One-time Traceable Ring Signatures

Alessandra Scafuro * and Bihan Zhang

North Carolina State University

Abstract. A ring signature allows a party to sign messages anonymously on behalf of a group, which is called ring. *Traceable* ring signatures are a variant of ring signatures that limits the anonymity guarantees, enforcing that a member can sign anonymously at most one message *per tag*. Namely, if a party signs two different messages for the *same tag*, it will be de-anomymized. This property is very useful in decentralized platforms to allow members to anonymously endorse statements in a controlled manner.

In this work we introduce *one-time traceable* ring signatures, where a member can sign anonymously only one message. This natural variant suffices in many applications for which traceable ring signatures are useful, and enables us to design a scheme that only requires a few hash evaluations and outperforms existing (non one-time) schemes.

Our one-time traceable ring signature scheme presents many advantages: it is *fast*, with a signing time of less than 1 second for a ring of 2^{10} signers (and much less for smaller rings); it is *post-quantum resistant*, as it only requires hash evaluations; it is *extremely simple*, as it requires only a black-box access to a generic hash function (modeled as a random oracle) and no other cryptographic operation is involved. From a theoretical standpoint our scheme is also the first anonymous signature scheme based on a black-box access to a symmetric-key primitive. All existing anonymous signatures are either based on specific hardness assumptions (e.g., LWE, SIS, etc.) or use the underlying symmetric-key primitive in a non-black-box way, i.e., they leverage the circuit representation of the primitive.

1 Introduction

Ring signatures, introduced by Kalai, Rivest and Shamir in [33], allow a party to anonymously sign a message on behalf of a group chosen in a spontaneous manner among a set of public keys. The crucial property of ring signatures that set them apart from group signatures [13] is that there is no manager who creates the keys, managing the group and de-anonymizing if necessary. In ring signatures, a party can generate its own pair of keys and the ring is simply the set of published public keys. Furthermore, at signing time, a signer can choose any subset of the published keys as a ring for its own signature, and no other party is able to de-anonymize it. Ring signatures are therefore particularly suitable

* Research supported by NSF grants #1718074 and NSF #1764025

for decentralized systems and have received renewed attention lately with the development of blockchains [30,29]. For instance, ring signatures could be used as a building block for a decentralized governance of a blockchain. In [37] for example, the goal is to provide a mechanism to vote on projects that should be funded with the blockchain treasury. In such an application, ring signatures could be used as a preliminary step to anonymously endorse projects that should be later considered for voting.

However, the lack of a manager in ring signatures enable members to abuse of their anonymity. In the example above, a member who wants to push a certain project, could anonymously compute multiple signatures endorsing the same project. Due to the anonymity guarantees, the other members cannot distinguish whether this project is endorsed by a single member or by multiple members[1].

Traceable Ring Signatures To overcome the unrestricted anonymity provided by ring signatures, Fujisaki and Suzuki in [19] introduced *traceable* ring signatures, where each message is associated to a 'tag' (a tag can be thought as a topic of discussion) and a party can anonymously sign only one message per tag. Specifically, traceable ring signatures provide an algorithm, called Trace, such that if Alice prepares two signatures σ, σ' for messages m and m' w.r.t the same tag,Trace will output the public key of Alice. Note, however, that if Alice computes two signatures σ, σ' for the *same* message m (signatures can be randomized) w.r.t. the same tag, the two signatures will be linked but Alice's identity will *not* be revealed. This property is called tag-linkability.

A downside of tag-linkability so defined is that a malicious party who just wants *to disrupt* the system, can mount a simple denial-of-service attack by continuously sending multiple signatures of the same message. While these signatures will be linked and then discarded, the identity of the attacker will not be revealed. Hence, the bad actor can keep the parties busy verifying and discarding signatures.

Traceable ring signatures have been constructed from the DDH assumption by Fujisaki and Suzuki [19,18] and from bilinear maps by Ho Au et al. in [2]. Such hardness assumptions however are not post-quantum resistant [35]. Only very recently, Branco and Mateus [10] provided the first[2] post-quantum resistant traceable ring signatures. Their construction is based on the syndrome decoding problem, a classical problem in coding theory that is conjectured to be post-quantum resistant. This scheme relies on the Fiat-Shamir heuristic and is proved secure in the classic random oracle model [5]. However, their construction is quite inefficient, with signature size of 240KB$\cdot N$, where N is the size of the ring, which would translate in 24 MB with a ring of just 100 people. Signing time estimations are not provided in [10], but they are expected to be high.

[1] A similar problem motivated the concept of *threshold* ring signatures [11], where a signature can be computed only if a least t members agree. This is very specific to applications where a *quorum* is required, and is not suitable in more general applications where we just want to enable members to express their opinion anonymously.

[2] Post-quantum *linkable* ring signatures existed in the literature before – and we discuss them in Section 2. However, they do *not* provide traceability.

Hence, the state-of-the-art of traceable ring signature offers only one scheme that is post-quantum resistant, but such scheme is currently impractical.

In this work we mitigate this situation by constructing a practical post-quantum resistant traceable ring signature scheme that only requires hash-function evaluations. Our key insight is to *enforce a one-time flavor* that gives us traceability almost for free, at the expenses of reusability. As we discuss below, this might not be a limitation and is an acceptable compromise in some applications where traceable ring signatures can be used. We elaborate on our contribution next.

1.1 Our Contribution

We introduce *one-time* traceable ring signatures and we construct them from a random oracle only. One-time means that the security properties – unforgeability, anonymity, non-frameability – are guaranteed only as long as a signer uses the secret key at most once. This allows us to provide a stronger public traceability guarantee that prevents denial-of-service attacks. Concretely, differently from the previous tag-linkability property achieved in [19,10], our public traceability property guarantees that if Alice signs twice, her identity will be revealed *even if she signed the same message*.

The one-time flavor can be a feature rather than a limitation in settings where signatures are used to endorse statements. Furthermore, our one-time traceable signature can be extended to many-time assuming a common public immutable state (e.g., the blockchain), using standard techniques that we discuss later in this section.

Our one-time traceable ring signature scheme advances the state of the art both from a practical and a theoretical perspective. From a practical standpoint the signing algorithm is extremely fast, requiring *less than a second* even for a ring of 1024 signers (see Table 1), and could be practical for applications such as the blockchain treasury decision discussed above (we stress however that we don't expect it to be suitable for applications such as anonymous payments).

Ring Size	Signature Size	Signing Time (sec)
2^6	131 KB	0.034
2^7	262 KB	0.068
2^8	524 KB	0.135
2^9	1 MB	0.273
2^{10}	2 MB	0.760

Table 1. Signature size and running times of our one-time traceable ring signature scheme Σ_{OTS} (described in Fig. 1) when $\mathcal{H}$ and G are instantiated with SHA3, and security parameter $\lambda = 128$.

The signature size of our scheme is $N \cdot \lambda^2$ *bits* (where N is the ring size and λ is the security parameter) and outperforms the size of the post-quantum

secure traceable signature of [10] (which size is $N \cdot 240$KB). For instance, for a ring of size 100, our scheme produces a signature of 204 KB versus the 24,000 KB required by the scheme of [10]. In Table 3 we compare the running time and size with various existing anonymous signatures that are linkable but not traceable (see Section 1.3 for details).

