TRIFORS: LINKable Trilinear Forms Ring Signature

Giuseppe D'Alconzo[1] and Andrea Gangemi[1]

[1] *Department of Mathematical Sciences, Politecnico di Torino*

Abstract

We present TRIFORS (TRIlinear FOrms Ring Signature), a logarithmic post-quantum (linkable) ring signature based on a novel assumption regarding equivalence of alternating trilinear forms. The basis of this work is the construction by Beullens, Katsumata and Pintore from 2020 to obtain a linkable ring signature from a cryptographic group action. The group action on trilinear forms used here is the same employed in the signature presented by Tang et al. at EUROCRYPT 22. We first define a sigma protocol that, given a set of public keys, the *ring*, allows to prove the knowledge of a secret key corresponding to a public one in the ring. Furthermore, some optimisations are used to reduce the size of the signature: among others, we use a novel application of the combinatorial number system to the space of the challenges. Using the Fiat-Shamir transform, we obtain a (linkable) ring signature of competitive length with the state-of-the-art among post-quantum proposals for security levels 128 and 192.

Keywords— Tensor Isomorphism, Alternating Trilinear Forms, Ring Signatures, Linkable Ring Signatures

1 Introduction

Ring signatures. Ring signatures were introduced in 2001 by Rivest, Shamir and Tauman [26]. They are a simplified variant of *group signature schemes* [8] and are useful when the involved members do not want to cooperate. A key element in these schemes is the *ring*: a set of R public keys which belong to different users. The signature is then produced by a single user, exploiting all the R public keys of the ring. Ring signatures must satisfy two key properties, *anonymity* and *unforgeability*. Shortly, the former means that the verifier should not be able to identify the signer of a transaction in a ring of size R with a probability greater than $\frac{1}{R}$, while the latter tells us that in order to produce a valid signature, it is necessary to know at least one secret key associated to one of the R public keys of the ring. In a ring signature, after the key generation phase, where each user will receive a secret/public key pair $(\mathsf{sk}, \mathsf{pk})$, the signer I will produce the signature σ starting from a message msg, the ring containing the R public keys and his secret key, sk_I. A verifier will then be able to check the correctness, knowing only the message msg, the ring and the signature σ. Moreover, a ring signature can also be *linkable*. In this case, an additional value τ will be produced during the sign phase. This value will not change if it is computed starting from the same secret key, so if the same user produces two different signatures, they will be linked. Formal definitions about ring signature properties can be found in Section 2.

Several ring signatures were proposed in these years. [26] described two different protocols based on the RSA and Rabin assumption, while Liu, Wei and Wong [18] introduced in 2004 a new group signature algorithm known as *Linkable Spontaneous Anonymous Group* (LSAG), based on the Discrete Logarithm Problem. More digital signatures based on this assumption have been introduced in recent years and have found application for example within the Monero blockchain [13, 22].

Nowadays, ring signatures schemes have an application in cryptocurrencies and e-voting [25].

Post-quantum cryptography and group actions. With recent developments that are bringing us closer to the advent of quantum computing, many cryptographic algorithms can no longer be considered secure. For example, all the signatures described in the previous paragraph are based on cryptographic assumptions that are broken by the well known Shor's algorithm [27].
In the last years, NIST launched a Post-Quantum Standardisation Contest, now come to an end, that aimed to find protocols based on cryptographic assumptions that appear to be resistant even in case the quantum computer arrives. The most promising ones are based on lattice-based cryptography, multivariate cryptography, hash-based cryptography, isogeny-based cryptography and code-based cryptography.
Other interesting post-quantum assumptions derive from Cryptographic Group Actions, introduced by Alamati, De Feo, Montgomery and Patranabis [1] in 2020. Group actions are an interesting tool to generalise some well-known assumptions like the Discrete Logarithm Problem. The most promising and studied post-quantum cryptographic group action is CSIDH [7], but the topic has received a lot of interest, both in theoretical and applied fashion [3, 5, 6, 10, 14, 16, 28].

Concurrent works. In recent years, various digital ring signatures based on post-quantum assumptions have been proposed. In 2019, Esgin, Zhao, Steinfeld, Liu and Liu proposed *MatRiCT* [11], while Lu, Au and Zhang described *Raptor* [19]; both signatures are based on lattices assumptions, more precisely the former is based on Module Shortest Integer Solution (MSIS) and Module Learning With Errors (MLWE), while the latter is based on the NTRU assumption. Shortly after, Beullens, Katsumata and Pintore proposed two different ring signatures, known as *Calamari* and *Falafl* [5]: Calamari is up to date the only ring signature based on isogenies, more precisely on the CSIDH assumption, while Falafl is again based on the MSIS and MLWE assumptions. In 2021 were proposed two non-linkable schemes, both based on MSIS and MLWE. The first one by Lyubashevsky, Nguyen and Seiler is called *SMILE* [20] and is based on set membership proofs. The second one, *DualRing* [29], uses an innovative construction based on two rings. By following the same line of research of [5], Barenghi, Biasse, Ngo, Persichetti and Santini proposed the (linkable) ring version [2] of *LESS* [3], a signature whose security assumption is based on the code equivalence problem. Finally, recently Bellini, Esser, Sanna and Verbel proposed *MRr-DSS* [4], a non-linkable ring signature based on the MinRank problem.

Our contribution. In this work we present TRIFORS, a logarithmic post-quantum (linkable) ring signature based on a novel assumption regarding equivalence of alternating trilinear forms. This work is built starting from the digital signature presented by Tang et al. [28] at EUROCRYPT 22 and the construction introduced by Beullens, Katsumata and Pintore [5] to obtain a linkable ring signature from a group action. The signature is based on a novel cryptographic assumption, stating that the *search Alternating Trilinear Form Equivalence* (sATFE) problem is intractable. We design a base OR Sigma protocol, having soundness error of $1/2$. The term "OR" refers to the fact that the prover knows at least a secret key for the public keys in the ring. Given a security parameter λ, we decrease the soundness error to $1/2^\lambda$, running the protocol in parallel and using some optimisations. In particular, we use a fixed-weight challenge and, via a well-known combinatorial technique, we compress it in a string of length $\sim \lambda$ bits. To the best of our knowledge, we are the first applying this technique to shorten the signatures, without affecting the security of the scheme. We present what we called "main OR Sigma protocol" in two versions, with and without tag. Applying the Fiat-Shamir transform [12], we get two constructions: a ring signature from the OR sigma protocol without tag, called TRIFORS, and a linkable ring signature Link-TRIFORS from the version with tag.
Our post-quantum ring signature has logarithmic length in the size of the ring and competes with the state-of-the-art, as shown in Table 1 and Table 2 for non-linkable version: the only scheme having signatures significantly shorter than ours (for small rings) is *Calamari* [5], but it pays the heavy computation induced by isogenies. Moreover, TRIFORS performs even slightly better than the novel ring signature on codes [2].
This work is organized as follows: Section 2 sets some notations and recalls some preliminaries, while in sections 3 and 4 we define the two sigma protocols on which the (linkable) ring signature given in Section 5 is based. In Section 6 is given an overview on the attacks and finally, in Section 7 are reported some optimal parameters of the scheme. We also give some hints for possible future works.

Scheme	λ	*Assumption*	R							
			2^1	2^3	2^5	2^6	2^{10}	2^{12}	2^{15}	2^{21}
TRIFORS	128	sATFE	**9.0**	**10.5**	**12.0**	**12.7**	**15.7**	**17.1**	**19.3**	**23.8**
Raptor [19]	100	NTRU	2.5	10	/	81	1290	5161	/	/
Calamari [5]	128	CSIDH-512	3.5	5.4	/	8.2	/	14	/	23
Falafl [5]	128	MSIS, MLWE	29	30	/	32	/	35	/	39
SMILE [20]	128	MSIS, MLWE	/	/	16	/	17.3	/	18.7	/
Ring LESS [2]	128	Perm. Code Eq.	/	10.8	/	13.7	/	19.7	/	28.6
MRr-DSS [4]	128	MinRank	/	27	32	36	145	422	/	/

Table 1: Size in KB of the signatures. R is the size of the ring.

Scheme	λ	*Assumption*	R				
			2^1	2^3	2^6	2^{12}	2^{21}
TRIFORS	192	sATFE	**20.2**	**23.2**	**27.7**	**36.6**	**50**
MatRiCT [11]	192	MSIS, MLWE	18	19	31	59	148
Falafl for 2 [5]	192	MSIS, MLWE	49	50	52	55	59
DualRing-LB [29]	192	MSIS, MLWE	4.5	4.6	6	55	/

Table 2: Size in KB of the signatures. R is the size of the ring.

2 Preliminaries

2.1 Notation

Let $\mathbb{N} = \{1, 2, \dots\}$ and $\mathbb{R}$ be the sets of natural and real numbers, respectively. We denote with λ the security parameter. We denote with $\mathsf{poly}(\cdot)$ a function that is polynomial in its argument, i.e. there exists a positive integer m such that $\mathsf{poly}(x) = O(x^m)$. A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible* if there exists n_0 such that for every $n > n_0$ we have $\epsilon(n) \leq 1/p(n)$ for every polynomial p. A function not having this propriety is called *non-negligible*. The probability of an event is *overwhelming* if it is equal to $1 - \epsilon$, where ϵ is a negligible function. We say that an adversary has an *advantage* $\mathfrak{a}$ when playing a game against a challenger if its probability of winning that game is $\frac{1}{2} + \mathfrak{a}$. For a prime power q, $\mathbb{F}_q$ is the finite field with q elements, and $(\mathbb{F}_q)^n$ is the n-dimensional vector space over $\mathbb{F}_q$. The group of invertible $n \times n$ matrices with coefficients in $\mathbb{F}_q$ is denoted with $\mathrm{GL}_n(q)$. The *Hamming weight* of a vector x is the number of its non-zero coordinates, and its denoted with $\mathrm{w}(x)$. With $||$ we denote the concatenation of strings or vectors. Given a binary string x, $|x|$ denotes the bit length of x.
We denote the Random Oracle with RO and we augment it with some functionalities:

- a commitment functionality $\mathsf{RO}_{\mathrm{Com}}(x, r)$ where x is the committed value and r a random string;
- a seed expansion functionality $\mathsf{RO}_E(\mathsf{seed})$, where the output is defined by the context;
- a collision-free hash function RO_H with output length 2λ.

In the pseudocode, "$\xleftarrow{\$}$" denotes the random sampling, "$\leftarrow$" is a variable assignement and "$=$" is the check of equality.

2.2 Alternating trilinear forms

Here we define alternating trilinear forms and the cryptographic assumptions at the base of our protocol.

Definition 1. Given positive integers k, n, m and a prime power q, a map

$$\phi : \underbrace{(\mathbb{F}_q)^n \times \cdots \times (\mathbb{F}_q)^n}_{k \text{ times}} \to (\mathbb{F}_q)^m$$

can be

1. *alternating*: if ϕ is equal to the zero vector whenever two of its arguments are equal;
2. *k-linear*: if ϕ is linear in each of its k arguments.

If $m = 1$, i.e. the codomain of ϕ is the field $\mathbb{F}_q$, we say that ϕ is a *form*.

An *alternating trilinear form* is a map

$$\phi : (\mathbb{F}_q)^n \times (\mathbb{F}_q)^n \times (\mathbb{F}_q)^n \to \mathbb{F}_q$$

that is alternating and trilinear. The set of alternating trilinear forms over $(\mathbb{F}_q)^n$ is denoted with $\mathrm{ATF}(q, n)$.

It is known that $\mathrm{ATF}(q, n)$ is a linear space over $\mathbb{F}_q$ of dimension $\binom{n}{3}$. This implies that any alternating trilinear form can be represented and stored with $\binom{n}{3}\lceil \log_2 q \rceil$ bits.

Starting from $\mathrm{GL}_n(q)$, the group of $n \times n$ invertible matrices over $\mathbb{F}_q$, a group action over $\mathrm{ATF}(q, n)$ can be defined.

Definition 2. The group action $(\mathrm{GL}_n(q), \mathrm{ATF}(q, n), \star)$ is defined by

$$\begin{aligned} \star : \mathrm{GL}_n(q) \times \mathrm{ATF}(q, n) &\to \mathrm{ATF}(q, n) \\ (A, \phi) &\mapsto \phi \circ A^t. \end{aligned} \tag{1}$$

In other words, the alternating trilinear form $(A \star \phi)(x, y, z)$ is the map

$$\phi(A^t x, A^t y, A^t z).$$

The group action above defines an equivalence, indeed we say that ϕ and $A \star \phi$ are equivalent. Given two alternating trilinear forms, we can define the problem of deciding if there is an equivalence, and the problem of finding a matrix that sends one into the other. We formalize this in the following definition.

Definition 3. The *Alternating Trilinear Form Equivalence* (ATFE) problem is given by

- *Input*: ϕ, ψ in $\mathrm{ATF}(q, n)$.
- *Output*: YES if there exists A in $\mathrm{GL}_n(q)$ such that $\phi = A \star \psi$ and NO otherwise.

The *search Alternating Trilinear Form Equivalence* (sATFE) problem is given by

- *Input*: ϕ, ψ in $\mathrm{ATF}(q, n)$ such that they are equivalent.
- *Output*: A in $\mathrm{GL}_n(q)$ such that $\phi = A \star \psi$.

