}||i||\cdot)$
7: $\quad$ Verifier outputs reject if $V_2'^{\mathcal{O}_i}(\mathsf{com}_i, c_i, \mathsf{rsp}_i)$ outputs reject
8: Verifier outputs accept

Figure 4. Construction of the main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}} = (P = (P_1, P_2), V = (V_1, V_2))$ based on the base OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$. The challenge space is defined as $C_{M,K} := \{\mathbf{c} \in \{0,1\}^M \mid \|\mathbf{c}\|_1 = M - K\}$. The seed tree and $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$ have access to the random oracle $\mathcal{O}$.

queries of the form $\mathsf{salt}||\cdot$ *to the random oracle* $\mathcal{O}$ *- where* salt *is the salt value included in the transcript returned by* $\widetilde{P}$ *or* Sim*, we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(\widetilde{P}^{\mathcal{O}}(\mathsf{X},\mathsf{W},\mathsf{chall}))\to 1]-\Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{X},\mathsf{chall}))\to 1]\right|\leq\frac{3Q}{2^{\lambda}}\,.$$

Remark 14. We notice that, for the application of (linkable) ring signatures, it suffices to be able to simulate non-aborting transcripts, because an aborting transcript will never be released by the signer.

3.6 From main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$ to ring signature

We apply the Fiat-Shamir transform [17] to our main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$ to obtain a ring signature. The resulting scheme is illustrated in Figure 5, where we also exploit the almost commitment recoverability of $\Pi_{\Sigma}^{\mathsf{RS}}$ (see Remark 11). There, $\mathcal{H}_{\mathsf{FS}}$ is a hash function, with range $C_{M,K}$, modeled as a random oracle. The correctness, anonymity, and unforgeability of the ring signature are a direct consequence of the correctness, high min-entropy, non-abort special zero-knowledge, and (relaxed) special soundness property of the underlying OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$. Since we believe the proofs are folklore (see for example [18, Theorem 4] for some details), we do not provide them (a brief sketch of them can be found in [5, Sec. A.1]).

$\underline{\mathsf{RS.KeyGen}(\mathsf{pp})}$
1: $s \leftarrow S_1$
2: $X \leftarrow s \star X_0$
3: **return** $(\mathsf{vk}=X, \mathsf{sk}=s)$

$\underline{\mathsf{RS.Sign}(\mathsf{sk},\mathsf{M},\mathsf{R})}$
1: $(\mathsf{vk}_1,\cdots\mathsf{vk}_N)\leftarrow\mathsf{R}$ ▷ Let vk_I be associated to $\mathsf{sk}=s_I$.
2: $\mathsf{com}=(\mathsf{salt},(\mathsf{com}_i)_{i\in[M]})\leftarrow P_1^{\mathcal{O}}(\mathsf{R},(\mathsf{sk},I))$
3: $\mathsf{chall}\leftarrow\mathcal{H}_{\mathsf{FS}}(\mathsf{M},\mathsf{R},\mathsf{com})$
4: $\mathsf{rsp}\leftarrow P_2^{\mathcal{O}}((\mathsf{sk},I),\mathsf{chall})$
5: **return** $\sigma=(\mathsf{salt},\mathsf{chall},\mathsf{rsp})$

$\underline{\mathsf{RS.Verify}(\mathsf{R},\mathsf{M},\sigma)}$
1: $(\mathsf{vk}_1,\cdots\mathsf{vk}_N)\leftarrow\mathsf{R}$
2: $(\mathsf{salt},\mathsf{chall},\mathsf{rsp})\leftarrow\sigma$
3: $\mathsf{com}\leftarrow\mathsf{RecoverCom}(\mathsf{R},\mathsf{salt},\mathsf{chall},\mathsf{rsp})$
4: **if** $\mathsf{accept}=V_2^{\mathcal{O}}(\mathsf{com},\mathsf{chall},\mathsf{rsp})\wedge\mathsf{chall}=\mathcal{H}_{\mathsf{FS}}(\mathsf{M},\mathsf{R},\mathsf{com})$ **then**
5: **return** $\top$
6: **else**
7: **return** $\bot$

Figure 5. Ring signature Π_{RS} from our main OR sigma protocol $\Pi_{\Sigma}^{\mathsf{RS}}$ with almost commitment revocability and access to a random oracle $\mathcal{O}$. The setup algorithm $\mathsf{RS.Setup}(1^{\lambda})$ outputs a description of an admissible group action $(G,\mathcal{X},S_1,S_2,D_{\mathcal{X}})$ with respect to a fixed $X_0\in\mathcal{X}$, together with a random group action $\star\leftarrow D_{\mathcal{X}}$ and as the public parameters pp.

4 From a pair of group actions to linkable ring signatures.

In this section we construct a linkable ring signature from a *pair* of group actions, $\star : G \times \mathcal{X} \to \mathcal{X}$ and $\bullet : G \times \mathcal{T} \to \mathcal{T}$, that satisfy certain properties. The proposed linkable ring signature is similar to the ring signature in Section 3. In particular, a secret key is a group element $s \in S_1 \subset G$ and the corresponding public key is $s \star X_0 \in \mathcal{X}$ for a fixed public element X_0. To achieve linkability, in this section the signature contains also a tag $T \in \mathcal{T}$, which is obtained as $T = s \bullet T_0$ for a fixed public element T_0. The signature consists of the tag T, as well as a proof of knowledge of s such that simultaneously $T = s \bullet T_0$ and $s \star X_0$ is a member of the ring of public keys. To check if two signatures are produced by the same party we simply check whether the tags included in the two signatures are "close". Looking ahead, the notion of closeness depends on the underlying algebraic structure used to instantiate the pair of group actions; in the isogeny case, this amounts to checking whether the tags are equal while in the lattice case, this amounts to checking whether the tags are close for the infinity norm.