Finally, our signature scheme is *extremely simple*, and can be easily understood by anyone who understands the security properties of an hash-function modeled as a random oracle. We see this as an important advantage of our scheme since it makes it less prone to implementation errors and more agile (since we use the hash function as an oracle, it is easy to swap between implementations).

From a theoretical standpoint, our signature scheme is the first (one-time) ring signature that uses a hash function in a *black-box* way. Existing ring signatures rely on hardness assumptions that have a trapdoor flavor and hence require structure, e.g., trapdoor one-way permutation [33] or Cameleon Hash Plus [3] [27] functions. Others rely on specific hardness assumptions such as RSA [15], DDH [25,19] or Ring-LWE, NTRU, SIS/ISIS [4,36,27], syndrome, LWE [10,8]. The only ring signatures based on generic symmetric-key type of assumption, such as pseudo-random functions (PRF), rely on zero-knowledge proofs (e.g., [21]) and use the underlying primitive in a non-black-box manner. See Remark 1 for further discussions on non-black-box usage of cryptographic primitives.

Finally, our signature scheme reduces the gap between what we can achieve from black-box access to symmetric-key primitives in the *non*-anonymous setting and the anonymous setting. Indeed, it is well known that in the regular, *non*-anonymous, setting we can construct one-time signature schemes given only black-box access to a hash function [22,28] [4]. In contrast, in the anonymous setting, *even for one-time security*, no construction was known. (But we stress again that it is known from *non*-black-box use of symmetric-key primitives).

Remark 1. Black-box vs Non-black-box Usage of a Cryptographic Primitive. Ring signatures can be constructed generically using zero-knowledge proofs as follow. To sign a message on for a ring of N public keys, simply compute a non-interactive zero-knowledge proof of knowledge of the secret key associated to one of the N public keys. While this approach allows one to use *any* one-way function, note that the size of the resulting zero-knowledge proof *depends* on the specific one-way function that one chooses. Hence, different one-way functions lead to different performances in both size and running time. As a consequence, to improve performances, existing works (e.g., [21]) use less standard one-way functions, such as lowMC [1], that yield shorter zero-knowledge proofs. In contrast, when a primitive is used only as an oracle, that is, in a black-box manner, the signature size is *independent* of the complexity of the particular choice of the

[3] Chameleon Hash Plus were introduced in [27], it is a special hash function equipped with a trapdoor such that given *any* value y, a party P can produce x' s.t. $H(pk_P, x') = y$.

[4] Precisely, one-way functions are sufficient for one-time signatures, hash functions are used for succinctness and reusability.

primitive, but it depends only on the security parameter and how it relates with the output size of the function. The concrete running times will vary of course with the actual implementation of the oracle, but not the size. Even more, if the oracle is implemented via hardware (e.g., GPU), the running time is dramatically reduced. Finally, note that hardware implementation of the primitive cannot be leveraged when the primitive is used in a non-black-box manner.

On the one-time flavor We observe that, when an immutable shared state is available – as it is the case in a blockchain system – the one-time flavor is not a strong limitation since the common state can be leveraged to bootstrap the one-time use to many-time use, following standard techniques. For instance, consider the scenario where members associated with the governance of a blockchain are identified with a permanent public key. Whenever there is a topic of discussion on which the members are asked to give opinions, members can create a one-time, *per-topic* key and sign this key using their permanent key. Once all interested members have published their per-topic key (or a certain time has elapsed), the ring has formed. Each party can now sign their message anonymously on behalf of the ring for the specific topic, and no party can express more than one opinion/vote on the topic without being caught. This process can be bootstrapped so that when anonymously signing a message for topic 1, the party also signs the next one-time public key for topic 2, that will be added to the next ring.

1.2 Our Technique

The idea behind our traceable one-time ring signature scheme is simple. It leverages the equivocability of Naor's bit commitment scheme [31].

To start, let us recall Naor's commitment scheme. This scheme consists of two rounds. The first round is a random string R of 3λ bits, chosen by the receiver of the commitment, where λ is the security parameter. The second round is the commitment c computed as follows: $c := G(s) \oplus (b \cdot R)$, where G is a Pseudorandom Generator (PRG) with expansion from λ to 3λ bits, and $(b \cdot R)$ means the multiplication of each bit of R with the bit b, where b is the bit the sender wants to commit to. To *open* a commitment c (computed over the string R), the sender simply sends the PRG seed s (that was used to computed c). From the seed s, the receiver can then infer if the bit committed in c was 0 or 1 by simply trying to recompute c as either $G(s)$ $(b = 0)$ or $G(s) \oplus R$ $(b = 1)$. Naor's bit commitment scheme is statistically binding. However, if the string R instead of being chosen at random, were computed in an *"equivocal mode"*, that is, as $R = G(s_0) \oplus G(s_1)$, for two random seeds s_0, s_1, the commitment can be computed in such a way that can be equivocated, that is, opened as 0 or 1. This is done as follows: to commit, one always sends $c = G(s_0)$. Then in the decommitment phase, one sends s_0 if it wishes to open to bit 0 and s_1 otherwise. The seeds s_0, s_1 are therefore *trapdoors* that can allow the sender to open the same commitment c adaptively to either 0 or 1. Naor's bit commitment can be straightforwardly extended to string commitment. To commit a λ-bit string equivocally, one simply needs λ strings $(R_1, \ldots, R_\lambda)$ computed in *"equivocal*

mode". As we will see shortly, these strings will be the public key in our traceable ring signature.

Given this equivocation property of Naor's commitment, we can immediately create a one-time *traceable ring* signature as follows.

Public Key of a Member. When a member $\mathcal{U}_i$ wants to join the system, it will compute strings $(R_1, \ldots, R_\lambda)$ in *equivocal mode* and set them to be their public key. More precisely, to generate its own pair of public and private signing key, a ring member $\mathcal{U}_i$ proceeds as follows. It chooses λ pairs of random seeds $s^0_{i,j}$ and $s^1_{i,j}$, for $j \in [\lambda]$ (each seed is λ bits) and computes the j-th component of its public key as $pk_{i,j} := G(s^0_{i,j}) \oplus G(s^1_{i,j})$. The final public key that $\mathcal{U}_i$ publishes is $\mathbf{pk}_i = (pk_{i,1}, \ldots, pk_{i,\lambda})$, and it has size $\lambda(3\lambda)$. The values $s^0_{i,j}$ and $s^1_{i,j}$ are the secret trapdoors that $\mathcal{U}_i$ will use as a secret key to sign a message.

Ring. A ring R of N public keys therefore corresponds to the vector of keys $(\mathbf{pk}_1, \ldots, \mathbf{pk}_N)$, where each $\mathbf{pk}_i = (pk_{i,1}, \ldots, pk_{i,\lambda})$, is the vector of λ first rounds of Naor's commitment chosen by member $\mathcal{U}_i$ and computed in equivocal mode.