The assumption that the sATFE problem is intractable comes from the fact that its decisional counterpart ATFE is TI-complete: the TI complexity class [14] contains all those problems reducible to d-Tensor Isomorphism for some positive integer d. The ATFE problem is polynomially equivalent to d-Tensor Isomorphism for $d \geq 3$, hence by definition it is TI-complete. This implies that ATFE is as hard as all TI-complete problems from [14]. At the moment, no polynomial-time algorithm solving any TI-complete problem is known.

2.3 Sigma protocols

Given an NP-relation $\mathcal{R}$, a *sigma protocol* is a three-move interactive protocol between a prover $\mathcal{P} = (\mathcal{P}_{\mathsf{com}}, \mathcal{P}_{\mathsf{resp}})$ and a verifier $\mathcal{V}$. We assume that the prover uses some fixed randomness for its algorithms $(\mathcal{P}_{\mathsf{com}}, \mathcal{P}_{\mathsf{resp}})$, and they share their internal states. The output of $\mathcal{V}$ is assumed to be in $\{0, 1\}$. More formally, given a pair $(x, w) \in \mathcal{R}$ where x is the instance and w is the witness for x, the protocol follows the flow in Figure 1. We assume that both the parties have access to a random oracle RO and the verifier accepts if the algorithm $\mathcal{V}$ returns 1. The *transcript* of the protocol is defined as the triple $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$. The challenge ch is sampled from the space S_{ch}.

To be suitable for our application, a sigma protocol must present the following security properties.

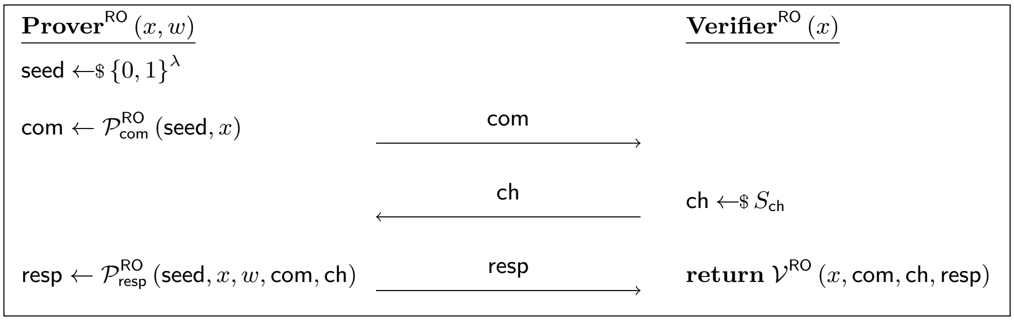
Definition 4. A sigma protocol is *correct* if, for all $(x, w) \in \mathcal{R}$ we have

$$\mathbf{P}\left[\mathcal{V}^{\mathsf{RO}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1 \;\middle|\; \begin{array}{c} \mathsf{com} \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{com}}(\mathsf{seed}, x), \\ \mathsf{ch} \leftarrow\!\$\, S_{\mathsf{ch}}, \\ \mathsf{resp} \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{resp}}(\mathsf{seed}, x, w, \mathsf{com}, \mathsf{ch}) \end{array} \right] = 1.$$

Definition 5. A sigma protocol has *special 2-soundness* if there exists a polynomial-time algorithm $\mathcal{E}$ called extractor such that, given two accepting transcripts $(\mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1)$ and $(\mathsf{com}, \mathsf{ch}_2, \mathsf{resp}_2)$ with $\mathsf{ch}_1 \neq \mathsf{ch}_2$, we have that the probability

$$\mathbf{P}\left[(x, w) \in \mathcal{R} : w \leftarrow \mathcal{E}^{\mathsf{RO}}(x, (\mathsf{com}, \mathsf{ch}_1, \mathsf{resp}_1), (\mathsf{com}, \mathsf{ch}_2, \mathsf{resp}_2)) \right]$$

is overwhelming.



$\mathbf{Prover}^{\mathsf{RO}}(x, w)$ $\mathbf{Verifier}^{\mathsf{RO}}(x)$

$\mathsf{seed} \leftarrow\$ \{0,1\}^{\lambda}$

$\mathsf{com} \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{com}}(\mathsf{seed}, x)$ $\xrightarrow{\mathsf{com}}$

$\xleftarrow{\mathsf{ch}}$ $\mathsf{ch} \leftarrow\$ S_{\mathsf{ch}}$

$\mathsf{resp} \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{resp}}(\mathsf{seed}, x, w, \mathsf{com}, \mathsf{ch})$ $\xrightarrow{\mathsf{resp}}$ $\mathbf{return}\ \mathcal{V}^{\mathsf{RO}}(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$

Figure 1: Generic Sigma Protocol

Definition 6. A sigma protocol has *special zero-knowledge* if there exists a probabilistic polynomial-time algorithm $\mathcal{S}$, the *simulator*, with access to the random oracle RO such that for any $(x, w) \in \mathcal{R}$, $\mathsf{ch} \in S_{\mathsf{ch}}$ and any adversary $\mathcal{A}$ making at most a polynomial number of queries to RO, we have that, if $\mathcal{P}$ denotes the pair of algorithms $(\mathcal{P}_{\mathsf{com}}, \mathcal{P}_{\mathsf{resp}})$, then

$$\left|\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{P}^{\mathsf{RO}}(x, w, \mathsf{ch})) = 1\right] - \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}^{\mathsf{RO}}(x, \mathsf{ch})) = 1\right]\right|$$

is negligible in the security parameter λ.

Definition 7. A sigma protocol has *high min-entropy* if for any $(x, w) \in \mathcal{R}$ and any adversary $\mathcal{A}$, the probability

$$\mathbf{P}\left[\mathsf{com}_1 = \mathsf{com}_2 \;\middle|\; \begin{array}{c} \mathsf{com}_1 \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{com}}(\mathsf{seed}, x, w), \\ \mathsf{com}_2 \leftarrow \mathcal{A}^{\mathsf{RO}}(x, w) \end{array}\right]$$

is negligible in the security parameter λ.

A useful property for obtaining a short signature when applying the Fiat-Shamir transform is the following.

Definition 8. A sigma protocol is *commitment reproducible* if there exists a PPT algorithm RecCom such that, for any pair (x, w) in $\mathcal{R}$, we have that

$$\mathbf{P}\left[\mathsf{RecCom}(x, \mathsf{ch}, \mathsf{resp}) = \mathsf{com} \;\middle|\; \begin{array}{c} \mathsf{com} \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{com}}(\mathsf{seed}, x), \\ \mathsf{ch} \leftarrow\$ S_{\mathsf{ch}}, \\ \mathsf{resp} \leftarrow \mathcal{P}^{\mathsf{RO}}_{\mathsf{resp}}(\mathsf{seed}, x, w, \mathsf{com}, \mathsf{ch}), \\ \mathcal{V}^{\mathsf{RO}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1 \end{array}\right]$$

is overwhelming in λ.

This property allows to send as signature only the challenge ch and the response resp, reducing its size. The verifier can reconstruct the commitment com using the algorithm RecCom.

2.4 Ring signatures

We define what a ring signature is and some additional properties that will be useful for the later sections of this paper.

Definition 9. A *ring signature* consists of four probabilistic polynomial-time (PPT) algorithms Setup, KGen, Sign and Verify such that:

- Setup: it takes as input the security parameter of length λ and it returns the public parameters pp used by the scheme.
- KGen: it takes as inputs the public parameters pp and some random coins rr. It returns the secret/public key pair $(\mathsf{sk}, \mathsf{pk})$.
- Sign: it takes as inputs a secret key sk_I, where $I \in \{1, \dots, R\}$ is the index of the signer in the ring, the message msg and the ring $\Omega = \{\mathsf{pk}_1, \dots, \mathsf{pk}_R\}$. The public key obtained starting from sk_I must belong to the ring, that is $\mathsf{pk}_I \in \Omega$. The output of the algorithm is the signature σ.

- Verify: it takes as inputs the ring $\Omega = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}$, the message msg and the signature σ. It returns 1 (accept) if the signature is valid and 0 (refuse) otherwise.

Ring signatures must satisfy three core properties: *correctness*, *anonymity* and *unforgeability.*

Definition 10. A ring signature is *correct* if for every security parameter $\lambda \in \mathbb{N}$, for every ring Ω composed of $R = \mathsf{poly}(\lambda)$ public keys, for every index $I \in \{1, \ldots, R\}$ and for every message msg, we have that

$$\mathbf{P}\left[\mathsf{Verify}(\Omega, \mathsf{msg}, \sigma) = 1 \;\middle|\; \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KGen}(\mathsf{pp}, \mathsf{rr_i}) \; \forall i \in \{1, \ldots, R\}, \\ \Omega := \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_I, \mathsf{msg}, \Omega). \end{array}\right] = 1.$$

Informally, this means that the verification algorithm of a signature generated correctly will always output 1.

Definition 11. A ring signature is *anonymous* if for every security parameter $\lambda \in \mathbb{N}$ and for every ring composed of $R = \mathsf{poly}(\lambda)$ public keys, any PPT adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:

(A) The challenger first runs the algorithm Setup that outputs pp and then the algorithm KGen, together with random coins $\mathsf{rr_i}$, to obtain R secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, $i \in \{1, \ldots, R\}$. He samples a bit $b \leftarrow\$ \{0, 1\}$.

(B) The challenger gives pp and the list of random coins $(\mathsf{rr}_1, \ldots, \mathsf{rr}_R)$ to $\mathcal{A}$.

(C) $\mathcal{A}$ sends to the challenger a challenge $(\Omega, \mathsf{msg}, i_0, i_1)$. The ring Ω must contain the public keys pk_{i_0} and pk_{i_1}. The challenger computes the signature $\sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}_{i_b}, \mathsf{msg}, \Omega)$ and sends it to $\mathcal{A}$.

(D) $\mathcal{A}$ outputs a bit b^* and wins if $b = b^*$.

This property means that it should not be possible to guess the secret key that was used to produce a signature, even if the adversary knows all the secret keys that were used to generate the public keys in the ring.

Definition 12. A ring signature is *unforgeable* if for every security parameter $\lambda \in \mathbb{N}$ and for every ring composed of $R = \mathsf{poly}(\lambda)$ public keys, any PPT adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:

(A) The challenger first runs the algorithm Setup that outputs pp and then the algorithm KGen, together with random coins $\mathsf{rr_i}$, to obtain R secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, $i \in \{1, \ldots, R\}$. He calls $V = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}$ the set with the public keys. He finally initialises two empty sets S and C.

(B) The challenger gives pp and the set V to $\mathcal{A}$.

(C) $\mathcal{A}$ can create signing and corruption queries a polynomial number of times:

  - $(\mathsf{AdvSign}, i, \mathsf{msg}, \Omega)$: the challenger checks if $\mathsf{pk}_i \in \Omega \subseteq V$. If that is true, he computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{pk}_i, \mathsf{msg}, \Omega)$. The challenger gives σ to $\mathcal{A}$ and adds $(i, \mathsf{msg}, \Omega)$ to S.
  - $(\mathsf{AdvCorrupt}, i)$: the challenger adds pk_i to C and returns $\mathsf{rr_i}$ to $\mathcal{A}$.

(D) $\mathcal{A}$ outputs $(\Omega^*, \mathsf{msg}^*, \sigma^*)$. If $\Omega^* \subset V \setminus C$, $(\cdot, \mathsf{msg}^*, \Omega^*) \notin S$ and $\mathsf{Verify}(\Omega^*, \mathsf{msg}^*, \sigma^*) = 1$, then the adversary wins.

Finally, this property means that it should be impossible to forge a valid signature without knowing one secret key corresponding to one of the public keys in the ring.

Moreover, a ring signature can have the additional property of being *linkable*, that is everyone can check if two signatures were produced by the same signer (i.e., by the same secret key).

Definition 13. A *linkable* ring signature is a scheme that consists of the four PPT algorithms previously described for a classic ring signature scheme, plus the following PPT algorithm:

- Link: the inputs are two different signatures σ_0 and σ_1. The algorithm outputs 1 if the two signatures were produced starting from the same secret key and 0 otherwise.

Furthermore, a linkable ring signature must satisfy the following additional properties: *linkability*, *linkable anonymity* and *non-frameability.* Notice that the correctness property must be slightly modified: if the same user generates two signatures correctly, both the Verify and the Link algorithms will always output 1.

Definition 14. A linkable ring signature is *linkable* if for every security parameter $\lambda \in \mathbb{N}$ and for every ring composed of $R = \mathsf{poly}(\lambda)$ public keys, any PPT adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:

(A) The challenger runs the algorithm Setup that outputs pp and gives it to $\mathcal{A}$.

(B) $\mathcal{A}$ runs the algorithm KGen and outputs the set $V = \{\mathsf{pk}_1, \dots, \mathsf{pk}_R\}$ and the set of tuples $\{(\sigma_1, \mathsf{msg}_1, \Omega_1), \dots, (\sigma_{R+1}, \mathsf{msg}_{R+1}, \Omega_{R+1})\}$.

(C) $\mathcal{A}$ wins if these three conditions hold:

- $\forall i \in \{1, \dots, R+1\}$, we have $\Omega_i \subseteq V$.
- $\forall i \in \{1, \dots, R+1\}$, we have that the algorithm $\mathsf{Verify}(\Omega_i, \mathsf{msg}_i, \sigma_i)$ outputs 1.
- $\forall i, j \in \{1, \dots, R+1\}$ such that $i \neq j$, we have $\mathsf{Link}(\sigma_i, \sigma_j) = 0$.