We require a number of properties from the group actions to make the signature scheme secure. Informally, we need one property per security property of linkable ring signatures (see [5, Sec. 2.3]) :

- **Linkability.** It is hard to find secret keys s and s' such that $s' \star X_0 = s \star X_0$ but $s' \bullet T_0 \not\approx s \bullet T_0$. Otherwise, an adversary can use s and s' to sign two messages under the same public key that do not link together.
- **Linkable anonymity.** For a random secret key s, the distributions $(s \star X_0, s \bullet T_0)$ and $(X, T) \leftarrow \mathcal{X} \times \mathcal{T}$ are indistinguishable. Otherwise, an adversary could link the tag to one of the public keys and break anonymity.
- **Non-Frameability.** Given $X = s \star X_0$ and $T = s \bullet T_0$ it is hard to find s' such that $s' \bullet T_0$ is close to T. Otherwise, an adversary can register $s' \star X_0$ as a public key and frame an honest party with public key $s \star X_0$ for signing a message.

4.1 Admissible pairs of group actions

Definition 15 (Admissible pair of group actions). *Let G be an additive group, S_1, S_2 two symmetric subsets of G, $\mathcal{X}$ and $\mathcal{T}$ two finite sets, δ in $[0, 1]$, and $D_{\mathcal{X}}$ and $D_{\mathcal{T}}$ distributions over a set of group actions $\star : G \times \mathcal{X} \to \mathcal{X}$ and $\bullet : G \times \mathcal{T} \to \mathcal{T}$, respectively. Finally, let $\mathsf{Link}_{\mathsf{GA}} : \mathcal{T} \times \mathcal{T} \to 1/0$ be an associated efficiently computable function. We say that $\mathsf{AdmPGA} = (G, \mathcal{X}, \mathcal{T}, S_1, S_2, D_{\mathcal{X}}, D_{\mathcal{T}}, \mathsf{Link}_{\mathsf{GA}})$ is a* δ-admissible pair of group actions *with respect to $(X_0, T_0) \in \mathcal{X} \times \mathcal{T}$ if the following holds:*

1. *One can efficiently compute $g \star X$, $g \bullet T$ for any $g \in S_1 \cup S_2$ and any $(X, T) \in \mathcal{X} \times \mathcal{T}$, sample uniformly from S_1, S_2, $D_{\mathcal{X}}$ and $D_{\mathcal{T}}$, and represent elements of $G, \mathcal{X}$ and $\mathcal{T}$ uniquely.*

2. *For any* $T \in \mathcal{T}$, $\mathsf{Link}_{\mathsf{GA}}(T, T) = 1$.
3. *The intersection of the sets* $S_2 + g$, *for* $g \in S_1$, *is large. Let* $S_3 = \bigcap_{g \in S_1} S_2 + g$, *then*

$$|S_3| = \delta \, |S_2| \, .$$

Furthermore, it is efficient to check whether an element $g \in G$ *belongs to* S_3, *and to compute* $g \star X$, $g \bullet T$ *for all* $g \in S_3, X \in \mathcal{X}, T \in \mathcal{T}$.
4. *For* g *sampled uniformly from* S_1, $(g \star X_0, g \bullet T_0)$ *is indistinguishable from* (X, T) *sampled uniformly from* $\mathcal{X} \times \mathcal{T}$:

$$\begin{aligned} &\{(\star, \bullet, g \star X_0, g \bullet T_0) \,|\, (\star, \bullet, g) \leftarrow D_{\mathcal{X}} \times D_{\mathcal{T}} \times S_1\} \\ &\qquad \approx_c \{(\star, \bullet, X, T) \,|\, (\star, \bullet, X, T) \leftarrow D_{\mathcal{X}} \times D_{\mathcal{T}} \times \mathcal{X} \times \mathcal{T}\}. \end{aligned}$$

5. *It is difficult to output* $g, g' \in S_2 + S_3$ *such that* $g \star X_0 = g' \star X_0$ *and* $\mathsf{Link}_{\mathsf{GA}}(g' \bullet T_0, g \bullet T_0) = 0$. *That is, for any efficient adversary* $\mathcal{A}$, *the following is negligible:*

$$\Pr\left[\begin{array}{c} g, g' \in S_2 + S_3 \\ g \star X_0 = g' \star X_0 \\ \mathsf{Link}_{\mathsf{GA}}(g \bullet T_0, g' \bullet T_0) = 0 \end{array} \middle| \begin{array}{c} (\star, \bullet) \leftarrow D_{\mathcal{X}} \times D_{\mathcal{T}} \\ (g, g') \leftarrow \mathcal{A}(\star, \bullet) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

6. *It is difficult to output* $g' \in S_2 + S_3$ *such that* $\mathsf{Link}_{\mathsf{GA}}(g' \bullet T_0, T) = 1$ *with non-negligible probability, given* $X = g \star X_0$ *and* $T = g \bullet T_0$ *for some* g *sampled uniformly from* S_1. *That is, for any efficient adversary* $\mathcal{A}$ *we have*

$$\Pr\left[\begin{array}{c} g' \in S_2 + S_3 \\ \mathsf{Link}_{\mathsf{GA}}(g' \bullet T_0, T) = 1 \end{array} \middle| \begin{array}{c} (\star, \bullet, g) \leftarrow D_{\mathcal{X}} \times D_{\mathcal{T}} \times S_1 \\ X \leftarrow g \star X_0 \\ T \leftarrow g \bullet T_0 \\ g' \leftarrow \mathcal{A}(\star, \bullet, X, T) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Hereafter, when the context is clear, we omit the description of the group actions $\star$ *and* $\bullet$ *provided to the adversary and implicitly assume the probabilities are taken over the random choice of the group actions.*

4.2 From an admissible pair of group actions to base OR sigma protocol with tag

As in Section 3, we start by introducing an intermediate *base* OR sigma protocol with tag that has a binary challenge space. The main OR sigma protocol with tag used for our linkable ring signature will run parallel executions of the base OR sigma protocol with tag to amplify the soundness error.