Signature. To sign a message m, on behalf of the ring R of size $|\mathsf{R}| = N$, a user $\mathcal{U}_i$ will proceed as follows. It will choose N random strings $x_1, \ldots, x_N$, one on behalf of each member of the chosen ring R, while it sets $x_i = 0$. Then, it will commit to each string x_q using the public key of member $\mathcal{U}_q$. Indeed, recall that the public key of $\mathbf{pk}_q$ is nothing but the first round of Naor's scheme that can be used by anyone to compute a commitment to a string. Hence, the signer $\mathcal{U}_i$ will compute N commitments $\mathbf{c}_1, \ldots, \mathbf{c}_N$, using the N public keys in R, and the i-th commitment is computed in equivocal mode. Once the commitments are fixed, the signer evaluates the random oracle $\mathcal{H}$ on input the message m to be signed, the ring R, and the commitments $\mathbf{c}_1, \ldots, \mathbf{c}_N$ just computed. It then obtains the value $z = \mathcal{H}(m, \mathsf{R}, \mathbf{c}_1, \ldots, \mathbf{c}_N)$ which is called the *target*. Note, the target z is a completely random string (due to the properties of the random oracle $\mathcal{H}$), that is sampled independently of the strings $x_1, \ldots, x_N$ committed in $\mathbf{c}_1, \ldots, \mathbf{c}_N$. After the target z is learnt, in order for the signature to be accepted, the signer must somehow show that the xor of the openings of all commitments $x_1 \oplus \cdots \oplus x_N$ is equal to z. Since $x_1 \oplus \cdots \oplus x_N$ were committed before the random oracle evaluation, this relation does not hold (with all but negligible probability). Hence, in order for the signer to satisfy the xor relation, it needs to equivocate at least one commitment. Since the signer $\mathcal{U}_i$ knows the trapdoors associated to the public key $\mathbf{pk}_i$, it can indeed equivocate the i-th commitment $\mathbf{c}_i$ so that it opens to a new string x^*_i that satisfies the xoring relation above.

The actual signature will consist only of openings, i.e., the PRG seeds, of the N string commitments. Note that among these PRG seeds, which are computed on-the-fly at random, there are the secret PRG seeds (i.e., $s^0_{i,j}$ or $s^1_{i,j}$) that $\mathcal{U}_i$ had chosen when computing its public key, and that are used specifically to equivocate the commitment. Looking ahead this means that upon a signature $\mathcal{U}_i$ is exposing a share of the secret key. If the same $\mathcal{U}_i$ tries to sign twice, it will end up using the other share and hence it will be traced. Our scheme is formally described in Figure 1.

Note that the idea of using some form of "trapdoor primitive", in combination with a target value z computed via the random oracle, is not new at all. In fact it is the pillar of most ring signatures. Trapdoors, however, are typically connected to cryptographic objects that have some structure and all previous works that follow this design did require specific structured assumptions (e.g., Discrete Log, Syndrome Decoding, Lattices, etc). The new insight of our paper is simply that, in the settings where traceability is required, one-time trapdoors can suffice, and we show that they can be derived *very cheaply* from an *unstructured object* such as the random oracle.

Security of Our Scheme. Next, we provide the intuition behind the security guarantees that our scheme provides. First, we discuss the security definition we adopt. We use the standard definition of traceable ring signature of [19], and *adapt it* to the *one-time* setting. Informally, a traceable one-time ring signature must satisfy the following properties: (1) anonymity, as long a party signs *up to one message* no-one can distinguish her identity, (2) traceability: there exists an algorithm Trace that given two signatures σ_1, σ_2 over a ring R, it outputs an identity $pk \in \mathsf{R}$ if both σ_1, σ_2 were computed with the secret associated to pk; (3) exculpability (also knows as non-frameability): no malicious party should be able to "frame" an honest party pk who signed only once; (4) one-time unforgeability: no malicious party can sign on behalf of a party who signed only once (this property is implied by exculpability and traceability).

We briefly argue why our scheme satisfy the above security properties.

Anonymity is guaranteed by the random oracle properties. Indeed, the only difference between a signature computed by member $\mathcal{U}_i$ and one computed by member $\mathcal{U}_j$ is in the position where the equivocation seeds are placed. If a signature is coming from $\mathcal{U}_i$, in position i we observe the equivocation seeds (either $s^0_{i,j}$ or $s^1_{i,j}$) used to computed the public key $\mathbf{pk}_i$, while for any other position $q \neq i$ we observe random PRG seeds that have no connection with how the public keys $\mathbf{pk}_q$ was computed. If the signature was computed by $\mathcal{U}_j$ we will observe the same but in position j. Now, first observe that the equivocation seeds ($s^0_{i,j}$ and $s^1_{i,j}$) were chosen uniformly at random when the public key was computed. Then observe that in the one-time setting, the adversary see at most one signature from each party. Hence it will only observe at most one equivocation seed (and never both). Computationally, an equivocation seed picked when computing the public key $\mathbf{pk}_i$ is distributed as a random PRG seed computed on the fly for the commitment on behalf $\mathbf{pk}_q$. Hence, given any signature, it is computationally infeasible to tell where the secret keys are placed (the formal argument is provided in Lemma 1).

Traceability follows directly from the fact that to successfully sign, a member $\mathcal{U}_i$ must use, and hence reveal, one of its equivocation seeds. If $\mathcal{U}_i$ computes two *distinct* signatures (even on the same message) σ, σ', it must hold that they had two different targets z, z'. Due to the random oracle properties, these target must be different in many positions. Now, recall that, in order to sign, a member $\mathcal{U}_i$ must equivocate the bit commitments $\mathbf{c}_i$ to hit a bit string x^*_i that satisfy the xor relation with the target. Now, if there are two targets z, z' that differ in

many positions, this means that there is at least on bit, say j, of x_i^* such that the j-th bit of x_i^* should be equal to 0 to accommodate for target z for σ while it should be 1 to accommodate for target z' for σ'. This will require to use seed $s_{i,j}^0$ in signature σ and seed $s_{i,j}^1$ signature σ'. Using σ, σ' it is therefore possible to recompute the j-th component of $\mathcal{U}_i$'s public key and hence de-anonymize these signatures.

Exculpability holds because in order to successfully frame a signer with public key $\mathbf{pk}_l$, the adversary needs to find two seeds s, s' such that $pk_{l,j} = G(s') \oplus G(s)$ for some index j. This is computationally infeasible (due to the one-way property of the random oracle), even if the adversary has observed one signature from $\mathbf{pk}_l$ – and therefore she has seen one of the seeds – and even if the adversary can create public keys maliciously and adaptively on honest public keys and signatures.

On Post-quantum Security. Our proof of security is carried in the classic random oracle model (ROM). Namely it assumes that a post-quantum adversary only has classical access to the Random Oracle (i.e., cannot make queries in superposition). This is consistent with all previous (traceable) ring signatures (e.g., [27,8,10]) that aimed at post-quantum security. The *Quantum* ROM [9], introduced by Boneh at al., considers an adversary that has quantum access to the Random Oracle and can make queries in superposition. In this setting, the practice of *programming* the Random Oracle, which is standard in the classic ROM, cannot be always applied and must be performed and analyzed very carefully. Exciting recent work [16,26] show techniques that facilitate the use programming in the QROM, paving the way to closing the gap between proofs in the classic ROM and QROM. We leave it as a future work to provide a security analysis of our scheme in the QROM model.