The linkability property tells us that if an adversary produces more than k signatures with a set of k public keys, then the Link algorithm will output 1 for at least one pair of signatures.

Definition 15. A linkable ring signature is *linkable anonymous* if for every security parameter $\lambda \in \mathbb{N}$ and for every ring composed of $R = \mathsf{poly}(\lambda)$ public keys, any PPT adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:

(A) The challenger first runs the algorithm Setup that outputs pp and then the algorithm KGen, together with random coins $\mathsf{rr_i}$, to obtain R secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, $i \in \{1, \dots, R\}$. He calls $V = \{\mathsf{pk}_1, \dots, \mathsf{pk}_R\}$ the set with the public keys. He also samples a bit $b \leftarrow\$ \{0, 1\}$.

(B) The challenger gives to the adversary pp and the set V.

(C) The adversary chooses and outputs two public keys $(\mathsf{pk}_0^*, \mathsf{pk}_1^*) \in V$. We denote with $(\mathsf{sk}_0^*, \mathsf{sk}_1^*)$ the respective secret keys.

(D) The challenger gives to $\mathcal{A}$ all the random coins $\mathsf{rr_i}$ related to the public keys $\mathsf{pk}_i \in V \backslash \{\mathsf{pk}_0^*, \mathsf{pk}_1^*\}$.

(E) $\mathcal{A}$ queries for signatures, giving as inputs to the challenger a public key $\mathsf{pk} \in \{\mathsf{pk}_0^*, \mathsf{pk}_1^*\}$, a message msg and a ring Ω that contains pk_0^* and pk_1^*:

- If $\mathsf{pk} = \mathsf{pk}_0^*$, the challenger outputs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_b^*, \mathsf{msg}, \Omega)$.
- If $\mathsf{pk} = \mathsf{pk}_1^*$, the challenger outputs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{1-b}^*, \mathsf{msg}, \Omega)$.

(F) $\mathcal{A}$ outputs a bit b^*, and he wins the game if $b = b^*$.

This property says that an adversary cannot guess which secret key was used to produce signatures. Differently from the anonymity property, in this case the adversary does not have access to all the secret keys, otherwise he could use the linkability to understand who was the signer.

Definition 16. A linkable ring signature is *non-frameable* if for every security parameter $\lambda \in \mathbb{N}$, for every ring composed of $R = \mathsf{poly}(\lambda)$ public keys, any PPT adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:

(A) The challenger first runs the algorithm Setup that outputs pp and then the algorithm KGen, together with random coins $\mathsf{rr_i}$, to obtain R secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, $i \in \{1, \dots, R\}$. He calls $V = \{\mathsf{pk}_1, \dots, \mathsf{pk}_R\}$ the set with the public keys. He finally initialises two empty sets S and C.

(B) The challenger gives pp and the set V to the adversary $\mathcal{A}$.

(C) $\mathcal{A}$ can create signing and corruption queries a polynomial number of times:

- $(\mathsf{AdvSign}, i, \mathsf{msg}, \Omega)$: the challenger checks if $\mathsf{pk}_i \in \Omega \subseteq V$. If that is true, he computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_i, \mathsf{msg}, \Omega)$. The challenger gives σ to $\mathcal{A}$ and adds $(i, \mathsf{msg}, \Omega)$ to S.
- $(\mathsf{AdvCorrupt}, i)$: the challenger adds pk_i to C and returns $\mathsf{rr_i}$ to $\mathcal{A}$.

(D) $\mathcal{A}$ outputs $(\Omega^*, \mathsf{msg}^*, \sigma^*)$; he wins if these two conditions hold:

- $\mathsf{Verify}(\Omega^*, \mathsf{msg}^*, \sigma^*) = 1$ and $(\cdot, \mathsf{msg}^*, \Omega^*) \notin S$;
- $\mathsf{Link}(\sigma^*, \sigma) = 1$ for some signature σ given by the challenger starting from a query of the form $(i, \mathsf{msg}, \Omega) \in S$ with $\mathsf{pk}_i \in V \setminus C$.

Finally, this property tells us that it should not be possible for an adversary to create a valid signature that is linked to a signature produced by an honest party.

2.5 Index-hiding Merkle trees

A Merkle tree [21] is a well known data structure used for cryptography applications. It is a binary tree, where each leaf contains the hashes $\{a_1, \dots, a_M\}$ of some data that we want to hide, and every other node which is not a leaf is given by the hash of the concatenation of the values of its two children. The root of the tree represents its commitment. Suppose the tree has depth c: to efficiently check that a_i is a leaf of the tree, the prover must send to the verifier one information for each level of the tree: the verifier will then compute c different hashes, and check if the final value he obtains equals the committed root.

For our applications, we will consider *complete balanced* Merkle trees, that is trees where each node has exactly two children, excluding the leaves, whose number is equal to $M = 2^c$ for some positive integer c. Moreover, we consider a slight modification of the construction we have just described, so that the prover, in addition to proving that an element a_i is in the list, does not reveal its position within the tree. We call this structure *index-hiding Merkle tree.*

Following the notation used in [5], we define the Merkle tree algorithms that will be later used to define our Sigma protocol.

- $\mathsf{MerkleTree}$: the input of this algorithm is the list of M elements $A = \{a_1, \dots, a_M\}$, that represent the leaves of the tree. It computes the nodes of the whole binary tree up to its root. To get any internal node b, the algorithm computes the hash of the concatenation of its two children b_{left} and b_{right}. However, to obtain a index-hiding Merkle tree we need to concatenate the two elements following the lexicographical order, that is $b = \mathit{hash}(b_{\text{left}}||b_{\text{right}})_{\text{lex}}$. Proof of this fact is given in [5]. The two outputs of the algorithm are its root root, together with a representation of the whole tree, tree.
- $\mathsf{getMerklePath}$: the two inputs of this algorithm are the Merkle tree tree and a certain index $i \in \{1, \dots, M\}$. The output will be a list path, that contains an information about the sibling of a_i (i.e. the node with the same parent of a_i), together will all the siblings of any ancestor of a_i, ordered by decreasing height.
- $\mathsf{ReconstructRoot}$: the inputs of this algorithm are the list of M elements $A = \{a_1, \dots, a_M\}$ and the path path, which is the output of the previous algorithm $\mathsf{getMerklePath}$. The output is the reconstructed root root of the Merkle tree.

2.6 Seed trees

A *seed tree* is yet again a complete balanced binary tree, but its construction is different with respect to the Merkle tree one given is Section 2.5. In this case, the tree is built starting from its root as follows: given a node T represented by a binary string, its two children T_1, T_2 are given by $T_i = \mathsf{RO}_E(T, i)$ for $i = 1, 2$, obtained via a 2λ hash evaluated in the value of T. Each binary string has finally length equal to λ. In this way, to compute the leaves of any subtree with root T, it is sufficient to know T.

Seed trees have been used recently as a clever optimisation to decrease the length of the signature [2,4,5]. We follow again the notation used in [5] to introduce the algorithms that will later be used to optimise our Sigma protocol:

- $\mathsf{SeedTree}$: the inputs of this algorithm are a binary string $\mathsf{seed}_{\mathsf{root}}$ of length λ, which represents the root of the tree, and an integer M, the number of leaves of the tree. It computes the complete balanced binary tree with M leaves, where every node is computed expanding recursively the seed of the previous level of the tree. The algorithm returns a list that contains M seed values, one for each leaf.
- $\mathsf{ReleaseSeeds}$: the inputs of this algorithm are a binary string $\mathsf{seed}_{\mathsf{root}}$ of length λ, which represents the root of the tree, and a challenge c, a binary string of length M. The algorithm outputs the set S containing the seeds that belong to the leaves with index equal to 1 in the challenge. In order to send less information, we can exploit the seed tree structure and send a subset $\mathsf{seeds}_{\mathsf{int}} \subseteq S$ of nodes, where the seeds computed starting from the set $\mathsf{seeds}_{\mathsf{int}}$ is equal to the seeds contained in the set S.
- $\mathsf{RecoverLeaves}$: the inputs of the algorithm are the set $\mathsf{seeds}_{\mathsf{int}}$ and the binary challenge c of length M. The output is given by the set of all the seeds belonging to the leaves that can be computed starting from the set $\mathsf{seeds}_{\mathsf{int}}$, that is the binary strings corresponding to the ones of the challenge c.
- $\mathsf{SimulateSeeds}$: the input of the algorithm is a binary challenge c of length M. It computes the leaves with index equal to 1 and then it samples a random seed of length λ for each of these leaves. The output is the set $\mathsf{seeds}_{\mathsf{int}}$.

3 The Base OR Sigma Protocol

In this section we define the sigma protocol on which our ring signature is based. Let R be a positive integer. Fix an element ϕ in $\mathrm{ATF}(q,n)$ and let $\phi_i = A_i \star \phi$, where A_i is a randomly generated matrix from $\mathrm{GL}_n(q)$ for each $i = 1, \ldots, R$. The NP-relation for the protocol is the following:

$$\mathcal{R} = \{(\{\phi_1, \ldots, \phi_R\}, A) \mid \exists I \in \{1, \ldots, R\} \text{ s.t. } \phi_I = A \star \phi\}\,.$$

In the signature, if we see A_I as the secret key for the public key $\phi_I = A_I \star \phi$, the above relation models that the witness is the knowledge of at least a secret key for one of the public keys $\phi_1, \ldots, \phi_R$ in the ring.

The problem induced by the relation $\mathcal{R}$ is a variation of sATFE.

Definition 17. Let R be a positive integer and ϕ a public element of $\mathrm{ATF}(q,n)$. The *R-search Alternating Trilinear Form Equivalence* (R-sATFE) problem is given by

- *Input*: $\phi_1, \ldots, \phi_R$ in $\mathrm{ATF}(q,n)$ such that they are pairwise equivalent.
- *Output*: A in $\mathrm{GL}_n(q)$ and distinct i, j in $\{1, \ldots, R\}$ such that $\phi_j = A \star \phi_i$.

We can adapt the proof from [3] Theorem 3, reducing tightly R-sATFE to sATFE.

Proposition 18. *Given an algorithm* Alg *that solves* R-sATFE *with probability* ϵ*, there exists a polynomial algorithm* Alg'*, using* Alg *as an oracle, that solves* sATFE *with probability* $\epsilon/2$.

Proof. Given the instance (ϕ, ψ) of sATFE, we want to find A in $\mathrm{GL}_n(q)$ such that $\phi = A \star \psi$. Without loss of generality let R be even. The algorithm Alg' uniformly samples $B_1, \ldots, B_R$ from $\mathrm{GL}_n(q)$ and sets

$$\phi_i = \begin{cases} B_i \star \phi & \text{for } i \in \left\{1, \ldots, \frac{R}{2}\right\} \\ B_i \star \psi & \text{for } i \in \left\{\frac{R}{2}+1, \ldots, R\right\}. \end{cases} \tag{2}$$

After a random permutation π, Alg' asks the query $\left\{\phi_{\pi(i)}\right\}_{i=1,\ldots,R}$ to Alg. Observe that every ϕ_i is equivalent to both ϕ and ψ and then there is no way to decide if it is obtained from ϕ or ψ. The oracle Alg returns a matrix C and indexes h, k such that $\phi_h = C \star \phi_k$ for $k \neq h$.
With probability $\frac{1}{2}$, k and h are not in the same partition from Equation (2). Suppose that this is the case, then this implies $\phi_k = B_k \star \phi$ and $\phi_h = B_h \star \psi$ (here we assume without loss of generality that π is the identity, otherwise apply its inverse on the indices of the B's). The algorithm Alg' returns $B_k^{-1} C B_h$. Otherwise, if k and h are in the same partition, Alg' outputs a rejection.
Observe that Alg' is a polynomial-time algorithm and solves sATFE with probability $\epsilon/2$ using Alg as oracle. □

3.1 The protocol

The construction is the same used in [5], with minor changes, for example, here we do not need to abort. The idea on which we build our base OR sigma protocol is the following: given a public trilinear form ϕ, secret keys $\{A_1, \ldots, A_R\}$ and public keys $\phi_1 = A_1 \star \phi, \ldots, \phi_R = A_R \star \phi$, the prover wants to show to the verifier that he knows at least a secret key A_I among the public keys of the ring $\{\phi_1, \ldots, \phi_R\}$. In order to prove that, for each $i \in \{1, \ldots, R\}$ he generates random matrices B_i and computes $\psi_i = B_i \star \phi_i$ for each public key ϕ_i in the ring, then he sends these elements in a random order to the verifier that replies with a random bit b. If $b = 0$, the response resp is $B_I A_I$ and the verifier checks that $\mathsf{resp} \star \phi$ is in $\{\phi_1, \ldots, \phi_R\}$. If $b = 1$, the response consists of $B_1, \ldots, B_R$ and the verifier checks the set equality $\{B_1 \star \psi_1, \ldots, B_R \star \psi_R\} = \{\phi_1, \ldots, \phi_R\}$. To make the proof size logarithmic in R, since the matrices $B_1, \ldots, B_R$ are generated at random, the prover can send the seed that generates them as response if $b = 1$. We can also use a Merkle tree to commit instead of sending all the elements $\psi_1, \ldots, \psi_R$ and send the Merkle root as commitment. In this way, when the challenge is $b = 0$, the prover appends a path of the Merkle tree to retrieve the root. Moreover, we can use the same matrix B instead of R different matrices $B_1, \ldots, B_R$ without affecting the security of the scheme. All these consideration leads to the OR sigma protocol showed below.