Let $\mathsf{AdmPGA} = (G, \mathcal{X}, \mathcal{T}, S_1, S_2, D_{\mathcal{X}}, D_{\mathcal{T}})$ be a pair of admissible group actions with respect to $(X_0, T_0) \in \mathcal{X} \times \mathcal{T}$, and suppose that $X_1 = s_1 \star X_0, \cdots, X_N = s_N \star X_0$ are N public keys and $T = s_I \bullet T_0$ a tag associated to the I-th user,

where the corresponding secret keys $s_1, \cdots, s_N$ are drawn uniformly from S_1. In this section, we introduce an efficient binary-challenge OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}} = (P' = (P'_1, P'_2), V' = (V'_1, V'_2))$ proving knowledge of $(s_I, I) \in S_1 \times [N]$, such that $s_I \star X_0 = X_I$ and $s_I \bullet T_0 = T$.[7]

We outline the base OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$. First, the prover samples an element r uniformly from S_2, and computes $R_i = r \star X_i$ for all $i \in [N]$ and $T' = r \bullet T$. The prover further samples random bit strings bits_i uniformly from $\{0,1\}^{\lambda}$ for $i \in [N]$ and commits R_i as $\mathcal{O}(\mathsf{Com} \| R_i \| \mathsf{bits}_i)$ (or, equivalently, $\mathsf{Com}(R_i, \mathsf{bits}_i)$). Then, the prover builds a index-hiding Merkle tree with $\mathsf{C}_1, \ldots, \mathsf{C}_N$ as its leaves and hashes the root root of the Merkle tree obtaining T' as $h = \mathcal{H}_{\mathsf{Coll}}(T', \mathsf{root})$. Here, the only reason for hashing (T', root) is to lower the communication complexity and it has no impact on the security. Moreover, we note that this whole procedure can be done deterministically, with randomness generated from a seed seed. Finally, the prover sends the hash value h to the verifier, who responds with a uniformly random bit c.

If the challenge bit c is 0, then the prover computes $z = r + s_I$. If $z \notin S_3$, then the prover aborts (this happens with probability $1 - \delta$). Otherwise, the prover sends z, the opening bits bits_I for the commitment C_I, and the path in the index-hiding Merkle tree that connects C_I to the root of the tree. The verifier then computes $\widetilde{R} = z \star X_0$, $\widetilde{T} = z \bullet T_0$ and $\widetilde{\mathsf{C}} = \mathsf{Com}(\widetilde{R}, \mathsf{bits}_I)$, and uses the path to reconstruct the root $\widetilde{\mathsf{root}}$ of the Merkle tree. It finally accepts if and only if h is equal to $\mathcal{H}_{\mathsf{Coll}}(\widetilde{T}, \widetilde{\mathsf{root}})$. If the challenge bit c is 1 then the prover reveals r and the bits_i, for all $i \in [N]$, to the verifier. This allows the verifier to recompute the Merkle tree and $T' = r \bullet T$, and to check if the hash of T' and the obtained root matches the value h received earlier. In this case, it suffices for the prover to just send seed, since r and bits_i are derived pseudorandomly from it. In the full protocol, displayed in Figure 6, we assume the PRG Expand and the commitment scheme Com to be instantiated by a random oracle $\mathcal{O}$. We further assume w.l.o.g. that the output length of the random oracle is adjusted appropriately.

The following Theorems 16 and 17 provide the security of $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$. Their proofs can be found in [5, Sec. A.2].

Theorem 16. *Let $\mathcal{O}$ be a random oracle. Define the relation*

$$R = \{((X_1, \cdots, X_N, T), (s, I)) \mid s \in S_1, X_i \in \mathcal{X}, T \in \mathcal{T}, I \in [N], X_I = s \star X_0, T = s \bullet T_0\}$$

and the relaxed relation

$$\tilde{R} = \left\{ ((X_1, \cdots, X_N, T), w) \;\middle|\; \begin{array}{c} X_i \in \mathcal{X}, T \in \mathcal{T} \text{ and } w \text{ such that :} \\ w = (s, I) : s \in S_2 + S_3, I \in [N], X_I = s \star X_0, T = s \bullet T_0 \\ \text{or} \\ w = (x, x') : \quad x \neq x', \mathcal{H}_{\mathsf{Coll}}(x) = \mathcal{H}_{\mathsf{Coll}}(x') \text{ or} \\ \mathcal{O}(\mathsf{Com}||x) = \mathcal{O}(\mathsf{Com}||x') \end{array} \right\}.$$

[7] To be accurate, we prove knowledge of $s_I \in S_2 + S_3$, as we consider "relaxed" special soundness.