1.3 Performance Comparison

We compare our scheme with most recent traceable and linkable ring signatures that are post-quantum resistant (in the classic random oracle model). Specifically, we compare with the traceable ring signatures of Branco and Mateus [10] based on syndrome decoding, the linkable ring signatures Calamari and Falafl of Beullens et al. [8] that are based on isogeny and LWE, and the linkable ring signature Raptor [27] by Lum Ho Au and Zhang, based on NTRU and SIS. We compare w.r.t. hardness assumptions (Table 2) and performances. The latter are measured in terms of signature size and signing time (Table 3). We stress that this *is not an apple to apple comparison*, firstly, because our scheme provides only one-time security while the others provide many-time unforgeability and non-frameability, and secondly, because [8] and [27] are not traceable. In terms of assumptions, thanks to the one-time setting, our scheme uses the minimum assumption – only a random oracle —, while all other schemes require specific hardness assumption *in addition to* a random oracle. We also stress that, like ours, all such works only consider the classic random oracle, and leave it as a future work to analyze the scheme in the quantum random oracle model. For

the running times, since our scheme only requires hash function evaluations, it is the fastest [5]. For the signature size, our scheme outperforms the traceable signature of [10] and it is asymptotically better than the linkable ring signatures of [27]. Calamari and Felafl [8] however have *much better* signature size than ours (but they are not traceable). For our implementation we used an Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz, using only a single core and 1GB of RAM. We instantiated the random oracle with SHA3 implemented in GO [6].

Public key and Signature Size. We recall that the public key is $\mathbf{3}\lambda^2$ bits (as explained in Section 1.1). Hence, for $\lambda = 128$, the public key of each party is fixed to be 6KB. Computing a public key only consists in choosing 2λ seeds, computing 2λ PRG evaluations and xoring. The signature size depends on the size N of the ring, and is computed as $N \cdot \lambda^2$.

	Hardness Assumptions	Random Oracle
Branco at al. [10]	Syndrome Decoding	YES
Calamari [8]	Isogeny CSIDH-512	YES
Falafl [8]	Lattices MSIS MLWE	YES
Raptor [27]	Lattices NTRU	YES
This work	none	YES

Table 2. Hardness Assumptions used in most recent post-quantum secure linkable/traceable ring signatures. Our work provides one-time security.

	(Ring Size) N					
	2^3		2^6		2^{10}	
Branco at al. [3]	1920KB	–	1536KB	–	245MB	–
Calamari [8]	5.4KB	79 sec	8.3KB	16 min	10KB	2.7 hrs
Falafl [8]	30KB	< 1 sec	32 KB	1 sec	33KB	9 sec
Raptor [27]	11KB	0.017 sec	82KB	> 0.06 sec	1.3MB	–
This work	**16KB**	**0.004sec**	**131KB**	**0.03 sec**	**1MB**	**0.7 sec**

Table 3. This table shows how the signature size and running time varies with the size of the ring N, when the security parameter is 128 bits. (Note that for Raptor [27], the reported values are for only 100 bits of security), with 64 bits of quantum security.

[5] Note that this is true even if we add to the signature time, the time to computed the public key.

[6] `https://godoc.org/golang.org/x/crypto/sha3#ShakeSum128`

2 Related Work

In this section we review the literature on ring signatures. Most of the existing work are not traceable, hence they are not directly relevant to our result.

Ring Signatures. Ring signatures were introduced by Rivest, Shamir and Tauman in [34]. Their construction is based on any trapdoor permutation (or trapdoor function) and is proved in the ROM [5]. Bender, Katz and Morselli in [6] formalized ring signatures more carefully and showed a scheme based on general assumptions and ZAPs (i.e., two-round witness indistinguishable proofs) treating the underlying cryptographic primitives in a non-black-box manner. Libert et al. in [23] construct the first ring signature with size logarithmic in the ring from a lattice-based accumulator. Groth and Kohlweiss [20] show how to construct logarithmic ring signatures from 1-out-N commit-and-prove scheme from DDH assumption, this scheme was improved by Libert, Peters and Qian in [24]. Chandran, Groth, and Sahai [12] show a ring signature scheme with signature size $O(\sqrt{N})$ based on the on composite order groups with a bilinear maps. Dodis et al [15] provides a constant-size ring signature scheme based on RSA accumulators and the strong RSA assumption, and Nguyen [32] extends it with a pairing-based accumulator. Derler, Ramacher and Slamanig [14] show the first ring sub-linear ring signature scheme based only on symmetric primitives. Their construction is in the random oracle model, it is non-black-box and non-linkable. Katz, Kolesnikov and Wang [21] later provided optimized zero-knowlege proofs that can be used to build shorter and faster ring signature for a pseudo-random function only, used in a non-black-box manner. Lattice-based ring signatures have been shown in [27] by Lu, Ho Au and Zhang from the SIS and NTRU assumption. Beullens in [7] shows more efficient Sigma protocols for the MQ, PKP and SIS problems, that yield to the construction of more efficient ring signatures based on the same problem. Beullens, Katsumata and Pintore in [8] and Esgin et al. [17] construct efficient ring signatures from the isogenies and lattices problem. The scheme shown in [8] scale very well with the number of signers, by using the Merkle Tree in a very elegant way and avoiding using the circuit of the hash function. Unfortunately, signing (and verification) time is very high, with 79s for a ring as small as 8 people for the isogeny-based signature. This is due to the fact that the construction requires the parties to perform expensive group "actions". For the lattice-based implementation the bottleneck is not the lattice arithmetic, but rather the use of symmetric primitives (i.e. hashing, commitments and expanding seeds).

Linkable Ring Signatures Linkable Ring Signatures were introduced by Liu, Wei and Wong in [25]. They differ from traceable ring signatures in that they only allow to detect that two signatures are linked – i.e., computed by the same signer – but the identity of the signer is never revealed. Post-quantum resistant linkable ring signatures have been provided in [36,27,4], which have large signatures and withstand a somewhat weaker adversary who cannot maliciously craft its keys. Very recently, the Calamari and Falafl shown by Beullens et al. [8] yield much shorter linkable ring signatures, though the running times are not practical in some cases (see Section 1.3). These schemes are not traceable.

Traceable ring signatures. Traceable ring signatures were introduced by Fujisaki and Suzuki in [19], who constructed them based on the DDH assumption and the Fiat-Shamir heuristic, in the ROM. Fujisaki [18] presents a sub-linear scheme (where the size of the signature is $O(\sqrt{N})$ if N is the size of the ring), which trades the RO assumption with the assumption that there exists a trusted common reference string (CRS). Ho Au et al. in [2] propose a construction based on bilinear maps. All such constructions are based on variants of the hardness of the discrete logarithm problem, and are not post-quantum resistant.

3 Definitions

Notation. We use notation $[n]$ to denote the set $\{1, \ldots, n\}$. We use $y \leftarrow \mathsf{F}(x)$ to indicate y is the output of a randomized algorithm F on input x and $y := \mathsf{F}(x)$ if F is a deterministic algorithm. PPT stands for "probabilistic polynomial time". A function negl is *negligible* if for every positive polynomial p there is an integer n_0 such that for all integers $n > n_0$ it holds that $\mathsf{negl}(n) \leq \frac{1}{p(n)}$.

3.1 One-time Traceable Ring Signatures

One-time Traceable Ring Signatures. A ring signature is a signature computed on behalf of a group of N public keys $pk_1, \ldots, pk_N$, called the ring. To compute a ring signature, a signer must know one of the corresponding secret keys, e.g., sk_i. Ring signatures provide two properties: unforgeability and anonymity. Unforgeability means that only members of the ring can produce valid signatures. Anonymity means that given a ring signature σ on behalf of the ring $\mathsf{R} \subset \{pk_1, pk_2, \ldots\}$ it is infeasible to distinguish which secret key was actually used to compute the signature. For simplicity of exposition we will always assume that the ring is the set of all N public keys. A *traceable* ring signature [19] poses *restrictions* on the number of times a signer can anonymously sign a certain message. Namely, if a signer signs a message two times, then the two messages will be linked. We introduce *one-time* traceable ring signatures where all security properties hold assuming that a secret key is used at most once. We adapt the definition of traceable ring signatures of [19] to the one-time setting, and we provide a stronger traceability guarantee.