We model a commitment scheme as a random oracle $\mathsf{RO}_{\mathrm{Com}}$, where the input is the committed value x and a random string r. We assume that the randomness produced by the prover derives from a seed seed expanded by the random oracle RO_E. The base OR Sigma protocol, using algorithms described in Figure 2, has the standard flow of every Sigma protocol: the prover sends the commitment com returned by $\mathcal{P}_{\mathcal{B}_{\mathsf{com}}}^{\mathsf{RO}}$ to the verifier, that replies with a random challenge ch in $\{0,1\}$; then, the prover computes its response running $\mathcal{P}_{\mathcal{B}_{\mathsf{resp}}}^{\mathsf{RO}}$ and sends it to the verifier, that accepts or rejects according to $\mathcal{V}_{\mathcal{B}}^{\mathsf{RO}}$.

```
$\mathcal{P}_{\mathcal{B}_{\mathsf{com}}}^{\mathsf{RO}}(\mathsf{seed}, \{\phi_1, \dots, \phi_R\})$ :
$(B, r_1, \dots, r_R) \leftarrow \mathsf{RO}_E(\mathsf{seed})$
for $i = 1, \dots, R$ do
    $C_i \leftarrow \mathsf{RO}_{\mathrm{Com}}(B \star \phi_i, r_i)$
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(C_1, \dots, C_R)$
$\mathsf{com} \leftarrow \mathsf{root}$
return $\mathsf{com}$

$\mathcal{P}_{\mathcal{B}_{\mathsf{resp}}}^{\mathsf{RO}}(\mathsf{seed}, A_I, \mathsf{tree}, \mathsf{ch})$ :
$(B, r_1, \dots, r_R) \leftarrow \mathsf{RO}_E(\mathsf{seed})$
if $\mathsf{ch} = 0$ then
    $D \leftarrow BA_I$
    $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I)$
    $\mathsf{resp} \leftarrow (D, \mathsf{path}, r_I)$
else
    $\mathsf{resp} \leftarrow \mathsf{seed}$
return $\mathsf{resp}$
```

```
$\mathcal{V}_{\mathcal{B}}^{\mathsf{RO}}(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ :
if $\mathsf{ch} = 0$ then
    $(D, \mathsf{path}, r) \leftarrow \mathsf{resp}$
    $\widetilde{\phi} \leftarrow D \star \phi$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\mathsf{path}, \mathsf{RO}_{\mathrm{Com}}(\widetilde{\phi}, r))$
else
    $(B, r_1, \dots, r_R) \leftarrow \mathsf{RO}_E(\mathsf{resp})$
    for $i = 1, \dots, R$ do
        $\widetilde{C}_i \leftarrow \mathsf{RO}_{\mathrm{Com}}(B \star \phi_i, r_i)$
    $(\widetilde{\mathsf{root}}, \widetilde{\mathsf{tree}}) \leftarrow \mathsf{MerkleTree}(\widetilde{C}_1, \dots, \widetilde{C}_R)$
if $\mathsf{com} = \widetilde{\mathsf{root}}$ then
    return accept
return reject
```

Figure 2: Algorithms for the base OR Sigma protocol

Theorem 19. *The base OR Sigma protocol with algorithms in Figure 2 is correct, special 2-sound and possesses special zero-knowledge in the Random Oracle Model under the assumption that R-sATFE is intractable.*

The proof of the theorem is standard and similar to the ones from [5], and it is reported in Appendix A.

4 Optimisations and the Main OR Sigma Protocol

In this section we modify the base Sigma protocol from Section 3 to decrease the soundness error from $1/2$ to $1/2^{\lambda}$, for a security parameter λ. A straightforward strategy is to run the protocol in parallel λ times. Moreover, we report some modification to this protocol to obtain shorter responses.

4.1 Using fixed weight challenges

Instead of picking the challenge uniformly from the space $\{0,1\}^{\lambda}$, we can force the number of ones to be a certain value, since the response for $\mathsf{ch} = 1$ is just a λ-bits seed, and is shorter compared to the response when $\mathsf{ch} = 0$. Let M be the length of the challenge and K be the number of zeros in the challenge. In this way we want to chose parameters such that $\binom{M}{K} > 2^{\lambda}$ and $M > \lambda$. The challenge space becomes $C_{M,K}$, the set of binary strings of length M and weight $M - K$ (or, equivalently, with exactly K zeros). This technique is well-known in literature.

We propose another optimization: instead of sending the challenge as a string in $C_{M,K}$, we can enumerate such $\binom{M}{K}$ strings and send the integer J_{ch} referring to the position of the challenge ch in such ordering. To convert an integer into a fixed weight binary string we use the combinatorial number system [17]. In this way only $\log_2 \binom{M}{K}$ bits per challenge are sent at the cost of a negligible increment of the computational effort in the signing and the verify processes. Note that usually the number of ones is fixed to be less than $M/2$, but here we want the reverse, i.e. we want a string with a large number of ones and few zeros; because of this, we use the reverse lexicographic order. To sample an element from $C_{M,K}$ we simply sample an integer J from $\{0, \dots, \binom{M}{K} - 1\}$ and see this as the position of the challenge ch_J in the lexicographic order in $C_{M,K}$. The challenge is encoded by the algorithm Unrank in Figure 3, having complexity $O(M)$.

The inverse procedure of computing the index J_{ch} from the challenge ch in $C_{M,K}$ is not needed in our protocol, but we report it for completeness. Let $j_1, \dots, j_K$ be the support of ch, i.e. the positions of the ones. The index J_{ch} is given by $\sum_{i=1}^{K} \binom{i}{j_i}$.

```
Given an integer $0 \leq J < \binom{M}{K}$ return $\mathsf{ch}_J$ in $C_{M,K}$
Unrank($J$) :
ch $\leftarrow (1, \dots, 1)$
$m \leftarrow M$
$I \leftarrow J$
for $i = 0, \dots, K-1$ do
    while $\binom{m}{K-i} \geq I$ do
        $m \leftarrow m - 1$
    $I \leftarrow I - \binom{m}{K-i}$
    $\mathsf{ch}_m \leftarrow 0$
    $m \leftarrow m - 1$
return ch
```

Figure 3: Unranking algorithm

4.2 Seed tree

We use a seed tree to communicate the seed seed for each repetition of the base sigma protocol having challenge bit ch $= 1$. Due to Section 4.1, the challenge has a larger number of ones, and this structure allows to reduce the response size.

Given a seed tree with M leaves, if we want to send $M-K$ leaves, the upper bound on the number of sent nodes is given by the following proposition. This fact is given in [15] without a proof, that we instead give in Appendix C. Observe that if the number of leaves is not a power of 2, we can add dummy leaves to reach the next power of 2 and complete the binary tree. We can exclude the case $K = 0$ since in that case it is sufficient to send the root, so we just send one node.

Proposition 20. *Given a seed tree with $M = 2^c$ leaves, if we want to send $M - K$ leaves, with $K \neq 0$, we transmit at most $K \log_2 M - K\lceil \log_2 K \rceil + 2^{\lceil \log_2 K \rceil} - K$ seeds.*

To give a more accurate estimate of the signature length, we also study the minimal number of nodes we have to transmit for a tree with M leaves, if we want to send $M - K$ leaves. The proof of this result is reported in Appendix C. For any non-negative integer K, we denote with $\mathbf{b}(K)$ its binary representation and with $\mathrm{w}(\mathbf{b}(K))$ the number of ones in $\mathbf{b}(K)$.

Proposition 21. *Given a seed tree with $M = 2^c$ leaves, if we want to send $M - K$ leaves, with $K \neq 0$, we transmit at least $c - \mathrm{w}(\mathbf{b}(K-1))$ seeds.*

Given the lower bound and the upper bound of the number of nodes of the seed tree that we must send, we want now to answer the following question: *Given $M = 2^c$ and given $M - K$ the number of leaves we want to send, how many nodes of the tree will be transmitted on average*? An analysis given in [24], about the average number of encryption in a CST broadcast encryption scheme, gives us the answer to the above question. In fact, the structure of our seed tree is the same as the key tree used in that protocol, and sending a seed coincides with giving a user the privilege of decrypting data. We can reformulate [24, Th. 8] to obtain the following result on seed trees.

Theorem 22. *Given a seed tree with $M = 2^c$ leaves, if we want to send $M - K$ leaves, the average number of seeds to transmit is given by*

$$\sum_{k=0}^{\lfloor \log_2(M-K) \rfloor} \frac{M-K}{2^k} \cdot \frac{\binom{M-2^k}{M-K-2^k} - \binom{M-2^{k+1}}{M-K-2^{k+1}}}{\binom{M-1}{M-K-1}}.$$

We use these bounds to estimate and minimize the length of the signature in Section 7.

4.3 Salting

In [5] they point out that to make tight reductions a salt is needed when we call the RO. Moreover for each repetition i, $1 \leq i \leq M$, of the base sigma protocol, we use the "salted" random oracle $\mathsf{RO}^i(\cdot) = \mathsf{RO}(\mathsf{salt}||i||\cdot)$, with a random string salt of 2λ bits.

4.4 The main OR sigma protocol

In this protocol we run the base OR sigma protocol in parallel to achieve a lower soundness error. Moreover we introduce some of the optimizations cited in this section, namely the use of a fixed weight challenge, the seed tree and the salting. The scheme is presented in Figure 4. Let M be the length of the challenge and K be the number of zeroes. Both M and K are chosen accordingly to the security parameter λ. The challenge space S_{ch} is the set $\{0, \dots, \binom{M}{K} - 1\}$.

```
$\underline{\mathcal{P_M}^{\mathsf{RO}}_{\mathsf{com}}(\{\phi_1, \dots, \phi_R\})}$ :
$\mathsf{seed} \leftarrow\$ \{0,1\}^\lambda$
$\mathsf{salt} \leftarrow\$ \{0,1\}^{2\lambda}$
$\mathsf{RO}' \leftarrow \mathsf{RO}(\mathsf{salt}||\cdot)$
$(\mathsf{seed}^{(i)}, \dots, \mathsf{seed}^{(M)}) \leftarrow \mathsf{SeedTree}^{\mathsf{RO}'}(\mathsf{seed}, M)$
for $i = 1, \dots, M$ do
  $\mathsf{RO}^i \leftarrow \mathsf{RO}'(i||\cdot)$
  $\mathsf{com}^{(i)} \leftarrow \mathcal{P_B}^{\mathsf{RO}^i}_{\mathsf{com}}\left(\mathsf{seed}^{(i)}, \{\phi_1, \dots, \phi_R\}\right)$
endfor
$\mathsf{com} \leftarrow (\mathsf{salt}, (\mathsf{com}^{(i)})_{i=1,\dots,M})$
return $\mathsf{com}$

$\underline{\mathcal{P_M}^{\mathsf{RO}}_{\mathsf{resp}}(A_I, J_{\mathsf{ch}})}$ :
$\mathsf{ch} \leftarrow \mathsf{Unrank}(J_{\mathsf{ch}})$
for $i$ s.t. $\mathsf{ch}_i = 0$ do
  retrieve $\mathsf{tree}^{(i)}$ and $\mathsf{seed}^{(i)}$
  $\mathsf{resp}^{(i)} \leftarrow \mathcal{P_B}^{\mathsf{RO}^i}_{\mathsf{resp}}\left(\mathsf{seed}^{(i)}, A_I, \mathsf{tree}^{(i)}, \mathsf{ch}_i\right)$
endfor
$\mathsf{seeds_{int}} \leftarrow \mathsf{ReleaseSeeds}^{\mathsf{RO}'}(\mathsf{seed}, \mathsf{ch})$
$\mathsf{resp} \leftarrow (\mathsf{seeds_{int}}, (\mathsf{resp}^{(i)})_{\mathsf{ch}_i=0})$
return $\mathsf{resp}$
```

```
$\underline{\mathcal{V_M}^{\mathsf{RO}}(\mathsf{com}, J_{\mathsf{ch}}, \mathsf{resp})}$ :
$\mathsf{ch} \leftarrow \mathsf{Unrank}(J_{\mathsf{ch}})$
$\mathsf{RO}' \leftarrow \mathsf{RO}(\mathsf{salt}||\cdot)$
$(\mathsf{seeds_{int}}, (\mathsf{resp}^{(i)})_{\mathsf{ch}_i=0}) \leftarrow \mathsf{resp}$
$(\mathsf{resp}^{(i)})_{\mathsf{ch}_i=1} \leftarrow \mathsf{RecoverLeaves}^{\mathsf{RO}'}(\mathsf{seeds_{int}}, \mathsf{ch})$
for $i = 1, \dots, M$ do
  if $\mathcal{V_B}^{\mathsf{RO}'}\left(\mathsf{com}^{(i)}, \mathsf{ch}_i, \mathsf{resp}^{(i)}\right)$ rejects then
    return reject
  endif
endfor
return accept
```

Figure 4: Algorithms for the main OR Sigma protocol

Theorem 23. *The main OR Sigma protocol with algorithms in Figure 4 is correct, special 2-sound, and possesses both high min-entropy and special zero-knowledge in the Random Oracle Model under the assumption that R-sATFE is intractable.*

4.5 Tags and linkability

Following the construction in [5], we add "tags" to our sigma protocol. Given the group action of $\mathrm{GL}_n(q)$ over $\mathrm{ATF}(q, n)$ from Definition 2, we need another action of $\mathrm{GL}_n(q)$ on a set Y. Invertible matrices can act on a huge variety of sets, such as spaces of matrices or tensors [14]. We use and define the following action, similar to ICE from [2].