round 1: $P_1'^{\mathcal{O}}((X_1, \cdots, X_N, T), (s_I, I))$

1: $\mathsf{seed} \leftarrow \{0,1\}^\lambda$ ▷ The only randomness used by the Prover
2: $(r, \mathsf{bits}_1, \cdots, \mathsf{bits}_N) \leftarrow \mathcal{O}(\mathsf{Expand} \| \mathsf{seed})$ ▷ Sample $r \in S_2$ and $\mathsf{bits}_i \in \{0,1\}^\lambda$.
3: $T' \leftarrow r \bullet T$
4: **for** i from 1 to N **do**
5: $\quad R_i \leftarrow r \star X_i$
6: $\quad \mathsf{C}_i \leftarrow \mathcal{O}(\mathsf{Com} \| R_i \| \mathsf{bits}_i)$ ▷ Create commitment $\mathsf{C}_i \in \{0,1\}^{2\lambda}$
7: $(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(\mathsf{C}_1, \cdots, \mathsf{C}_N)$ ▷ Index-hiding Merkle tree
8: $h \leftarrow \mathcal{H}_{\mathsf{Coll}}(T', \mathsf{root})$
9: Prover sends $\mathsf{com} \leftarrow h$ to Verifier.

round 2: $V_1'(\mathsf{com})$

1: $c \leftarrow \{0,1\}$
2: Verifier sends $\mathsf{chall} \leftarrow c$ to Prover.

round 3: $P_2'((s_I, I), \mathsf{chall})$

1: $c \leftarrow \mathsf{chall}$
2: **if** $c = 0$ **then**
3: $\quad z \leftarrow r + s_I$
4: $\quad$ **if** $z \notin S_3$ **then**
5: $\quad\quad P$ aborts the protocol.
6: $\quad \mathsf{path} \leftarrow \mathsf{getMerklePath}(I, \mathsf{tree})$
7: $\quad \mathsf{rsp} \leftarrow (z, \mathsf{path}, \mathsf{bits}_I)$
8: **else**
9: $\quad \mathsf{rsp} \leftarrow \mathsf{seed}$
10: Prover sends rsp to Verifier

Verification: $V_2'^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{rsp})$

1: $(h, c) \leftarrow (\mathsf{com}, \mathsf{chall})$
2: **if** $c = 0$ **then**
3: $\quad (z, \mathsf{path}, \mathsf{bits}) \leftarrow \mathsf{rsp}$
4: $\quad \widetilde{R} \leftarrow z \star X_0$
5: $\quad \widetilde{\mathsf{C}} = \mathcal{O}(\mathsf{Com} \| \widetilde{R} \| \mathsf{bits})$
6: $\quad \widetilde{T} \leftarrow z \bullet T_0$
7: $\quad \widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\widetilde{\mathsf{C}}, \mathsf{path})$
8: $\quad$ Verifier outputs accept if $z \in S_3$ and $\mathcal{H}_{\mathsf{Coll}}(\widetilde{T}, \widetilde{\mathsf{root}}) = h$, and otherwise outputs reject.
9: **else**
10: $\quad$ Verifier repeats the computation of **round 1** with rsp as seed.
11: $\quad$ Verifier outputs accept iff the computation results in h, and otherwise outputs reject.

Figure 6. Construction of the base OR sigma protocol with tag $\Pi_\Sigma^{\mathsf{LRS\text{-}base}} = (P' = (P_1', P_2'), V' = (V_1', V_2'))$, given an admissible pair of group actions $\mathsf{AdmPGA} = (G, \mathcal{X}, \mathcal{T}, S_1, S_2, D_\mathcal{X}, D_\mathcal{T}, \mathsf{Link}_{\mathsf{GA}})$ with respect to $(X_0, T_0) \in \mathcal{X} \times \mathcal{T}$, together with random group actions $(\star, \bullet) \leftarrow D_\mathcal{X} \times D_\mathcal{T}$. Above, the PRG Expand and the commitment scheme Com are modeled by a random oracle $\mathcal{O}$.

Then the base OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$ *of Figure 6 has correctness with probability of aborting* $(1-\delta)/2$ *and relaxed special soundness for the relations* $(R, \tilde{R})$.

Theorem 17. *The OR sigma protocol with tag* $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$ *of Figure 6 is non-abort honest-verifier zero-knowledge. More concretely, there exists a simulator* Sim *such that, for any* $(\mathsf{X}, \mathsf{W}) \in R$, $\mathsf{chall} \in \mathsf{ChSet}$ *and any (computationally unbounded) adversary* $\mathcal{A}$ *that makes* Q *queries to the random oracle* $\mathcal{O}$*, we have*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(\widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) \to 1]\right| \leq \frac{2Q}{2^{\lambda}}.$$

4.3 From base OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$ to main OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS}}$

As in Section 3.4, we enlarge the challenge space of our base OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$ to obtain our main OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS}}$. We also include the same optimization techniques presented in Section 3.4. Since the description of our main OR sigma protocol with tag is almost identical to the one depicted in Figure 4, we omit the details. The only notable difference between $\Pi_{\Sigma}^{\mathsf{RS}}$ from Figure 4 and $\Pi_{\Sigma}^{\mathsf{LRS}}$ is that in the latter, the statement additionally includes a tag T and runs $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$ as a subroutine instead of $\Pi_{\Sigma}^{\mathsf{RS\text{-}base}}$. Otherwise, the way we transform our base to our main OR sigma protocol is identical. We also note that it is easy to check that our $\Pi_{\Sigma}^{\mathsf{LRS\text{-}base}}$ enjoys almost commitment revocability (see Remark 11). We use this fact when constructing a linkable ring signature in Section 4.4.

The following Theorems 18 and 19 provide the security of $\Pi_{\Sigma}^{\mathsf{LRS}}$. We refer to [5, Sec. A.3] for their proofs.