Definition 1. *A* one-time *traceable ring signature scheme is a tuple of PPT algorithms* $(\mathsf{GenKey}, \mathsf{RSign}, \mathsf{RVer}, \mathsf{Trace})$ *where:*

- **Key Generation:** $(pk_i, sk_i) \leftarrow \mathsf{GenKey}(1^\lambda)$ *A randomized algorithm run by a user* $\mathcal{U}_i$*. It takes in input the security parameter* λ *and outputs a verification key* pk_i *and a secret key* sk_i*.*
- **Signing Algorithm:** $(\mathsf{R}, \sigma, m) \leftarrow \mathsf{RSign}(\mathsf{R}, m, sk_l)$ *On input a ring* $\mathsf{R} \subseteq \{pk_1, \ldots, pk_N\}$*, a message* m *and a secret key* sk_l*, it outputs a signature* σ*. We assume that* $|\mathsf{R}| \geq 2$ *and each public key in the ring is distinct.*
- **Verification Algorithm:** $b \leftarrow \mathsf{RVer}(\mathsf{R}, m, \sigma)$ *it verifies a signature* σ *for message* m *and w.r.t ring* R*. It outputs 1 if the signature verifies, 0 otherwise.*

– **Trace:** $\mathsf{Trace}(\mathsf{R}, m_1, \sigma_1, m_2, \sigma_2)$ *on input two distinct signatures* σ_1 *and* σ_2 *on messages* m_1, m_2 *it outputs either "indep" or* $pk \in \mathsf{R}$.

Completeness. A one-time traceable ring signature scheme is complete if: for all $(pk_i, sk_i) \leftarrow \mathsf{GenKey}(1^\lambda)$, for all $\mathsf{R} \subseteq \{ pk_1, \ldots, pk_N\}$, for all $\sigma \leftarrow \mathsf{RSign}(\mathsf{R}, m, sk_l)$ s.t. $l \in [N]$: $Pr\left[\mathsf{RVer}(\mathsf{R}, m, \sigma) \rightarrow 1\right] = 1$

One-time Anonymity. This property guarantees that if an honest signer computes a single signature w.r.t an arbitrary ring R, the secret key used by the signer is anonymous w.r.t the honest public keys present in the ring R. To capture this, and following the definition of anonymity provided in [19], in the one-time anonymity game $\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Pi}$ all keys are maliciously computed by the adversary $\mathcal{A}$, except for two keys (pk_0, pk_1) which are honestly generated by the challenger. In the challenge phase, the adversary chooses a ring R, containing the keys pk_0 and pk_1, and a message m to sign. The challenger returns a signature σ^* which is computed with one of the secret keys sk_b where b is chosen at random. The adversary wins if it guesses the bit correctly. To capture the one-time setting, our adversary cannot observe any previous signature from pk_0 or pk_1. Concerning the other public keys besides pk_0, pk_1, note that the adversary "chooses" the ring R, hence it is assumed that the adversary controls those public keys and can obtain any signature computed by these *other* public keys (again, this follows the definition of [19]) [7]. Also notes that the adversary gets at most one signature. This capture the fact that anonymity is guaranteed if the honest party only signs at most once in an absolute sense. If the party uses the same secret keys with different rings, no anonymity is guaranteed.

Experiment $\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Pi}(1^\lambda)$

– Generate keys. Run $(pk_0, sk_0) \leftarrow \mathsf{GenKey}(1^\lambda)$ and $(pk_1, sk_1) \leftarrow \mathsf{GenKey}(1^\lambda)$ and send (pk_0, pk_1) to $\mathcal{A}$.
– Challenge phase. Upon receiving a message m and a ring R of public keys from the adversary proceed as follows. If pk_0, pk_1 are in R, pick bit $b \leftarrow \{0, 1\}$ and output $\sigma^* \leftarrow \mathsf{Sign}(\mathsf{R}, m, sk_b)$.
– (Decision). When $\mathcal{A}$ outputs b', output 1 iff $b = b'$.

Definition 2 (One-time Anonymity). *A one-time traceable ring signature scheme* Π *is anonymous if for all PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that:* $Pr\left[\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Pi}(1^\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$

Public Traceability. Traceability is a security property that protects the system against malicious users. To capture this, in the security game we assume that all keys are computed by the adversary, and we want that, given N adversarially-crafted keys, it should be infeasible for the adversary to produce $N+1$ signatures

[7] A stronger anonymity definition introduced in [6] demands that an adversary cannot break anonymity of a honest ring signature even if it knows the secret keys all honest parties. As remarked in [19] (Remark 2.3), this property cannot be achieved in combination with public traceability.

on $N+1$ distinct messages *without being traced*. We note that our definition is stronger than the traceability definition of [19]. Indeed, in the latter, traceability is considered a correctness property and not a security concern. Thus, [19] only guarantees that two messages signed with the same secret key are linkable, but the identity of the malicious signer is not necessarily revealed. Instead, in our definition, we guarantee that if the same secret key is used twice, to compute two distinct signatures, the corresponding public key is detected and revealed by the Trace algorithm. The security game $\mathsf{Exp}^{\mathsf{Trace}}$ is formally defined below.

Experiment $\mathsf{Exp}^{\mathsf{Trace}}_{\mathcal{A},\Pi}(1^\lambda)$

1. $\mathcal{A}$ on input the security parameter 1^λ outputs a ring R of N public keys, and a list of $N+1$ *distinct* signatures for the same ring R: $\{(m_1,\sigma_1),\ldots,(m_{N+1},\sigma_{N+1})\}$.
2. Return 1 if:
   (a) $\mathsf{RVer}(\mathsf{R},m_i,\sigma_i)=1$ for all $i\in[N+1]$ and
   (b) $\mathsf{Trace}(\mathsf{R},m_i,\sigma_i,m_j,\sigma_j)\notin\mathsf{R}$ for all $i,j\in[N+1]$, where $i\neq j$

Definition 3 (Traceability). *A one-time ring signature scheme Π is traceable if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$Pr\left[\mathsf{Exp}^{\mathsf{Trace}}_{\mathcal{A},\Pi}(1^\lambda)=1\right]\leq\mathsf{negl}(\lambda)$$

One-time Exculpability (Non-frameability) This property guarantees that if an honest ring member signed at most once, it cannot be framed. To capture this, we consider a target public key pk that is honestly generated, while all other keys are maliciously generated by the adversary. The goal of the adversary is to generate rings R,R' and two signatures σ,σ' such that algorithm Trace will output the target key pk. The adversary can ask for one signature computed by pk on an arbitrary ring and message. More formally, the experiment is described below:

Experiment $\mathsf{Exp}^{\mathsf{Frame}}_{\mathcal{A},\Pi}(1^\lambda)$

1. Generate target key. Run $(pk,sk)\leftarrow\mathsf{GenKey}(1^\lambda)$ and send pk to $\mathcal{A}$.
2. Signature request. Upon receiving m,S,pk from the adversary, where S is a set of arbitrary public keys and m is the message the adversary wants to see signed. If $pk\in S$ then output $\sigma\leftarrow\mathsf{Sign}(S,m,sk)$.
3. Output. Upon receiving (R,m,σ) and (R,m',σ') from the adversary. Output 1 if $\mathsf{Trace}(\mathsf{R},m,\sigma,m',\sigma')=pk$.