Definition 24. The group action $(\mathrm{GL}_n(q), \mathrm{ATF}(q,n), \bullet)$ is defined by

$$\begin{aligned} \bullet : \mathrm{GL}_n(q) \times \mathrm{ATF}(q,n) &\to \mathrm{ATF}(q,n) \\ (A,\phi) &\mapsto \phi \circ A^{-1}. \end{aligned} \tag{3}$$

This action leads to the following problem.

Definition 25. The *Inverse Alternating Trilinear Form Equivalence* (IATFE) problem is given by

- *Input*: ϕ, ψ in $\mathrm{ATF}(q,n)$.
- *Output*: YES if there exists A in $\mathrm{GL}_n(q)$ such that $\phi = A \bullet \psi$ and NO otherwise.

The *search Inverse Alternating Trilinear Form Equivalence* (sIATFE) problem is given by

- *Input*: ϕ, ψ in $\mathrm{ATF}(q,n)$ such that they are equivalent.
- *Output*: A in $\mathrm{GL}_n(q)$ such that $\phi = A \bullet \psi$.

Given two fixed elements ϕ, ψ in $\mathrm{ATF}(q,n)$ and a set of secret keys $\{A_1, \dots, A_R\}$ corresponding to public keys $\{\phi_1, \dots, \phi_R\}$ such that $\phi_i = A_I \star \phi$ for each i from 1 to R, we define the "tag" associated to the I-th public key as $\tau_I = A_I \bullet \psi$. When the I-th user signs a message, he appends its tag to the signature. A verifier can link two signatures if they possess the same tag. The OR sigma protocol is modified to add the proof that τ_I is generated by the same secret key A_I. We present the base OR Sigma protocol with tags using the same structure of the base Sigma protocol in Figure 2 with algorithms from Figure 5. The main differences with the protocol of Section 3 is the introduction of the proof of knowledge for the tag τ_I and the replacement of the commitment as the hash of the concatenation of root and the masked tag τ'.

```
$\mathcal{P}^{\mathsf{RO}}_{\mathcal{B}L,\mathsf{com}}(\mathsf{seed}, \{\phi_1, \dots, \phi_R\}, \tau)$ :
$(B, r_1, \dots, r_R) \leftarrow \mathsf{RO}_E(\mathsf{seed})$
for $i = 1, \dots, R$ do
    $C_i \leftarrow \mathsf{RO}_{\mathrm{Com}}(B \star \phi_i, r_i)$
$(\mathsf{root}, \mathsf{tree}) \leftarrow \mathsf{MerkleTree}(C_1, \dots, C_R)$
$\tau' \leftarrow B \bullet \tau$
return $\mathsf{RO}_H(\mathsf{root}, \tau')$

$\mathcal{P}^{\mathsf{RO}}_{\mathcal{B}L,\mathsf{resp}}(\mathsf{seed}, A_I, \mathsf{tree}, \mathsf{ch})$ :
$(B, r_1, \dots, r_R) \leftarrow \mathsf{RO}_E(\mathsf{seed})$
if $\mathsf{ch} = 0$ then
    $D \leftarrow BA_I$
    $\mathsf{path} \leftarrow \mathsf{getMerklePath}(\mathsf{tree}, I)$
    $\mathsf{resp} \leftarrow (D, \mathsf{path}, r_I)$
else
    $\mathsf{resp} \leftarrow \mathsf{seed}$
return $\mathsf{resp}$
```

```
$\mathcal{V}^{\mathsf{RO}}_{\mathcal{B}L}(\tau, \mathsf{com}, \mathsf{ch}, \mathsf{resp})$ :
if $\mathsf{ch} = 0$ then
    $(D, \mathsf{path}, r) \leftarrow \mathsf{resp}$
    $\widetilde{\phi} \leftarrow D \star \phi$
    $\widetilde{\mathsf{root}} \leftarrow \mathsf{ReconstructRoot}(\mathsf{path}, \mathsf{RO}_{\mathrm{Com}}(\widetilde{\phi}, r))$
    $\widetilde{\tau} \leftarrow D \bullet \psi$
else
    $(B, r_1, \dots, r_R) \leftarrow \mathsf{RO}_E(\mathsf{resp})$
    for $i = 1, \dots, R$ do
        $\widetilde{C}_i \leftarrow \mathsf{RO}_{\mathrm{Com}}(B \star \phi_i, r_i)$
    $\widetilde{\tau} \leftarrow B \bullet \tau$
    $(\widetilde{\mathsf{root}}, \widetilde{\mathsf{tree}}) \leftarrow \mathsf{MerkleTree}(\widetilde{C}_1, \dots, \widetilde{C}_R)$
if $\mathsf{com} = \mathsf{RO}_H(\widetilde{\mathsf{root}}, \widetilde{\tau})$ then
    return accept
return reject
```

Figure 5: Algorithms for the base OR Sigma protocol with tag

Starting from the base Sigma protocol with tags, we can reduce the soundness error from $\frac{1}{2}$ to $\frac{1}{2^\lambda}$, for a security parameter λ, performing parallel repetitions of the protocol. We adopt the same optimisations used for the main Sigma protocol, obtaining the algorithms $\mathcal{P}_{\mathcal{M}L,\mathsf{com}}$,$\mathcal{P}_{\mathcal{M}L,\mathsf{resp}}$ and $\mathcal{V}_{\mathcal{M}L}$. We do not report the full protocol here since it is straightforward.

Observe that both the base and the main OR sigma protocol with tag share the same security properties of Theorem 19 and Theorem 23. The proofs can be easily adapted from the ones given in Appendix A, having care of the tag τ.

5 The (Linkable) Ring Signature Scheme

Given the main OR sigma protocol of the previous section, we apply the Fiat-Shamir transform to achieve a ring signature. We observe that our sigma protocol is *commitment reproducible* if we add the salt used by the prover. We report the algorithms RecCom to recover the commitment, both for the main OR sigma protocol and for the main OR sigma protocol with tags, in Appendix B, Figure 8. In this way, the signing algorithm Sign returns the challenge and the response, reducing the size of the signature. The scheme uses an hash function $\mathcal{H}_{\text{FS}}$ with digest in the challenge space $\{0, \ldots, \binom{M}{K} - 1\}$, modelled by a Random Oracle. We report the (non-linkable) ring signature scheme TRIFORS in Figure 6. The algorithm Setup sets the public parameters accordingly to the analysis of Section 7; moreover, alternating trilinear forms ϕ and ψ are sampled at random from $\text{ATF}(n, q)$.

```
Setup(1^λ) :
    pp ← $(q, n, M, K, \phi)$
    return pp

KGen(pp) :
    $A \leftarrow\!\$\ \text{GL}_n(q)$
    sk ← $A$
    pk ← $A \star \phi$
    return (sk, pk)
```

```
Sign (sk_I, msg, {pk_1, ..., pk_R}) :
    com ← $\mathcal{P}_{\mathcal{M}}{}^{\mathsf{RO}}_{\mathsf{com}}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\})$
    $(\mathsf{salt}, (\mathsf{com}_i)_{i=1,\ldots,M}) \leftarrow \mathsf{com}$
    $J_{\mathsf{ch}} \leftarrow \mathcal{H}_{\text{FS}}(\mathsf{msg}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \mathsf{com})$
    ch ← Unrank($J_{\mathsf{ch}}$)
    resp ← $\mathcal{P}_{\mathcal{M}}{}^{\mathsf{RO}}_{\mathsf{resp}}(\mathsf{sk}_I, \mathsf{ch})$
    return $\sigma \leftarrow (\mathsf{salt}, J_{\mathsf{ch}}, \mathsf{resp})$

Verify ({pk_1, ..., pk_R}, msg, σ) :
    $(\mathsf{salt}, J_{\mathsf{ch}}, \mathsf{resp}) \leftarrow \sigma$
    com ← RecCom (salt, {pk_1, ..., pk_R}, $J_{\mathsf{ch}}$, resp)
    $J' \leftarrow \mathcal{H}_{\text{FS}}(\mathsf{msg}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \mathsf{com})$
    return $J' = J_{\mathsf{ch}} \wedge \mathcal{V}_{\mathcal{M}}{}^{\mathsf{RO}}(\mathsf{com}, J_{\mathsf{ch}}, \mathsf{resp})$
```

Figure 6: TRIFORS algorithms.

Using standard techniques, we can prove the following result on the security of TRIFORS.

Theorem 26. *The ring signature TRIFORS from Figure 6 is correct, unforgeable and non-frameable in the Random Oracle Model under the assumption that R-sATFE is intractable.*

We can use the same techniques to obtain a linkable ring signature from the main OR sigma protocol with tags. The construction is the same as the non-linkable scheme, this time using the main OR sigma protocol with tags of Subsection 4.5. We call it Link-TRIFORS and it is reported in Figure 7.

Theorem 27. *The linkable ring signature Link-TRIFORS from Figure 7 is correct, linkable, linkable anonymous and non-frameable in the Random Oracle Model under the assumption that both R-sATFE and slATFE are intractable.*

The proof of the above theorem is quite common in the literature and is a slight modification of the one given in [5].

6 Solving sATFE to Attack the Schemes

We distinguish two scenarios: attacking the ring signature scheme and attacking the linkable ring signature scheme. Forging the non-linkable ring signature can be reduced to attacking the construction from [28], while the linkable version involves additional information regarding the tag τ.

6.1 Attacks to the ring signature TRIFORS

A possible approach to solve a generic instance (ϕ, ψ) of sATFE is solving the following polynomial system

$$\begin{cases} XY = I_n \\ \phi(Xu, Xv, w) = \phi_I(u, v, Yw). \end{cases} \tag{4}$$

```
$\underline{\mathsf{Setup}_L(1^\lambda)}$ :
$\mathsf{pp} \leftarrow (q, n, M, K, \phi, \psi)$
return $\mathsf{pp}$

$\underline{\mathsf{KGen}_L(\mathsf{pp})}$ :
$A \leftarrow\!\$\ \mathrm{GL}_n(q)$
$\mathsf{sk} \leftarrow A$
$\mathsf{pk} \leftarrow A \star \phi$
return $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{Sign}_L\,(\mathsf{sk}_I, \mathsf{msg}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\})}$ :
$\tau \leftarrow \mathsf{sk}_I \bullet \psi$
$\mathsf{com} \leftarrow \mathcal{P}_{\mathcal{M}}{}^{\mathsf{RO}}_{L,\mathsf{com}}(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \tau)$
$(\mathsf{salt}, (\mathsf{com}_i)_{i=1,\ldots,M}) \leftarrow \mathsf{com}$
$J_{\mathsf{ch}} \leftarrow \mathcal{H}_{\mathrm{FS}}(\mathsf{msg}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \tau, \mathsf{com})$
$\mathsf{ch} \leftarrow \mathsf{Unrank}(J_{\mathsf{ch})}$
$\mathsf{resp} \leftarrow \mathcal{P}_{\mathcal{M}}{}^{\mathsf{RO}}_{L,\mathsf{resp}}(\mathsf{sk}_I, \mathsf{ch})$
return $\sigma \leftarrow (\mathsf{salt}, \tau, J_{\mathsf{ch}}, \mathsf{resp})$
```

```
$\underline{\mathsf{Link}_L\,(\sigma_1, \sigma_2)}$ :
$(\mathsf{salt}_1, \tau_1, \mathsf{ch}_1, \mathsf{resp}_1) \leftarrow \sigma_1$
$(\mathsf{salt}_2, \tau_2, \mathsf{ch}_2, \mathsf{resp}_2) \leftarrow \sigma_2$
if $\tau_1 = \tau_2$ then
  return true
return false

$\underline{\mathsf{Verify}_L\,(\{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \mathsf{msg}, \sigma)}$ :
$(\mathsf{salt}, J_{\mathsf{ch}}, \mathsf{resp}) \leftarrow \sigma$
$\mathsf{com} \leftarrow \mathsf{RecCom}_L\,(\mathsf{salt}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \tau, J_{\mathsf{ch}}, \mathsf{resp})$
$J' \leftarrow \mathcal{H}_{\mathrm{FS}}\,(\mathsf{msg}, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_R\}, \tau, \mathsf{com})$
return $J' = J_{\mathsf{ch}} \wedge \mathcal{V}_{\mathcal{M}}{}^{\mathsf{RO}}_{L}(\tau, \mathsf{com}, J_{\mathsf{ch}}, \mathsf{resp})$
```

Figure 7: Link-TRIFORS algorithms.

Here, the public key A is represented by variables X, imposed to be invertible with inverse Y, while the second row models how the matrix A acts on the form ϕ. This leads to a system of $n^2 + \binom{3}{n}$ quadratic equations in $2n^2$ variables.

In [28] is showed an estimation of the complexity of the F5 algorithm for the above system. They obtain an upper bound of $O(2^{6\omega n \log_2 n})$, where ω is the matrix multiplication exponent, taken equal to 2 for conservative reasons.