Theorem 18. *Define the relation* R *and the relaxed relation* $\tilde{R}$ *as in Theorem 16. Then the OR sigma protocol with tag* $\Pi_{\Sigma}^{\mathsf{LRS}}$ *has correctness with probability of aborting* $1-\delta^K$*, high min-entropy and relaxed special soundness for the relations* $(R, \tilde{R})$.

Theorem 19. *The OR sigma protocol with tag* $\Pi_{\Sigma}^{\mathsf{LRS}}$ *is non-abort special zero-knowledge. More concretely, there exists a simulator* Sim *such that, for any* $(\mathsf{X}, \mathsf{W}) \in R$, $\mathsf{chall} \in \mathsf{ChSet}$ *and any (computationally unbounded) adversary* $\mathcal{A}$ *that makes* Q *queries of the form* $\mathsf{salt}||\cdot$ *to the random oracle* $\mathcal{O}$ *- where* salt *is the salt value included in the transcript returned by* $\widetilde{P}$ *or* Sim*, we have:*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(\widetilde{P}^{\mathcal{O}}(\mathsf{X}, \mathsf{W}, \mathsf{chall})) \to 1] - \Pr[\mathcal{A}^{\mathcal{O}}(\mathsf{Sim}^{\mathcal{O}}(\mathsf{X}, \mathsf{chall})) \to 1]\right| \leq \frac{3Q}{2^{\lambda}}.$$

4.4 From main OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS}}$ to linkable ring signatures

We apply the Fiat-Shamir transform [17] to our main OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS}}$ to obtain a linkable ring signature Π_{LRS}. This is illustrated in Figure 7,

where we also rely on the almost commitment recoverable property of $\Pi_{\Sigma}^{\mathsf{LRS}}$ (see Remark 11). Here, $\mathcal{H}_{\mathsf{FS}}$ is a hash function, with range $C_{M,K}$, modeled as a random oracle. The correctness and security of Π_{LRS} are provided in the following theorem, whose proof can be found in [5, Sec. A.4] .

$\underline{\mathsf{LRS.KeyGen}(\mathsf{pp})}$
1: $s \leftarrow S_1$
2: $X := s \star X_0$
3: **return** $(\mathsf{vk} = X, \mathsf{sk} = s)$

$\underline{\mathsf{LRS.Link}(\sigma_0, \sigma_1)}$
1: $(\mathsf{salt}_b, T_b, \mathsf{chall}_b, \mathsf{rsp}_b) \leftarrow \sigma_b$ for $b \in \{0,1\}$
2: **if** $1 \leftarrow \mathsf{Link}_{\mathsf{GA}}(T_0, T_1)$ **then**
3: **return** $\top$
4: **else**
5: **return** $\bot$

$\underline{\mathsf{LRS.Sign}(\mathsf{sk}, \mathsf{M}, \mathsf{R})}$
1: $(\mathsf{vk}_1, \cdots \mathsf{vk}_N) \leftarrow \mathsf{R}$ ▷ Let vk_I be associated to $\mathsf{sk} = s_I$.
2: $T := s_I \bullet T_0$
3: $\mathsf{com} = (\mathsf{salt}, (\mathsf{com}_i)_{i\in[M]}) \leftarrow P_1^{\mathcal{O}}((\mathsf{R}, T), (\mathsf{sk}, I))$
4: $\mathsf{chall} \leftarrow \mathcal{H}_{\mathsf{FS}}(\mathsf{M}, (\mathsf{R}, T), \mathsf{com})$
5: $\mathsf{rsp} \leftarrow P_2^{\mathcal{O}}((\mathsf{sk}, I), \mathsf{chall})$
6: **return** $\sigma = (\mathsf{salt}, T, \mathsf{chall}, \mathsf{rsp})$

$\underline{\mathsf{LRS.Verify}(\mathsf{R}, \mathsf{M}, \sigma)}$
1: $(\mathsf{vk}_1, \cdots \mathsf{vk}_N) \leftarrow \mathsf{R}$
2: $(\mathsf{salt}, T, \mathsf{chall}, \mathsf{rsp}) \leftarrow \sigma$
3: $\mathsf{com} \leftarrow \mathsf{RecoverCom}((\mathsf{R}, T), \mathsf{salt}, \mathsf{chall}, \mathsf{rsp})$
4: **if** $\mathsf{accept} = V_2^{\mathcal{O}}(\mathsf{com}, \mathsf{chall}, \mathsf{rsp}) \wedge \mathsf{chall} = \mathcal{H}_{\mathsf{FS}}(\mathsf{M}, (\mathsf{R}, T), \mathsf{com})$ **then**
5: **return** $\top$
6: **else**
7: **return** $\bot$

Figure 7. Linkable ring signature Π_{LRS} from our main OR sigma protocol with tag $\Pi_{\Sigma}^{\mathsf{LRS}}$, with almost commitment revocability and access to a random oracle $\mathcal{O}$. The setup algorithm $\mathsf{LRS.Setup}(1^{\lambda})$ outputs a description of a pair of admissible group actions $(G, \mathcal{X}, S_1, S_2, D_{\mathcal{X}}, D_{\mathcal{T}})$ with respect to a fixed $(X_0, T_0) \in \mathcal{X} \times \mathcal{T}$, together with random group actions $(\star, \bullet) \leftarrow D_{\mathcal{X}} \times D_{\mathcal{T}}$ as the public parameters pp.