Definition 4 (One-time Exculpability (Non-frameability)). *A one-time traceable ring signature scheme Π is exculpable (non-frameable) if for all PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:*

$$Pr[\mathsf{Exp}^{\mathsf{Frame}}_{\mathcal{A},\Pi}(1^\lambda)=1]\leq\mathsf{negl}(\lambda)$$

Remark. One can also consider a more general experiment where there are several target honest keys. In this case the adversary could ask for signatures for each target party and then attempt to frame one of them. As remarked in [19] a construction satisfying the simpler experiment with one target key would satisfy also the more general one.

One-time Unforgeability. As noted in [19] (see Theorem 2.6, Pag. 8) if a signature scheme is traceable and exculpable, then it is also unforgeable. The intuition on why this is true is that, if it was not the case, namely, if the scheme is exculpable but not forgeable, then the adversary producing the forgery can be used to break exculpability and frame an honest user. On the other hand, if the forgery could not be used to frame the honest user, then this means that the scheme is not traceable. Thus, in this paper we will not explicitly prove unforgeability.

4 One-Time Traceable Ring Signature Scheme

The high-level idea of our one-time traceable ring signature scheme has been provided in Section 1.1. In this section we describe the scheme formally and provide the formal security proof.

Notation. For a bit string x we use notation $x[j]$ to denote the j-th bit of x. We use the subscript notation $pk_{i,j}$ to denote the j-th element of vector $\mathbf{pk}_i$. Namely, $\mathbf{pk}_i = (pk_{i,1}, \ldots, pk_{i,\lambda})$. Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ and $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be two random oracles. The one-time traceable signature scheme $\Sigma_{\mathsf{OTS}} = (\mathsf{GenKey}, \mathsf{RSign}, \mathsf{RVer}, \mathsf{Trace})$ is described in Figure 1.

Theorem 1. *If $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ and $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ are random oracles, then scheme $\Sigma_{\mathsf{OTS}} = (\mathsf{GenKey}, \mathsf{RSign}, \mathsf{RVer}, \mathsf{Trace})$ in Figure 1 is a one-time traceable ring signature.*

One-time Anonymity If a signature is computed by signer $\mathbf{pk}_l$, this means that the l-th set of commitments is equivocal, while the others are not. Recall that being equivocal means that each seed $r_{l,j}$ provided as part of the signatures has the property that $pk_{l,j} = G(r_{l,j}) \oplus G(\tilde{r}_{l,j})$ for some $\tilde{r}_{l,j}$. For any other $l' \neq l$ instead no such $\tilde{r}_{l',j}$ is known. Hence, in order to break anonymity, an adversary should be able to distinguish whether the seeds present/do not present such property. We prove that it is infeasible for any PPT adversary to distinguish if this is the case through a sequence of hybrid games. We do so by considering a mental experiment where the signature is computed by programming the random oracle rather than using the equivocation property of Naor's commitment. In such experiment, no secret key is used, all commitments are opened without a trapdoor, hence, a signature carries no information about the signer. The formal proof consists of a sequence of hybrid games, from the real experiments $\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Sigma_{\mathsf{OTS}}}(1^\lambda)$ where signatures are computed using the secret keys and equivocating the commitment, to the final experiment, where all signatures are computed by programming the random oracle and no commitment is equivocated.

One-Time Traceable Ring Signature Scheme: Σ_{OTS}

Key Generation $\mathsf{GenKey}(1^\lambda)$: A member $\mathcal{U}_i$ joins the system as follows:

1. Pick 2λ seeds: $s^0_{i,j} \leftarrow \{0,1\}^\lambda$, $s^1_{i,j} \leftarrow \{0,1\}^\lambda$ for $j \in [\lambda]$.
2. Set $pk_{i,j} := G(s^0_{i,j}) \oplus G(s^1_{i,j})$, and $sk_{i,j} = (s^0_{i,j}, s^1_{i,j})$.
3. Output public key $\mathbf{pk}_i = (pk_{i,1}||\ldots||pk_{i,\lambda})$ and secret key $\mathbf{sk}_i = (sk_{i,1}, \ldots, sk_{i,\lambda})$.

Signature $\mathsf{RSign}(\mathsf{R}, \mathbf{sk}_l, m)$. A member $\mathcal{U}_l$ signs a message as follows.

1. Parse $\mathsf{R} = (\mathbf{pk}_1, \ldots, \mathbf{pk}_N)$. Parse the secret key $\mathbf{sk}_l = (s^0_{l,1}, s^1_{l,1}, \ldots, s^0_{l,\lambda}, s^1_{l,\lambda})$.
2. For all $i \neq l$
   (a) Commit to a random string $x_i \in \{0,1\}^\lambda$ using the i-th public key $\mathbf{pk}_i$.
      i. For $j \in [\lambda]$, commit to the j-th bit of x_i: (1) pick a seed $r_{i,j} \leftarrow \{0,1\}^\lambda$; (2) compute $c_{i,j} := G(r_{i,j}) \oplus (x_i[j] \cdot pk_{i,j})$.
      ii. Set $\mathbf{c}_i := [c_{i,1}, \ldots, c_{i,\lambda}]$.
3. For $i = l$, set $\mathbf{c}_l := [G(s^0_{l,1}), \ldots, G(s^0_{l,\lambda})]$.
4. Compute target $z := \mathcal{H}(\mathsf{R}, m, \mathbf{c}_1, \ldots, \mathbf{c}_N)$.
5. Compute adjustment string $x^*_l := \bigoplus_i x_i \oplus z$. (Recall, x_i were defined in Step 2a)
6. Equivocate the l-th commitments $\mathbf{c}_l$ so that they open to x^*_l. Namely: set seed $r_{l,j}$ as $r_{l,j} := s^{x_l[j]}_{l,j}$ for each $j \in [\lambda]$.
7. Signatures σ:= $(x_i[j], r_{i,j})_{i\in[N], j\in[\lambda]}$. Output (R, σ, m).

Verification $\mathsf{RVer}(\mathsf{R}, \sigma, m)$.

1. Parse $\mathsf{R} = (\mathbf{pk}_1, \ldots, \mathbf{pk}_N)$, where the i-th key is: $\mathbf{pk}_i = (pk_{i,1}, \ldots, pk_{i,\lambda})$. If keys are not all distinct (or are not of the correct size) abort.
2. Parse $\sigma = (r_{i,j}, x_i[j])_{i\in[N], j\in[\lambda]}$.
3. Compute commitments: if $x_i[j] = 0$ set $c_{i,j} := G(r_{i,j})$, else set $c_{i,j} := G(r_{i,j}) \oplus pk_{i,j}$. Set $\mathbf{c}_i = (c_{i,1}, \ldots, c_{i,\lambda})$.
4. Compute $z' := \mathcal{H}(\mathsf{R}, m, \mathbf{c}_1, \ldots, \mathbf{c}_N)$. If $z' = \bigoplus_{i\in N} x_i$ accept the signature, else reject.