From [28, Sect. 5.2], we recall the heuristic attack *Grobner basis with partial information* that solves sATFE in time

$$O(q^{2n/3} \cdot n^{2\omega} \cdot \log_2(q)). \tag{5}$$

A collision finding approach is used to find partial information, based on the birthday attack. The "partial information" means that a column of A, and hence of X, is known. This implies constrains also on variables in Y, leading to a system of polynomials in $2(n^2 - n)$ variables. Some experiments show that this new system is solvable in polynomial time but finding the partial information needed is still an exponential task, leading to the overall complexity given in Eq. (5). This is the best-known attack to the problem and we will select parameters according to it. We select n and q such that

- $2^\lambda \leq q^{2/3} \cdot n^{2\omega} \cdot \log_2(q)$, from Eq (5), and
- $q^{2/3} \leq n^{12}$, from the F5 algorithm analysis.

6.2 Attacks to the linkable ring signature Link-TRIFORS

In the case of the linkable scheme, the use of the action $\bullet$ gives more information about the secret key A. We can use the same approach of the previous section, building the system

$$\begin{cases} XY = I_n \\ \phi(Xu, Xv, w) = \phi_I(u, v, Yw) \\ \psi(Yu, Yv, w) = \tau(u, v, Xw) \end{cases} \tag{6}$$

where τ is the tag and is given by the action of the inverse of A, that is modelled by variables Y. This system has $n^2 + 2\binom{3}{n}$ quadratic equations in $2n^2$ variables.

Using the estimation of the degree of regularity, we have that it is asymptotically $\frac{3}{2}n$. Hence the F5 algorithm runs in time at most

$$O\left((2n^2)^{\omega n 3/2}\right) = O\left(n^{3\omega n}\right)$$

The number of equations is less than the double of the case without linkability and the analysis done before can be adapted here, since the collision argument from [28] does not involve the number of equations and is only based on the secret matrix A.

We can conclude that, even in this case, the best-known algorithm attacking the scheme runs in time

$$O(q^{2n/3} \cdot n^{2\omega} \cdot \log_2(q)).$$

Furthermore, additional information on the secret key, in the form of how its inverse acts on the trilinear form ϕ, could be crucial to devising an efficient attack: this fact requires further analysis in the future. For the linkable scheme, we select n and q such that

- $2^\lambda \leq q^{2/3} \cdot n^{2\omega} \cdot \log_2(q)$, from Eq (5), and
- $q^{2/3} \leq n^6$, from the F5 algorithm analysis.

7 Parameters and Conclusions

Signature length and parameters. Given a security parameter λ, we want to find parameters n, q, M and K that minimise the signature size. Using the analysis done in Section 6, we want that the best known attack to sATFE has a running time greater than 2^λ. We can also refer to the analysis reported in [28] for the choice of parameters q and n.

The size in bits of the (linkable) ring signature, with respect to parameters n, q, M and K is given by the following values:

- the secret key sk, an invertible matrix A with coefficients in $\mathbb{F}_q$ represented by $n^2\lceil\log_2 q\rceil$ bits;
- the public key pk, an alternating trilinear form which can hence be stored with $\binom{n}{3}\lceil\log_2 q\rceil$ bits;
- the signature length, given by

$$|\mathsf{salt}| + |\mathrm{tag}| + |\mathsf{ch}| + K\left|\mathsf{resp}_{\mathsf{ch}_i=0}\right| + \left|\mathsf{resp}_{\mathsf{ch}_i=1}\right|.$$

We have that:

- the length of the salt is taken as the double of λ: $|\mathsf{salt}| = 2\lambda$;
- a tag is an alternating trilinear form: $|\mathrm{tag}| = \binom{n}{3}\lceil\log_2 q\rceil$;
- the challenge is a positive integer smaller than $\binom{M}{K}$: $|\mathsf{ch}| = \left\lceil\log_2 \binom{M}{K}\right\rceil$;
- whenever the challenge is 0, the response contains an invertible $n \times n$ matrix D, a path in the Merkle tree with R leaves, where R is the size of the ring, and λ random bits r, hence we have

$$\left|\mathsf{resp}_{\mathsf{ch}_i=0}\right| = |D| + |\mathsf{path}| + |r| = n^2 + 2\lambda\lceil\log_2 R\rceil + \lambda;$$

- whenever the challenge is 1, the response is equal to the set of internal nodes of the seed tree needed to obtain the associated leaves. The number of such nodes is studied in Subsection 4.2 and we can follow different approaches: minimising the average, the best case or the worst case. In our tests, the best choice of the parameters is not affected by which approach has been chosen. For this reason, we report here the worst case:

$$\left|\mathsf{resp}_{\mathsf{ch}_i=1}\right| = K\lceil\log_2 M\rceil - K\lceil\log_2 K\rceil + 2^{\lceil\log_2 K\rceil} - K.$$

Hence, the (non-linkable) signature has a bit length of

$$2\lambda + \left\lceil \log_2 \binom{M}{K} \right\rceil + K(n^2 + 2\lambda\lceil\log_2 R\rceil + \lambda) + K\lceil\log_2 M\rceil - K\lceil\log_2 K\rceil + 2^{\lceil\log_2 K\rceil} - K. \quad (7)$$

Observe that in the case of a linkable ring signature we add the size of a tag $\binom{n}{3}\lceil\log_2 q\rceil$ in the above equation.

Given a security parameter λ, values for n, q, M and K minimizing Eq. (7) have been found such that they match the security required. Since parameters M and K contribute as $\left\lceil\log_2 \binom{M}{K}\right\rceil$ ($\sim \lambda$) and only K is involved linearly in Eq. (7), the number of repetitions M is selected to be not too high, to avoid a slowdown in the performance.

Proposed parameters for $\lambda = 128, 192$ are reported in Table 3, together with the signature length in kilobytes for different ring sizes R. Observe that the sizes refer to the non-linkable ring signature scheme: if the linkability is needed, the tag must be included in the signature adding $\binom{n}{3}\lceil\log_2 q\rceil$ bits.

λ	q	n	M	K	R				
					2^1	2^3	2^6	2^{12}	2^{21}
128	$2^{19} - 1$	9	512	23	7.4 ± 1.7	8.8 ± 1.7	11.0 ± 1.7	15.4 ± 1.7	22.1 ± 1.7
192	$2^{27} - 39$	10	1024	31	16.6 ± 3.7	19.5 ± 3.7	24 ± 3.7	32.9 ± 3.7	46.3 ± 3.7

Table 3: Parameters and signature sizes in KB of the non-linkable ring signature. R is the size of the ring.

Conclusions and future work. In this paper we have described TRIFORS, a logarithmic post-quantum (linkable) ring signature based on the assumption that the sATFE problem is intractable. To obtain the digital signature, we followed the construction of Beullens, Katsumata and Pintore [5]: starting from the base OR sigma protocol, we first modified it to reduce the soundness error from $\frac{1}{2}$ to $\frac{1}{2^\lambda}$, where λ is the security parameter, obtaining the so called main OR sigma protocol, and finally we applied the Fiat-Shamir transform to obtain the ring signature TRIFORS. We also modified the base OR sigma protocol, adding a tag that is always the same when using the same private key, and, by repeating the same steps above, we obtained the linkable ring signature Link-TRIFORS.

The length of the signature produced by TRIFORS is logarithmic with respect to ring size and is competitive with the state-of-the-art. More precisely, having fixed the security parameter λ, we calculated the optimal M and K parameters to minimise the signature length. These parameters can be found in Table 3. For example, for $\lambda = 128$ our construction is competitive with the state-of-the-art of post-quantum ring signature: only *Calamari* [5] obtains smaller signatures, but the price to pay is the less practical time required to compute isogenies. For huge rings, our signature turns out to be the shortest one. This result is also due to an additional optimisation we have introduced, in fact we use the combinatorial number system on the space of the challenges to further reduce the length of the signature.

As future work, we plan to implement this signature to calculate how long it takes on average to sign a message, varying certain parameters such as the size of the ring or the number of zeros in the challenge. Furthermore, one must not forget that the assumptions on which the entire work is based is very recent, and further cryptanalysis is necessary to be convinced of the security of the signature.

References

[1] Alamati, N., Feo, L.D., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 411–439. Springer (2020)

[2] Barenghi, A., Biasse, J.F., Ngo, T., Persichetti, E., Santini, P.: Advanced signature functionalities from the code equivalence problem. International Journal of Computer Mathematics: Computer Systems Theory **7**(2), 112–128 (2022)

[3] Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: Less-fm: fine-tuning signatures from the code equivalence problem. In: International Conference on Post-Quantum Cryptography, pp. 23–43. Springer (2021)

[4] Bellini, E., Esser, A., Sanna, C., Verbel, J.: Mr-dss–smaller minrank-based (ring-) signatures. Cryptology ePrint Archive (2022)

[5] Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 464–492. Springer (2020)

[6] Beullens, W., Kleinjung, T., Vercauteren, F.: Csi-fish: efficient isogeny based signatures through class group computations. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 227–247. Springer (2019)

[7] Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: Csidh: an efficient post-quantum commutative group action. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 395–427. Springer (2018)

[8] Chaum, D., van Heyst, E.: Group signatures. In: Workshop on the Theory and Application of of Cryptographic Techniques, pp. 257–265. Springer (1991)

[9] Chistikov, D., Iván, S., Lubiw, A., Shallit, J.: Fractional coverings, greedy coverings, and rectifier networks. In: 34th Symposium on Theoretical Aspects of Computer Science (2017)

[10] De Feo, L., Galbraith, S.D.: Seasign: compact isogeny signatures from class group actions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 759–789. Springer (2019)

[11] Esgin, M.F., Zhao, R.K., Steinfeld, R., Liu, J.K., Liu, D.: Matrict: efficient, scalable and post-quantum blockchain confidential transactions protocol. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 567–584 (2019)

[12] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the theory and application of cryptographic techniques, pp. 186–194. Springer (1986)

[13] Goodell, B., Noether, S., Blue, A.: Concise linkable ring signatures and forgery against adversarial keys. Cryptology ePrint Archive (2019)

[14] Grochow, J.A., Qiao, Y.: On the complexity of isomorphism problems for tensors, groups, and polynomials i: Tensor isomorphism-completeness. In: 12th Innovations in Theoretical Computer Science Conference (ITCS 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2021)

[15] Gueron, S., Persichetti, E., Santini, P.: Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. Cryptography **6**(1), 5 (2022)

[16] Ji, Z., Qiao, Y., Song, F., Yun, A.: General linear group action on tensors: a candidate for post-quantum cryptography. In: Theory of Cryptography Conference, pp. 251–281. Springer (2019)

[17] Knuth, D.E.: Generating all combinations and partitions, volume 4, fascicle 3 of the art of computer programming (2005)

[18] Liu, J.L., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for ad hoc groups. In: Australasian Conference on Information Security and Privacy, pp. 325 – 335. Springer (2004)

[19] Lu, X., Au, M.H., Zhang, Z.: Raptor: a practical lattice-based (linkable) ring signature. In: International Conference on Applied Cryptography and Network Security, pp. 110–130. IEEE (2019)

[20] Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Smile: set membership from ideal lattices with applications to ring signatures and confidential transactions. In: Annual International Cryptology Conference, pp. 611–640. Springer (2021)

[21] Merkle, R.C.: A digital signature based on a conventional encryption function. In: Conference on the theory and application of cryptographic techniques, pp. 369–378. Springer (1987)

[22] Noether, S., Mackenzie, A., et al.: Ring confidential transactions. Ledger **1**, 1–18 (2016)

[23] OEIS Foundation Inc.: The On-Line Encyclopedia of Integer Sequences (2022). Published electronically at `http://oeis.org`

[24] Park, E., Blake, I.F.: On the mean number of encryptions for tree-based broadcast encryption schemes. Journal of Discrete Algorithms **4**(2), 215–238 (2006)

[25] Perera, M.N.S., Nakamura, T., Hashimoto, M., Yokoyama, H., Cheng, C.M., Sakurai, K.: A survey on group signatures and ring signatures: Traceability vs. anonymity. Cryptography **6**(1), 3 (2022)

[26] Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 552–565. Springer (2001)

[27] Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science, pp. 124–134. Ieee (1994)

[28] Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 582–612. Springer (2022)

[29] Yuen, T.H., Esgin, M.F., Liu, J.K., Au, M.H., Ding, Z.: Dualring: Generic construction of ring signatures with efficient instantiations. In: Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I, pp. 251–281 (2021)

A Security Proofs

A.1 Proof of Theorem 19

Proof.

- **Correctness.** Given a pair $((\phi_1, \dots, \phi_R), A_I)$ such that $\phi_I = A_I \star \phi$, if the protocol is executed honestly, the verifier accepts with probability 1 by construction. If $\mathsf{ch} = 0$, the verifier computes

$$D \star \phi = BA_I \star \phi = B \star \phi_I$$

and uses it to reconstruct the root of the Merkle tree using the path given in the response resp. When $\mathsf{ch} = 1$, the response is the seed used by the prover, and the verifier repeats the same computations of the prover getting the root of the Merkle tree.

- **Special 2-soundness.** Given two transcripts $(\mathsf{root}, 0, (D, \mathsf{path}, r_I))$ and $(\mathsf{root}, 1, \mathsf{seed})$, the extractor $\mathcal{E}$ acts as follows. It expands the seed using RO_E to obtain the random matrix B, and use it to compute the secret key $A_I = B^{-1}D$.