Theorem 20. *Assuming that* AdmPGA *is an admissible pair of group actions (Definition 15) and $\mathcal{H}_{\mathsf{FS}}$ is a collision-resistant hash function, then the linkable ring signature scheme Π_{LRS} in Figure 7 is correct, linkable, linkable anonymous and non-frameable in the random oracle model.*

5 Post-quantum admissible (pair of) group actions from isogeny and lattice assumptions

For concrete instantiations of our generic framework for ring signatures (Section 3) and linkable ring signatures (Section 4), we consider three admissible (pairs of) group actions, based on isogenies between elliptic curves and lattices.

5.1 Isogeny-based instantiations

The isogeny-based instantiations we propose exploit the CSIDH paradigm. For the three sets of CSIDH parameters that have been proposed so far - CSIDH-512, CSIDH-1024 and CSIDH-1792 ([8, 10]) - the structure of the corresponding

ideal class group $\mathcal{C}\ell(\mathcal{O})$ is only known for the first set [6]. We can instantiate our RS and LRS with any CSIDH parameter set regardless of whether the class group is known or not, but the resulting schemes are much more efficient in the former case. We first discuss the case when the structure of $\mathcal{C}\ell(\mathcal{O})$ is known.

Known class group For simplicity, we assume that the ideal class group $\mathcal{C}\ell(\mathcal{O})$ is cyclic with generator $\mathfrak{g}$ of order cl. Then, the group $\mathbb{Z}_{cl}$ acts freely and transitively on $\mathcal{E}\ell\ell_p(\mathcal{O},\pi)$ via the group action $\star$ defined as $a \star X := \mathfrak{g}^a * X$ (see Section 2.3). In practice, the action of each $a \in \mathbb{Z}_{cl}$ can be computed efficiently when p has a suitable form (in that case the approximate closest vector problem can be solved efficiently in the relation lattice [6]). It can be verified (see [5, Thm. 5.1]) that this group action satisfies all the properties of an admissible group action assuming the hardness of the GAIP_p problem. In this case we have $S_1 = S_2 = S_3 = G$, so $\delta = 1$ and the signing algorithm will never need to abort. Moreover, if we define $\star^2$ to be the group action of $\mathbb{Z}_{cl}$ on $\mathcal{E}\ell\ell_p(\mathcal{O},\pi)$ defined by $a \star^2 X := (2a) \star X$, then $(\star, \star^2)$ satisfies all the properties of an admissible pair of group actions, assuming the hardness of the GAIP_p and $\mathsf{sdCSIDH}_p$ problem ([5, Thm. 5.1]).

Unknown class group. When the structure of the ideal class group $\mathcal{O}$ is not known, computing the action $[\mathfrak{a}] * [E_0]$ of an arbitrary $[\mathfrak{a}] \in \mathcal{C}\ell(\mathcal{O})$ on some $[E_0] \in \mathcal{E}\ell\ell_p(\mathcal{O},\pi)$ has exponential complexity. However, the ideal class action $*$ can still be efficiently computed for a small set of class group elements [8]. In particular, considering p of the form $4\ell_1\ell_2\cdots\ell_k - 1$, with $\ell_1, \ldots, \ell_k$ small odd primes, a special fractional ideal $\mathfrak{I}_{\ell_i}$ can be associated to each prime ℓ_i. The action of one of these ideals (and their inverses) can be computed very efficiently, since it is determined by an isogeny whose kernel is the unique subgroup of $E_0(\mathbb{F}_p)$ of order ℓ_i. We can thus efficiently compute the action of elements in $\mathcal{C}\ell(\mathcal{O})$ of the form $\prod_{i=1}^{k}[\mathfrak{I}_{\ell_i}]^{e_i}$ when the integral exponents e_i are chosen from some small interval $[-B, B]$.

We denote by $\star$ the group action of $\mathbb{Z}^k$ on $\mathcal{E}\ell\ell_p(\mathcal{O},\pi)$ defined by

$$((e_1, \ldots, e_k), X) \mapsto \prod_{i=1}^{k}[\mathfrak{I}_{\ell_i}]^{e_i} * X\,.$$

Then it can be verified that, for the sets $S_1 = [-B, B]^k$ and $S_2 = [-B', B']^k$ (with $B' > B$) the group action $\star$ satisfies all the properties of an admissible group action with $\delta = ((2(B' - B) + 1)/(2B' + 1))^k$, assuming the hardness of the GAIP_p problem (see [5, Thm. 5.4]) . We note that, for a fixed value of B, the bigger the value of B', the bigger δ, and the smaller the aborting probability of the ring signature scheme. However, a big B' implies high computational costs for the action of elements in S_2 and S_3. Consequently, in concrete instantiations the value of B' must be tuned to balance the two effects. Moreover, if we define

$\star^2$ similarly as before, then $(\star, \star^2)$ satisfies all the properties of an admissible pair of group actions, assuming the hardness of the GAIP_p and $\mathsf{sdCSIDH}_p$ problem ([5, Thm. 5.4]).

Remark 21. To avoid using the sdCSIDH hardness assumption, we can formulate an admissible pair of group actions differently. If, considering $\bullet = \star$, we can determine a uniformly random base point T_0 for the tag space such that the element $g \in G$ satisfying $T_0 = g \star X_0$ is unknown to any user, instead of the sdCSIDH hardness assumption we then only require the standard dCSIDH assumption. The drawback is that we require a trusted setup to choose such a T_0. Alternatively, we can look at this as a linkable group signature scheme where the group manager sets $T_0 = t \star X_0$ and remembers t. The group manager can deanonymize any signature because $(-t) \star T$ is the public key of the signer.