Trace $\mathsf{Trace}(\mathsf{R}, m_1, \sigma_1, m_2, \sigma_2)$.

1. Parse $\sigma_1 = (a_{i,j}, \alpha_{i,j})_{i\in[N], j\in[\lambda]}$.
2. Parse $\sigma_2 = (b_{i,j}, \beta_{i,j})_{i\in[N], j\in[\lambda]}$.
3. If there exist a public key $\mathbf{pk}_i \in \mathsf{R}$ and a j such that $G(\alpha_{i,j}) \oplus G(\beta_{i,j}) = pk_{i,j}$ then output $(1, \mathbf{pk}_i)$. Else output 0.

Fig. 1. One-time Traceable Ring Signature Scheme Σ_{OTS}.

Lemma 1 (One-time Anonymity). *If $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\lambda$ is a random oracle and $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ is a PRG, then scheme Σ_{OTS} achieves one-time anonymity according to Definition 2.*

Proof. Towards a contradiction, assume that there exists a PPT adversary $\mathcal{A}$ that wins game $\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Sigma_{\mathsf{OTS}}}(1^\lambda)$ with non-negligible probability $p(\lambda)$. We show an adversary that distinguishes the output of G with the same probability. The proof goes by hybrid arguments, and leverage the programmability of the random oracle $\mathcal{H}$.

Hybrid 0. Real World Experiment. Hybrid H_0 corresponds to experiment $\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Sigma_{\mathsf{OTS}}}(1^\lambda)$ instantiated with Σ_{OTS}. By contradicting hypothesis we assume that there is a PPT adversary $\mathcal{A}$ that wins game $\mathsf{Exp}^{\mathsf{OneTimeAnon}}_{\mathcal{A},\Sigma_{\mathsf{OTS}}}(1^\lambda)$ with probability $1/2 + p(\lambda)$.

Hybrid 1. Programming $\mathcal{H}$ and Avoiding Equivocation. In this experiment we slightly modify the computation of the signature in the following way. The value z is not computed as $z := \mathcal{H}(S, m, \mathbf{c}_1, \ldots, \mathbf{c}_N)$. Instead, in this experiment the challenger fixes values $(x_1, \ldots, x_N)$ and computes $z = \bigoplus x_l$. Later, the challenger programs the output of the RO so that when queried with input $q = (\mathsf{R}, m, \mathbf{c}_1, \ldots, \mathbf{c}_N)$ it outputs z. The commitments $\mathbf{c}_l$ associated to public key $\mathbf{pk}_l$ however are still computed as equivocal and the secret key is used to open them, however, the value x_l that must be opened to is defined in advance. Note that in this experiment the challenger can abort if two events happen: Event 1: the adversary queried a value $q = (\mathsf{R}, m, \mathbf{c}_1, \ldots, \mathbf{c}_N)$ before seeing the signature. Event 2: the value z chosen by the challenger was already used to answer previous random oracle queries.

Analysis. The difference between experiment H_0 and H_1 is only in the way the output of $\mathcal{H}$ is computed. H_0 and H_1 are distinguishable only if Event 1 or Event 2 happens, which happen with negligible probability as we show in Lemma 2 and Lemma 3.

Lemma 2. *If $\mathcal{H}$ is modeled as a programmable random oracle, Pr[Event 1] $\leq \frac{t}{2^{N\lambda^2}}$, where t is the number of queries to $\mathcal{H}$.*

Proof. Event 1 happens when adversary $\mathcal{A}$ queries the RO with input $q = (\mathsf{R}, m, \mathbf{c}_1, \ldots, \mathbf{c}_N)$ and later exactly the same commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_N)$ are generated by the challenger. Since each $\mathbf{c}_i$ is a Naor's commitment and is the output of the RO on a randomly chosen seed, the probability that $\mathcal{A}$ queried values $\mathbf{c}_1, \ldots, \mathbf{c}_N$ prior to see the signature corresponds to guessing such values, which happens with probability less than $\frac{1}{2^{N\lambda^2}}$.

Lemma 3. *If $\mathcal{H}$ is modeled as a programmable random oracle, Pr[Event 2] $\leq \frac{t}{2^\lambda}$, where t is the number of queries to $\mathcal{H}$.*

Proof. Event 2 happens when the output z chosen by the oracle challenger was already provided as the output of $\mathcal{H}$ from previous queries. Since each output of $\mathcal{H}$ is simulated by sampling a string uniformly at random in $\{0,1\}^\lambda$, this events happen with probability $\frac{t}{2^\lambda}$ where t is the (polynomial) number of queries.

Hybrid 2. Replacing honest public keys with truly random strings. In Hybrid H_2 the honest keys are truly random strings instead of the xor of two pseudorandom values. Note that since the random oracle is programmed, there is no need for trapdoors in this experiment. An adversary distinguishing H_2 from H_1 can be reduced to a PRG distinguisher. Since no secret key exist in the system, no identity can be leaked in this experiment. This concludes the proof of one-time anonymity. □

Traceability. This property guarantees that if two signatures were computed by the same signer, they will be linked and the identity of the signer detected. To win the traceability game $\mathsf{Exp}^{\mathsf{Trace}}$, an adversary must provide a ring R of N keys and $N+1$ distinct signatures $\sigma_1, \ldots, \sigma_{N+1}$ on $N+1$ messages $m_1, \ldots, m_{N+1}$, such that for any pair (m_a, σ_a), (m_b, σ_b), $\mathsf{Trace}(\mathsf{R}, m_a, \sigma_a, m_b, \sigma_b)$ outputs 0, i.e., no pairs of signatures is linkable. Intuitively, this is not possible since, in order to compute $N+1$ distinct signatures the adversary needs to compute at least $N+1$ equivocal commitments and thus use the trapdoor twice, hence revealing which key was used.

We now argue this claim more formally. First, recall that according to the Trace procedure two distinct signatures σ_a, σ_b for messages m_a, m_b are linked when the following condition is satisfied: There exists a tuple a, b, i, j, such that: $G(r^a_{i,j}) \oplus G(r^b_{i,j}) = K_{i,j}$.

Now, assume that an adversary is able to provide $N+1$ distinct signatures w.r.t the same ring $R = (K_1, \ldots, K_N)$ of *adversarially chosen* keys. Namely, the adversary provides:

$$\begin{array}{lll} \sigma_1 = & m^1, z^1 & (x^1_{i,j}, r^1_{i,j})_{i\in[N],j\in[\lambda]} \\ \ldots\ \ldots & \ldots & \ldots\ldots \\ \sigma_N = & m^{N+1}, z^{N+1} & (x^{N+1}_{i,j}, r^{N+1}_{i,j})_{i\in[N],j\in[\lambda]} \end{array}$$

Recall that to verify a signature σ_δ the verifier computes for each i, j, $c^\delta_{i,j} = G(r^\delta_{i,j}) \oplus x^\delta_i[j] \cdot K_{i,j}$ and then checks that: $z^\delta = \mathcal{H}(m^\delta, c^\delta_{i,j})_{i\in[N],j\in[\lambda]}$. Now, since the signatures are distinct, each must have a different target z. Since $\mathcal{H}$ is modelled as a RO, the following observations hold:

1. For each pair a, b, values z^a, z^b differ in at least $v \geq \lambda/2$ positions w.h.p. because they are the output of $\mathcal{H}$.
2. For any δ, adversary learns z^δ only after being committed to the values $x^\delta_i[j]$.
3. Due to (1) and (2) the adversary needs to equivocate at least one bit commitment to guarantee that $\bigoplus_{i,j} x^\delta_i[j] = z^\delta$.
4. To equivocate one commitment $c^\delta_{i,j}$ which is either $G(r^\delta_{i,j})$ or $G(r^\delta_{i,j}) \oplus K_{i,j}$, one needs to provide *another* seed $r^*_{i,j}$ such that $K_{i,j} = G(r^k_{i,j}) \oplus G(r^*_{i,j})$.