- **Special zero-knowledge.** We want to show that there exists a simulator $\mathcal{S}$ such that, for any $((\phi_1, \dots, \phi_R), A_I)$ with $\phi_I = A_I \star \phi$, for any $\mathsf{ch} = 0, 1$, and for any adversary $\mathcal{A}$ making at most a polynomial number of queries Q to the random oracle, we have that

$$\left| \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}({\mathcal{P}_{\mathcal{B}}}^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, \mathsf{ch}) = 1 \right] - \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}((\phi_1, \dots, \phi_R), \mathsf{ch})) = 1 \right] \right| \tag{8}$$

is negligible in λ. The simulator $\mathcal{S}$ is defined as follows:

  - $\mathsf{ch} = 1$: it runs the prover and outputs a valid transcript, since in this case the witness A_I is not used in the computation, the response consists of seed.
  - $\mathsf{ch} = 0$: it picks a random invertible matrix D and a random string r of λ bits, then it computes the commitment $C_1 = \mathsf{RO}_{\mathrm{Com}}(D \star \phi, r)$. The simulator randomly generates $R-1$ dummy commitments $C_2, \dots, C_R$ in $\{0,1\}^{2\lambda}$ and creates the Merkle tree $(\mathsf{root}, \mathsf{tree}) = \mathsf{MerkleTree}(C_1, \dots, C_R)$. Finally, it outputs $(\mathsf{root}, 0, (D, \mathsf{path}, r))$.

We prove that Eq. (8) is at most $\frac{2Q}{2^\lambda}$, where Q is the number of queries of $\mathcal{A}$ to the random oracle. In order to prove the above statement, we introduce a sequence of simulators $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$, where $\mathcal{S}_1 = \mathcal{P}_{\mathcal{B}}$, $\mathcal{S}_4 = \mathcal{S}$ and the other are defined below. Fix a pair $((\phi_1, \dots, \phi_R), A_I)$ with $\phi_I = A_I \star \phi$ and an adversary $\mathcal{A}$. The case $\mathsf{ch} = 1$ is straightforward, since the witness A_I is not used in the response, then, for $\mathsf{ch} = 0$, Eq. (8) becomes

$$\left| \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_1^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, 0) = 1 \right] - \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}^{\mathsf{RO}}((\phi_1, \dots, \phi_R), 0) = 1 \right] \right|.$$

The second simulator $\mathcal{S}_2$ behaves like both algorithms of the prover $\mathcal{P}_{\mathcal{B}}$, except that, instead of expanding the seed with the random oracle $\mathsf{RO}_E(\mathsf{seed})$, it picks B and $r_1, \dots, r_R$ uniformly at random. This does not change the view of $\mathcal{A}$ unless it queries to the oracle the same input seed in $\{0,1\}^\lambda$. This happens with probability at most $\frac{Q}{2^\lambda}$ and we have

$$\left| \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_1^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, 0) = 1 \right] - \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_2^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, 0) = 1 \right] \right| \leq \frac{Q}{2^\lambda}.$$

The thirds simulator $\mathcal{S}_3$ acts the same as $\mathcal{S}_2$, except that commitments C_i, for $i \neq I$, are chosen uniformly at random. The adversary $\mathcal{A}$ does not notice this unless it queries $\mathsf{RO}_{\mathrm{Com}}$ on input (ψ_i, r_i), where $\psi_i = B \star \phi_i$, for $i \neq I$. Let Q_{ψ_i} be the number of queries of the form $\mathsf{RO}_{\mathrm{Com}}(\psi_i, \cdot)$, since r_i has λ bits of min-entropy, the probability that $\mathcal{A}$ asks $\mathsf{RO}_{\mathrm{Com}}$ on input (ψ_i, r_i) is at most $\frac{Q_{\psi_i}}{2^\lambda}$. Without loss of generality we can assume that all the public keys $\{\phi_1, \dots, \phi_R\}$ are distinct, and so are $\{\psi_1, \dots, \psi_R\}$. This implies that $\sum_{i=1}^R \frac{Q_{\psi_i}}{2^\lambda} \leq \frac{Q}{2^\lambda}$ and

$$\left| \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_2^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, 0) = 1 \right] - \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_3^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, 0) = 1 \right] \right| \leq \frac{Q}{2^\lambda}.$$

The fourth simulator $\mathcal{S}_4$ is the same as $\mathcal{S}_3$ but instead of computing $\psi_I = B \star \phi_I$, it picks a uniformly random invertible matrix B' and sets $\psi_I = B' \star \phi_I$. Moreover it uses $I = 1$ instead of the value of I given in the witness. From [5, Lemma 2.10], we have that the index-hiding property of the Merkle trees used in the scheme does not change the view of the adversary $\mathcal{A}$ and we have

$$\left| \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_3^{\mathsf{RO}}((\phi_1, \dots, \phi_R), A_I, 0) = 1 \right] - \mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_4^{\mathsf{RO}}((\phi_1, \dots, \phi_R), 0) = 1 \right] \right| = 0.$$

Combining all the results gives us

$$\left|\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{P_B}^{\mathsf{RO}}((\phi_1,\dots,\phi_R),A_I,\mathsf{ch})=1\right]-\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}^{\mathsf{RO}}((\phi_1,\dots,\phi_R),\mathsf{ch})=1\right]\right|\leq\frac{2Q}{2^\lambda}$$

and the thesis is proven since Q is polynomial in λ and hence $\frac{2Q}{2^\lambda}$ is negligible.

□

A.2 Proof of Theorem 23

Proof.

- **Correctness.** Since the main OR sigma protocol is a parallel repetition of the base OR sigma protocol with some optimisations, the correctness is implied by Theorem 19 and by the correctness of the algorithms of the seed trees.
- **High min-entropy.** Since the commitment com depends on a sandom salt of 2λ bits and on a λ bits seed, the scheme has high min-entropy.
- **Special 2-soundness.** Let $(\mathsf{com},\mathsf{ch},\mathsf{resp})$ and $(\mathsf{com},\mathsf{ch}',\mathsf{resp}')$ be two accepting transcripts, where $\mathsf{com}=(\mathsf{salt},(\mathsf{root}^{(i)})_{i=1,\dots,M})$ and $\mathsf{ch}\neq\mathsf{ch}'$. We define the extractor $\mathcal{E}$ as follows. Since $\mathsf{ch}\neq\mathsf{ch}'$, there exists j such that the j-th bits of ch and ch' are different. Without loss of generality let $\mathsf{ch}_j=0$ and $\mathsf{ch}'_j=1$. By construction, we have $\mathsf{resp}=(\mathsf{seeds}_{\text{int}},(\mathsf{resp}^{(i)})_{\mathsf{ch}_i=0}))$, with $\mathsf{resp}^{(j)}=(D^{(j)},\mathsf{path}^{(j)},r_I^{(j)})$. Moreover, $\mathsf{resp}'=(\mathsf{seeds}'_{\text{int}},(\mathsf{resp}^{(i)})_{\mathsf{ch}'_i=0}))$ and from $\mathsf{seeds}'_{\text{int}}$, the extractor $\mathcal{E}$ retrieves the seed for the j-th parallel execution of the protocol, computing $\mathsf{seed}^{(j)}=\mathsf{RecoverLeaves}(\mathsf{seeds}'_{\text{int}},\mathsf{ch}')$. In this way, if we proceed as in the proof of Theorem 19, the extractor can retrieve the secret key A_I.
- **Special zero-knowledge.** We want to show that there exists a simulator $\mathcal{S}$ such that, for any $((\phi_1,\dots,\phi_R),A_I)$ with $\phi_I=A_I\star\phi$, for any ch, and for any adversary $\mathcal{A}$ making at most a polynomial number of queries Q to the "salted" random oracle (the oracle having as prefix the string salt given in the transcript), we have

$$\left|\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{P_M}^{\mathsf{RO}}((\phi_1,\dots,\phi_R),A_I,\mathsf{ch})=1\right]-\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}((\phi_1,\dots,\phi_R),\mathsf{ch}))=1\right]\right|\tag{9}$$

is negligible in λ. The simulator $\mathcal{S}$ is defined as follows:

  - it generates at random the 2λ bit string salt. Then it computes $\mathsf{seeds}_{\text{int}}$ calling $\mathsf{SimulateSeeds}(\mathsf{ch})$, and then $(\mathsf{seed}^{(i)})_{\mathsf{ch}_i=1}$ are given by $\mathsf{RecoverLeaves}(\mathsf{seeds}_{\text{int}},\mathsf{ch})$.
  - For $\mathsf{ch}_i=0$ the simulator behaves as the simulator of the base OR sigma protocol from the proof of Theorem 19, outputting $(\mathsf{com}^{(i)},0,\mathsf{resp}^{(i)})$.
  - For $\mathsf{ch}_i=1$, the simulator $\mathcal{S}$ computes the root of the Merkle tree $\mathsf{root}^{(i)}$ from $\mathsf{seed}^{(i)}$.
  - Then $\mathcal{S}$ outputs the transcript

$$((\mathsf{salt},(\mathsf{com}_i)_{i=1,\dots,M}),\mathsf{ch},(\mathsf{seeds}_{\text{int}},(\mathsf{resp}_i)_{\mathsf{ch}_i=0})).$$

We prove that Eq. (9) is at most $\frac{3Q}{2^\lambda}$, introducing a sequence of simulators $\mathcal{S}_1=\mathcal{P_M},\mathcal{S}_2,\mathcal{S}_3=\mathcal{S}$. Fix a pair $((\phi_1,\dots,\phi_R),A_I)$ with $\phi_I=A_I\star\phi$ and an adversary $\mathcal{A}$.
The second simulator $\mathcal{S}_2$ behaves like the prover $\mathcal{P_M}$, except the way it generates $\mathsf{seed}^{(i)}$ for $i=1,\dots,M$. The simulator runs $\mathsf{SimulateSeeds}$ and $\mathsf{RecoverLeaves}$ as explained above, hence it determines $(\mathsf{seed}_i)_{\mathsf{ch}_i=1}$. Then it picks uniformly at random the remaining seeds $(\mathsf{seed}_i)_{\mathsf{ch}_i=1}$. Using [5, Lemma 2.11] we obtain that

$$\left|\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_1^{\mathsf{RO}}((\phi_1,\dots,\phi_R),A_I,\mathsf{ch})=1\right]-\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_2^{\mathsf{RO}}((\phi_1,\dots,\phi_R),A_I,\mathsf{ch})=1\right]\right|\leq\frac{Q}{2^\lambda}.$$

The thirds simulator $\mathcal{S}_3$ acts the same as $\mathcal{S}_2$, except that uses the simulator for the base OR sigma protocol to compute $\mathsf{com}^{(i)}$ and $\mathsf{resp}^{(i)}$ for $\mathsf{ch}_i=0$. Using the zero-knowledge property of the latter, we have that the advantage of any adversary $\mathcal{A}$ for distinguishing the simulator from a honest prover is at most $\frac{2Q_i}{2^\lambda}$, where Q_i is the number of queries of the form $\mathsf{RO}(\mathsf{salt}||i||\cdot)$ that $\mathcal{A}$ makes to the random oracle. Hence we have

$$\left|\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_2^{\mathsf{RO}}((\phi_1,\dots,\phi_R),A_I,\mathsf{ch})=1\right]-\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}_3^{\mathsf{RO}}((\phi_1,\dots,\phi_R),\mathsf{ch})=1\right]\right|\leq\sum_{\substack{i\text{ s.t.}\\ \mathsf{ch}_i=0}}\frac{2Q_i}{2^\lambda},$$

and moreover, $\sum_{\substack{i \text{ s.t.} \\ \mathsf{ch}_i=0}} \frac{2Q_i}{2^\lambda} \leq \frac{2Q}{2^\lambda}$.
Combining these results gives us

$$\left|\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{P_B}^{\mathsf{RO}}((\phi_1,\dots,\phi_R),A_I,\mathsf{ch})=1\right]-\mathbf{P}\left[\mathcal{A}^{\mathsf{RO}}(\mathcal{S}^{\mathsf{RO}}((\phi_1,\dots,\phi_R),\mathsf{ch})=1\right]\right| \leq \frac{3Q}{2^\lambda}$$

and the thesis is proven since Q is polynomial in λ and hence $\frac{3Q}{2^\lambda}$ is negligible.