Remark 22. Recently, a variant of CSIDH, called CSURF, has been proposed [7]. This work considers the maximal order $\mathcal{O}_{\mathbb{K}}$ and the corresponding set of supersingular elliptic curves $\mathcal{E}\ell\ell_p(\mathcal{O}_{\mathbb{K}}, \pi)$. The action of $\mathcal{C}\ell(\mathcal{O}_{\mathbb{K}})$ on $\mathcal{E}\ell\ell_p(\mathcal{O}_{\mathbb{K}}, \pi)$ can be used in our framework instead of the CSIDH group action.

5.2 Lattice-based instantiation

We instantiate an admissible group action (AdmGA) and an admissible pair of group actions (AdmPGA) based on lattices under the MSIS and MLWE assumptions. For the AdmGA, we consider $(G, \mathcal{X})$ to be $(R_q^\ell \times R_q^\ell, R_q^k)$ and $S_b := \{(\mathbf{s}, \mathbf{e}) \in G \mid \|\mathbf{s}\|_\infty, \|\mathbf{e}\|_\infty \leq B_b\}$ for $b \in \{1, 2\}$, where $B_1 < B_2 < q$ are given positive integers. Then, the group action $\star_{\mathbf{A}}$, uniquely defined by a matrix $\mathbf{A} \in R_q^{k \times \ell}$, is defined as $(\mathbf{s}, \mathbf{e}) \star_{\mathbf{A}} \mathbf{w} := (\mathbf{A}\mathbf{s} + \mathbf{e}) + \mathbf{w}$, for any $\mathbf{w}$ in R_q^k.

We can similarly instantiate the AdmPGA, with the only difference that we have to take care of the tag. To this end, we define $G = R_q^\ell \times R_q^\ell \times R_q^\ell$ and extend S_1, S_2 accordingly, in order to be subsets of G. Then, the group actions $\star_{\mathbf{A}}, \bullet_{\mathbf{B}}$ (where $\mathbf{B} \in R_q^{k \times \ell}$) are defined as $(\mathbf{s}, \mathbf{e}, \tilde{\mathbf{e}}) \star_{\mathbf{A}} \mathbf{w} := (\mathbf{A}\mathbf{s} + \mathbf{e}) + \mathbf{w}$ and $(\mathbf{s}, \mathbf{e}, \tilde{\mathbf{e}}) \star_{\mathbf{B}} \mathbf{w} := (\mathbf{B}\mathbf{s} + \tilde{\mathbf{e}}) + \mathbf{w}$, for any $\mathbf{w}$ in R_q^k. Finally, for two tags $\mathbf{v}, \mathbf{v}'$, we define $\mathsf{Link}_{\mathsf{GA}}(\mathbf{v}, \mathbf{v}') = 1$ if and only if $\|\mathbf{v} - \mathbf{v}'\|_\infty \leq 2 \cdot (2B_2 - B_1)$. It is an easy calculation checking that our instantiations satisfy the required properties of an AdmGA and AdmPGA, assuming the MSIS and MLWE assumptions (with appropriate parameters). For a formal treatment we refer to [5, Sec. 5.2].

Further optimization using Bai-Galbraith [2]. Although we can no longer capture it by our generic construction from admissible (pair of) group actions, we can apply the simple optimization technique of Bai-Galbraith [2], which uses the specific algebraic structure of lattices, to our base OR sigma protocols in Figures 3 and 6. Effectively, this allows to lower the signature size of our lattice-based (linkable) ring signature scheme with no additional cost. The main observation is that for MLWE, proving knowledge of a short $\mathbf{s} \in R_q^\ell$ indirectly proves knowledge of a short $\mathbf{e} \in R_q^k$ since $\mathbf{e}$ is uniquely defined as $\mathbf{v} - \mathbf{A}\mathbf{s}$. We incorporate

this idea to our base OR sigma protocol by letting the prover only send a short vector $\mathbf{z}$ in R_q^ℓ rather than a short vector $\mathbf{z}$ in $R_q^k \times R_q^\ell$ (for ring signatures) or $\mathbf{z}$ in $R_q^k \times R_q^\ell \times R_q^\ell$ (for linkable ring signatures) as the response. Since $k \approx \ell$, this shortens the response without any actual cost. We believe this optimization is standard by now as it is used by most of the recent proposals for efficient lattice-based signature schemes. Therefore we refer to [5, Appendix B] for the full details. In terms of security, the only difference is that the extracted witness from the base OR sigma protocol will be slightly larger than before. Otherwise, all our proofs in Sections 3 and 4 are unmodified by this optimization.

6 Parameter selection, implementation results and conclusions

We implemented the isogeny-based instantiations with known class group and the lattice-based instantiations of our ring signature schemes (standard and linkable). We reuse parameter sets from the pre-existing cryptosystems CSI-FiSh and Dilithium. This allows us to reuse large portions of code from the CSIDH/CSI-FiSh and Dilithium implementations and to rely on earlier work to estimate the concrete security of our parameter choices. We use 128-bit seeds and commitment randomness, and we use 256-bit salts, commitments, and hash values.

Isogeny parameters. We use the CSIDH-512 prime p, and define our first group action $g \star X$ exactly as in CSI-FiSh. This parameter set was proposed to achieve NIST security level 1. State of the art analysis of this parameter set suggests that it provides 128 bits of classical security and about 60 bits of security against quantum adversaries [27]. We set $M = 247$ and $K = 30$ such that the challenge space consists of binary strings of length $M = 247$ with hamming weight $M - K = 217$. The number of these strings is $\binom{247}{30} \approx 2^{128.1}$.