The above observations simply tell us that in each signature there is at least one equivocal opening, that is, a real seed $r^\delta_{i,j}$ such that $K_{i,j} = G(r^\delta_{i,j}) \oplus G(r')$. If the adversary provided only N signatures, this means that for each signature one seed (trapdoor) has been provided. Now if the adversary provided $N+1$ signatures then, for at least one of the keys the other seed must have been provided. In which case, that public key is linked. To complete the proof, we need to discard the possibility that an adversary is able to find multiple pairs of seeds that yield to the same public key $K_{i,j}$. Namely, the adversary could find two pairs $r^1_{i,j}, r^2_{i,j}, r^3_{i,j}, r^4_{i,j}$ such that $K_{i,j} = G(r^1_{i,j}) \oplus G(r^2_{i,j})$ and $K_{i,j} = G(r^3_{i,j}) \oplus G_{r^4_{i,j}}$. Since G is modeled as a Random Oracle, then the probability of finding such tuple is negligible.

Exculpability (Non-frameability). In the exculpability game the adversary participates at the same experiment as the anonymity experiment $\mathsf{Exp}^{\mathsf{OneTimeAnon}}$

but without the challenge phase. The goal of the adversary here is to output two signatures (R, m, σ) and (R, m', σ') with $m \neq m'$ such that there exists a *honest public key* $\mathbf{pk}_i \in \mathbb{H}$ such that: (1) $\mathbf{pk}_i \in R \cap R'$; (2) $\mathsf{SigVfy}(R, m, \sigma) = \mathsf{SigVfy}(R', m', \sigma') = 1$; (3) $\mathsf{Trace}(R, m, \sigma, m', \sigma') = \mathbf{pk}_i, 1$.

Recall the algorithm Trace shown in Figure 1. The condition for linking a key $pk_{i,j}$ is that for two seeds $\alpha_{i,j}$ and $\beta_{i,j}$ it holds that: $G(\alpha_{i,j}) \oplus G(\beta_{i,j}) = pk_{i,j}$. Note that in $\mathsf{Exp}^{\mathsf{OneTimeAnon}}$ the adversary is able to observe at most one signature computed under secret key $\mathbf{pk}_i$, and thus he is able to learn at most one seed $s^b_{i,j}$ for each $j \in [\lambda]$. From this view the adversary can win the exculpability in two ways: (1) Finding the preimage of $pk_{i,j} \oplus G(s^b_{i,j})$, (2) Finding two seeds α and β such that $G(\alpha) \oplus G(\beta) = pk_{i,j}$. In both cases the adversary is breaking the security of the PRG. The formal proof follows the same hybrid arguments shown for one-time anonymity (Lemma 1). The idea is to replace each honest public key $pk_{i,j}$ with a truly random string, and equivocating by programming the output $\mathcal{H}$. In this hybrid world, the adversary wins exculpability only by finding two seeds that satisfying the linkability equation, which corresponds to break the statically binding property of Naor's commitment.

One-Time Unforgeability. This property is implied by one-time exculpability and one-time traceability (see [19], Theorem 2.6 of).

References

1. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.
2. Man Ho Au, Joseph K. Liu, Willy Susilo, and Tsz Hon Yuen. Secure id-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theor. Comput. Sci.*, 469:1–14, 2013.
3. Rachid El Bansarkhani and Rafael Misoczki. G-merkle: A hash-based group signature scheme from standard assumptions. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography PQCrypto*, volume 10786 of *Lecture Notes in Computer Science*, pages 441–463. Springer, 2018.
4. Carsten Baum, Huang Lin, and Sabine Oechsner. Towards practical lattice-based one-time linkable ring signatures. *IACR Cryptology ePrint Archive*, 2018:107, 2018.
5. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.
6. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC*, pages 60–79, 2006.
7. Ward Beullens. Sigma protocols for mq, PKP and sis, and fishy signature schemes. In *EUROCRYPT (3)*, volume 12107 of *Lecture Notes in Computer Science*, pages 183–211. Springer, 2020.
8. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492. Springer, 2020.

9. Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011*, pages 41–69, 2011.
10. Pedro Branco and Paulo Mateus. A traceable ring signature scheme based on coding theory. In *PQCrypto*, volume 11505 of *Lecture Notes in Computer Science*, pages 387–403. Springer, 2019.
11. Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer, 2002.
12. Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In *Automata, Languages and Programming*, pages 423–434, 2007.
13. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.
14. David Derler, Sebastian Ramacher, and Daniel Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *Post-Quantum Cryptography*, pages 419–440, 2018.
15. Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 609–626. Springer, Heidelberg, May 2004.
16. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 356–383. Springer, 2019.
17. Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: New techniques for shorter and faster constructions and applications. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 115–146. Springer, 2019.
18. Eiichiro Fujisaki. Sub-linear size traceable ring signatures without random oracles. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 393–415. Springer, 2011.
19. Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *Public Key Cryptography - PKC 2007*, pages 181–200, 2007.
20. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015*, pages 253–280, 2015.
21. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM SIGSAC CCS 2018*, pages 525–537, 2018.
22. Leslie Lamport. Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979.
23. Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *Advances in Cryptology - EUROCRYPT 2016*, pages 1–31, 2016.
24. Benoît Libert, Thomas Peters, and Chen Qian. Logarithmic-size ring signatures with tight security from the DDH assumption. In *Computer Security - ESORICS 2018*, pages 288–308, 2018.

25. Joseph K. Liu, Victor K. Wei, and Duncan S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *Information Security and Privacy: ACISP*, pages 325–335, 2004.
26. Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 326–355. Springer, 2019.
27. Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In *ACNS*, volume 11464 of *Lecture Notes in Computer Science*, pages 110–130. Springer, 2019.
28. Ralph C. Merkle. A certified digital signature. In *Advances in Cryptology - CRYPTO '89*, pages 218–238, 1989.
29. Monero. Monero, a secure, private and untraceable digital currency. 2016.
30. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. (2012):28, 2008.
31. Moni Naor. Bit commitment using pseudo-randomness. In *CRYPTO*, 1989.
32. Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA 2005*, pages 275–292, 2005.
33. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001.
34. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001*, pages 552–565, 2001.
35. Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. Ieee, 1994.
36. Wilson Abel Alberto Torres, Ron Steinfeld, Amin Sakzad, Joseph K. Liu, Veronika Kuchta, Nandita Bhattacharjee, Man Ho Au, and Jacob Cheng. Post-quantum one-time linkable ring signature and application to ring confidential transactions in blockchain (lattice ringct v1.0). In *Information Security and Privacy*, pages 558–576, 2018.
37. Bingsheng Zhang, Roman Oliynykov, and Hamed Balogun. A treasury system for cryptocurrencies: Enabling better collaborative intelligence. In *Network and Distributed System Security Symposium, NDSS*, 2019.