□

B Commitment Reproducibility

Here we show the algorithms for the commitment reproducibility. A sigma protocol is *commitment reproducible* if, given an accepting transcript $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, there exists a polynomial time algorithm RecCom such that, on input ch and resp, returns the commitment com with overwhelming probability. In Figure 8 we present the algorithms for both the main OR sigma protocol and the version with tag.

```
$\mathsf{RecCom}\,(\mathsf{salt}, \{\mathsf{pk}_1,\dots,\mathsf{pk}_R\}, J_{\mathsf{ch}}, \mathsf{resp})$ :
$\{\phi_1,\dots,\phi_R\} \leftarrow \{\mathsf{pk}_1,\dots,\mathsf{pk}_R\}$
$\mathsf{RO}' \leftarrow \mathsf{RO}(\mathsf{salt}||\cdot)$
$\mathsf{ch} \leftarrow \mathsf{Unrank}(J_{\mathsf{ch}})$
$(\mathsf{seeds}_{\mathrm{int}}, (\mathsf{resp}^{(i)})_{\mathsf{ch}_i=0}) \leftarrow \mathsf{resp}$
for $i$ s.t. $\mathsf{ch}_i = 0$ then
  $\mathsf{RO}^i \leftarrow \mathsf{RO}'(i||\cdot)$
  $(D^{(i)}, \mathsf{path}^{(i)}, r^{(i)}) \leftarrow \mathsf{resp}^{(i)}$
  $\phi^{(i)} \leftarrow D^{(i)} \star \phi$
  $c^{(i)} \leftarrow \mathsf{RO}_{\mathrm{Com}}(\phi^{(i)}, r^{(i)})$
  $\mathsf{root}^{(i)} \leftarrow \mathsf{ReconstructRoot}(\mathsf{path}^{(i)}, c^{(i)})$
$(\mathsf{seed}^{(i)})_{\mathsf{ch}=1} \leftarrow \mathsf{RecoverLeaves}(\mathsf{seeds}_{\mathrm{int}}, \mathsf{ch})$
for $i$ s.t. $\mathsf{ch}_i = 1$ then
  $\mathsf{RO}^i \leftarrow \mathsf{RO}'(i||\cdot)$
  $(B^{(i)}, r_1^{(i)},\dots,r_R^{(i)}) \leftarrow \mathsf{RO}^i_E(\mathsf{seed}^{(i)})$
  for $j = 1,\dots,R$ do
    $C_j^{(i)} \leftarrow \mathsf{RO}^i_{\mathrm{Com}}(B^{(i)} \star \phi_j, r_j^{(i)})$
  $(\mathsf{root}^{(i)}, \mathsf{tree}^{(i)}) \leftarrow \mathsf{MerkleTree}(C_1^{(i)},\dots,C_R^{(i)})$
$\mathsf{com} \leftarrow (\mathsf{root}^{(i)})_{i=1,\dots,M}$
return $\mathsf{com}$
```

```
$\mathsf{RecCom}_L\,(\mathsf{salt}, \{\mathsf{pk}_1,\dots,\mathsf{pk}_R\}, \tau, J_{\mathsf{ch}}, \mathsf{resp})$ :
$\{\phi_1,\dots,\phi_R\} \leftarrow \{\mathsf{pk}_1,\dots,\mathsf{pk}_R\}$
$\mathsf{RO}' \leftarrow \mathsf{RO}(\mathsf{salt}||\cdot)$
$\mathsf{ch} \leftarrow \mathsf{Unrank}(J_{\mathsf{ch}})$
$(\mathsf{seeds}_{\mathrm{int}}, (\mathsf{resp}^{(i)})_{\mathsf{ch}_i=0}) \leftarrow \mathsf{resp}$
for $i$ s.t. $\mathsf{ch}_i = 0$ then
  $\mathsf{RO}^i \leftarrow \mathsf{RO}'(i||\cdot)$
  $(D^{(i)}, \mathsf{path}^{(i)}, r^{(i)}) \leftarrow \mathsf{resp}^{(i)}$
  $\phi^{(i)} \leftarrow D^{(i)} \star \phi$
  $c^{(i)} \leftarrow \mathsf{RO}_{\mathrm{Com}}(\phi^{(i)}, r^{(i)})$
  $\mathsf{root}^{(i)} \leftarrow \mathsf{ReconstructRoot}(\mathsf{path}^{(i)}, c^{(i)})$
$(\mathsf{seed}^{(i)})_{\mathsf{ch}=1} \leftarrow \mathsf{RecoverLeaves}(\mathsf{seeds}_{\mathrm{int}}, \mathsf{ch})$
for $i$ s.t. $\mathsf{ch}_i = 1$ then
  $\mathsf{RO}^i \leftarrow \mathsf{RO}'(i||\cdot)$
  $(B^{(i)}, r_1^{(i)},\dots,r_R^{(i)}) \leftarrow \mathsf{RO}^i_E(\mathsf{seed}^{(i)})$
  for $j = 1,\dots,R$ do
    $C_j^{(i)} \leftarrow \mathsf{RO}^i_{\mathrm{Com}}(B^{(i)} \star \phi_j, r_j^{(i)})$
  $(\mathsf{root}^{(i)}, \mathsf{tree}^{(i)}) \leftarrow \mathsf{MerkleTree}(C_1^{(i)},\dots,C_R^{(i)})$
$\iota \leftarrow i$ s.t. $\mathsf{ch}_i = 1$
$\tau' \leftarrow B^{(\iota)} \bullet \tau$
$\mathsf{com} \leftarrow (\mathsf{RO}^i_H(\mathsf{root}^{(i)}, \tau'))_{i=1,\dots,M}$
return $\mathsf{com}$
```

Figure 8: Commitment Reproducibility algorithms

C Seed Trees Estimations

We use a seed tree to communicate the seed seed for each repetition of the base sigma protocol having challenge bit $\mathsf{ch} = 1$. Due to Section 4.1, the challenge has a larger number of ones, and this structure allows to reduce the size of the response.

Given a seed tree with M leaves, if we want to send $M-K$ leaves, we transmit at most

$$K\lceil\log_2 M\rceil - K\lceil\log_2 K\rceil + 2^{\lceil\log_2 K\rceil} - K.$$

This fact is given in [15] without a proof, here we prove it using the next technical lemma.

Lemma 28. *Let $b(n)$ be the* binary entropy function *[23, A003314] given by*

$$b(n) = \begin{cases} 0 & \text{if } n = 1 \\ \min_{i=1,\dots,n-1}\{n + b(n-i) + b(i)\} & \text{if } n > 1 \end{cases},$$

then, for each natural number n we have

$$b(n) = n + n\lceil\log_2 n\rceil - 2^{\lceil\log_2 n\rceil}.$$

Proof. Set $a(n) = n + n\lceil\log_2 n\rceil - 2^{\lceil\log_2 n\rceil}$. Using the characterization of $b(n)$ reported in [9, Cor. 5], we can write

$$b(n) = n\lfloor\log_2 n\rfloor + 2n - 2\cdot 2^{\lfloor\log_2 n\rfloor}.$$

Observe that, for any n power of 2, we have $\lceil\log_2 n\rceil = \lfloor\log_2 n\rfloor$, otherwise, if n is not a power of 2, then $\lceil\log_2 n\rceil - \lfloor\log_2 n\rfloor = 1$. This implies that for every n we have $a(n) = b(n)$. □

Now we can show the above claim. Observe that if the number of leaves is not a power of 2, we can add dummy leaves to reach the next power of 2 and complete the binary tree. We can exclude the case $K = 0$ since the number of nodes to be sent is equal to 1.

Proposition 29. *Given a seed tree with $M = 2^c$ leaves, if we want to send $M-K$ leaves, with $K \neq 0$, we transmit at most $K\log_2 M - K\lceil\log_2 K\rceil + 2^{\lceil\log_2 K\rceil} - K$ seeds.*

Proof. Set $f(c,K)$ to be the function counting the maximum number of nodes to be transmitted in a tree with 2^c leaves, hiding K leaves. It is easy to see that

$$f(c,K) = \max_{i=1,\dots,K-1}\{f(c-1,K-i) + f(c-1,i)\} \tag{10}$$

and we proceed by induction on c.
The base step is given by $c = 1$ and it is easy to check. Now let $c > 1$ and suppose

$$f(c',K) = Kc' - K\lceil\log_2 K\rceil + 2^{\lceil\log_2 K\rceil} - K$$

for every $c' < c$ and every $K = 1,\dots,c'-1$. Equation (10) gives us

$$\begin{aligned} f(c,K) = \max_{i=1,\dots,K-1}\{&(K-i)(c-1) - (K-i)\lceil\log_2(K-i)\rceil + 2^{\lceil\log_2(K-i)\rceil} \\ &- (K-i) + i(c-1) - i\lceil\log_2 i\rceil + 2^{\lceil\log_2 i\rceil} - i\} \end{aligned}$$

and rearranging the terms we have

$$f(c,K) = Kc + \max_{i=1,\dots,K-1}\{-2K - (K-i)\lceil\log_2(K-i)\rceil + 2^{\lceil\log_2(K-i)\rceil} - i\lceil\log_2 i\rceil + 2^{\lceil\log_2 i\rceil}\}.$$

Now we can take the minimum changing the signs in the parentheses

$$f(c,K) = Kc - \min_{i=1,\dots,K-1}\{2K + (K-i)\lceil\log_2(K-i)\rceil - 2^{\lceil\log_2(K-i)\rceil} + i\lceil\log_2 i\rceil - 2^{\lceil\log_2 i\rceil}\}$$

and setting $a(n) = n + n\lceil\log_2 n\rceil - 2^{\lceil\log_2 n\rceil}$ we obtain

$$f(c,K) = Kc - \min_{i=1,\dots,K-1}\{K + a(K-i) + a(i)\}.$$

Using Lemma 28, we have that

$$f(c,K) = Kc - K\lceil\log_2 K\rceil + 2^{\lceil\log_2 K\rceil} - K.$$

□

To give a more accurate estimate of the signature length, we study the minimal number of nodes to send for a tree with M leaves, if we want to transmit $M-K$ leaves. For any non-negative integer K, we denote with $\mathbf{b}(K)$ its binary representation and with $\mathrm{w}(\mathbf{b}(K))$ the number of ones in $\mathbf{b}(K)$.

Proposition 30. *Given a seed tree with $M = 2^c$ leaves, if we want to send $M - K$ leaves, with $K \neq 0$, we transmit at least $c - \mathrm{w}(\mathbf{b}(K-1))$ seeds.*

Proof. Suppose to transmit $M - K$ leaves in a seed tree with $M = 2^c$ leaves. Denote with $g(c, K)$ the smallest number of nodes to send, depending on the position of the K leaves that should kept secret. Given K leaves to hide, the best scenario is when they are close to each other as much as possible. Without loss of generality we can assume that these K leaves are all on the right. The height of the largest subtree having only leaves to keep secret is $r = \lfloor \log_2 K \rfloor$. If we examine the tree vertically, from the root to the leaves, we send a node for each level from $c-1$ to $r+2$ plus $g(r, K - 2^r)$, the minimal number of node to send having a tree of height r with $K - 2^r$ leaves to hide. A graphic example is reported in Figure 9: we send a node for each level from $c-1$ to $r+2$, without sending the node at level $r+1$. Then we iterate on the small sub-tree on the left.

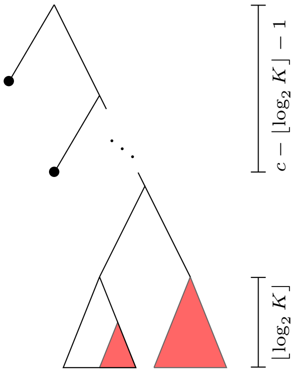

$c - \lfloor \log_2 K \rfloor - 1$
$\lfloor \log_2 K \rfloor$

Figure 9: Red sub-trees contains only leaves to hide. Black circles are nodes that we send.

Considering the base case $g(c, 0) = 1$, where we can send the root, we obtain the following equation:

$$g(c,K) = \begin{cases} 1 & \text{if } K = 0 \\ c - \lfloor \log_2 K \rfloor - 1 + g\left(\lfloor \log_2 K \rfloor, K - 2^{\lfloor \log_2 K \rfloor}\right) & \text{if } K > 0 \end{cases} . \tag{11}$$

For $K > 0$ we can write $g(c, K) = c - d(K)$, where $d(K) = \lfloor \log_2 K \rfloor + 1 - g\left(\lfloor \log_2 K \rfloor, K - 2^{\lfloor \log_2 K \rfloor}\right)$. Let $\mathbf{b}(K) = \left(b_0, \ldots, b_{\lfloor \log_2 K \rfloor}\right)$ be the binary expansion of $K = \sum_{i=0}^{\lfloor \log_2 K \rfloor} b_i 2^i$ and let $a_1 < \cdots < a_t$ be its support. We have that $t = \mathrm{w}(\mathbf{b}(K))$ and $\lfloor \log_2 K \rfloor = a_t$. Then K can be written as $K = \sum_{i=1}^{t} 2^{a_i}$. We claim that $d(K) = a_1 + t - 1$. To show this, we define

$$g_j = g\left(a_j, \sum_{i=1}^{j-1} 2^{a_i}\right) \qquad \forall j = 1, \ldots, t,$$

in particular $g_1 = g(a_1, 0) = 1$ by (11). Then, from (11) we have $g_j - g_{j-1} = a_j - a_{j-1} - 1$ and summing over all indices j from 2 to t, we have

$$\sum_{j=2}^{t} (g_j - g_{j-1}) = \sum_{j=2}^{t} (a_j - a_{j-1} - 1) .$$

The left hand side is $g_t - g_1 = g_t - 1$, while the right hand side gives $a_t - a_1 - (t-1)$. Combining this results we have $g_t = a_t - a_1 - t + 2$. From the definition of d we can write

$$d(K) = \lfloor \log_2 K \rfloor + 1 - g_t = a_t + 1 - g_t = a_1 + t - 1,$$

and the claim is proven.
It is known that $a_1 = 1 + \mathrm{w}(\mathbf{b}(K-1)) - \mathrm{w}(\mathbf{b}(K))$ [23, A007814]. Using this fact, we have

$$d(K) = a_1 + t - 1 = 1 + \mathrm{w}(\mathbf{b}(K-1)) - \mathrm{w}(\mathbf{b}(K)) + t - 1 = \mathrm{w}(\mathbf{b}(K-1))$$

and this concludes the proof. □