Lattice parameters. We use the "medium" parameter set from the NIST PQC candidate Dilithium. More concretely we use the ring $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$, where $q = 8380417$. The parameters of the MLWE problem are $(k, l) = (3, 4)$ and the coefficients of the LWE secrets are sampled uniformly from $[-6, 6]$. In our implementation we use the optimization by Bai and Galbraith [2]. We chop off $d = 20$ bits of the commitment vector, in such a way that the parameters of the MSIS problem match the parameters of the MSIS problem relevant for the security of the Dilithium scheme. Since we work with binary challenges, the probability that a single rejection sampling check fails is much lower compared to Dilithium. This effect is roughly canceled out by the fact that in our protocol we need a number of parallel checks to succeed all at the same time. The Dilithium "medium" parameters are believed to achieve NIST security level I. Since the lattice signatures are fast, we can afford to have a large number of iterations with a small number of $c = 0$ challenges. This trades signing and verification speed for smaller signatures. Concretely, we set $M = 1749$ and $K = 16$.

6.1 Implementation

For the isogeny-based instantiations we reuse the non-constant-time implementation of the group action CSI-FiSh, which in turn relies on the implementation of the CSIDH group action by Castryck et al. [6, 8]. For the lattice-based instantiations we reuse code of the Dilithium NIST submission for arithmetic and packing/unpacking operations. For both instantiations we use cSHAKE to instantiate the random oracles [21]. In the isogeny-based implementation, the performance bottleneck is the evaluation of the CSIDH group action. In the lattice-based implementation the bottleneck is not the lattice arithmetic, but rather the use of symmetric primitives (i.e. hashing, commitments and expanding seeds). This is especially true in the case of large ring sizes since the number of multiplications in R_q is independent of the ring size. The signature sizes and signing times of our implementations are displayed in Figure 1. Our implementation is publicly available on

`https://github.com/WardBeullens/Calamari-and-Falafl` .

6.2 Conclusions

So far, no *efficient* logarithmic ring signatures have been proven secure in the quantum random oracle model, since the usual multiple rewinding of the adversary in the unforgeability proof is non-trivial in the quantum setting. It remains an interesting open problem to provide security proofs of our schemes in the QROM.

In terms of practical efficiency, we believe the lattice-based implementation can be speed up significantly by using more efficient symmetric primitives and/or by using vectorized implementations. Concerning the isogeny case, we note that, using the larger CSIDH parameters CSIDH-1024 and CSIDH-1792 under the hypothesis that the structure of the ideal class group was know also for them, the signatures sizes would increase with respect to the CSIDH-512 parameters of 0.9 KB or 2.3 KB respectively, independently of the ring size N. This shows that the impact of the CSIDH parameters on the signature size is not dramatic, especially for large N.

References

[1] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 351–367. Springer, Heidelberg, September 2018.

[2] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, Heidelberg, February 2014.

[3] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of Diffie-Hellman problem. In Sihan Qing, Dieter Gollmann, and Jianying Zhou, editors, *ICICS 03*, volume 2836 of *LNCS*, pages 301–312. Springer, Heidelberg, October 2003.

[4] Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. In David Naccache, Shouhuai Xu, Sihan Qing, Pierangela Samarati, Gregory Blanc, Rongxing Lu, Zonghua Zhang, and Ahmed Meddahi, editors, *ICICS 18*, volume 11149 of *LNCS*, pages 303–322. Springer, Heidelberg, October 2018.

[5] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (Linkable) Ring Signatures from Isogenies and Lattices. In *Cryptology ePrint Archive, Report 2020/646, https://eprint.iacr.org/2020/646*, 2020.

[6] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

[7] Wouter Castryck and Thomas Decru. CSIDH on the surface. In PQCRYPTO 2020 [28], pages 111–129.

[8] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Heidelberg, December 2018.

[9] Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In PQCRYPTO 2020 [28], pages 169–186.

[10] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Ishai and Rijmen [19], pages 759–789.

[11] David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, pages 419–440. Springer, Heidelberg, 2018.

[12] Itai Dinur and Niv Nadler. Multi-target attacks on the Picnic signature scheme and related protocols. In Ishai and Rijmen [19], pages 699–727.

[13] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018. `https://tches.iacr.org/index.php/TCHES/article/view/839`.

[14] Ali El Kaafarani, Shuichi Katsumata, and Federico Pintore. Lossy CSI-FiSh: Efficient signature scheme with tight reduction to decisional CSIDH-512. In *PKC 2020, Part II*, LNCS, pages 157–186. Springer, Heidelberg, 2020.

[15] Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 115–146. Springer, Heidelberg, August 2019.

[16] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019.

[17] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[18] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 253–280. Springer, Heidelberg, April 2015.

[19] Yuval Ishai and Vincent Rijmen, editors. *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*. Springer, Heidelberg, May 2019.

[20] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[21] John Kelsey, Shu-jen Chang, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash, and ParallelHash. Technical report, National Institute of Standards and Technology, 2016.

[22] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[23] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 1–31. Springer, Heidelberg, May 2016.

[24] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 110–130. Springer, Heidelberg, June 2019.

[25] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, December 2009.

[26] Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *CRYPTO'87*, volume 293 of *LNCS*, pages 369–378. Springer, Heidelberg, August 1988.

[27] Chris Peikert. He gives C-sieves on the CSIDH. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, LNCS, pages 463–492. Springer, Heidelberg, May 2020.

[28] *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*. Springer, Heidelberg, 2020.

[29] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001.

[30] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[31] Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice RingCT v1.0). In Willy Susilo and Guomin Yang, editors, *ACISP 18*, volume 10946 of *LNCS*, pages 558–576. Springer, Heidelberg, July 2018.