Count Me In! Extendability for Threshold Ring Signatures

Diego F. Aranha[1]([✉]), Mathias Hall-Andersen[1], Anca Nitulescu[3],
Elena Pagnin[2], and Sophia Yakoubov[1]

[1] Aarhus University, Aarhus, Denmark
{dfaranha,ma,sophia.yakoubov}@cs.au.dk
[2] Lund University, Lund, Sweden
elena.pagnin@eit.lth.se
[3] Protocol Labs, Wilmington, USA
anca@protocol.ai

Abstract. Ring signatures enable a signer to sign a message on behalf of a group *anonymously*, without revealing her identity. Similarly, threshold ring signatures allow several signers to sign the same message on behalf of a group; while the combined signature reveals that some threshold t of the group members signed the message, it does not leak anything else about the signers' identities. Anonymity is a central feature in threshold ring signature applications, such as whistleblowing, e-voting and privacy-preserving cryptocurrencies: it is often crucial for signers to remain anonymous even from their fellow signers. When the generation of a signature requires interaction, this is difficult to achieve. There exist threshold ring signatures with non-interactive signing—where signers locally produce *partial* signatures which can then be aggregated—but a limitation of existing threshold ring signature constructions is that all of the signers must agree on the group on whose behalf they are signing, which implicitly assumes some coordination amongst them. The need to agree on a group before generating a signature also prevents others—from outside that group—from endorsing a message by adding their signature to the statement post-factum.

We overcome this limitation by introducing *extendability* for ring signatures, same-message linkable ring signatures, and threshold ring signatures. Extendability allows an untrusted third party to take a signature, and *extend* it by enlarging the anonymity set to a larger set. In the extendable threshold ring signature, two signatures on the same message which have been extended to the same anonymity set can then be combined into one signature with a higher threshold. This enhances signers' anonymity, and enables new signers to anonymously support a statement already made by others.

For each of those primitives, we formalize the syntax and provide a meaningful security model which includes different flavors of anonymous extendability. In addition, we present concrete realizations of each primitive and formally prove their security relying on signatures of knowledge and the hardness of the discrete logarithm problem. We also describe a generic transformation to obtain extendable threshold ring signa-

tures from same-message-linkable extendable ring signatures. Finally, we implement and benchmark our constructions.

Keywords: Threshold ring signatures · Anonymity · Extendability

1 Introduction

Anonymity has become a requirement in many real-world implementations of cryptographic systems and privacy-enhancing technologies, including electronic voting [24], direct anonymous attestation [9], and private cryptocurrencies [27]. Another compelling scenario is whistleblowing of organizational wrongdoing. In this case, an insider publishes a *secret* in a manner that convinces the public of its authenticity, while having his/her identity protected [25]. In all of these applications, a large *anonymity set*, i.e., set of users who may have performed a certain action, is crucial in order to not reveal who exactly is behind it.

Group signatures enable any member of a given group to sign a message, without revealing which member signed. However, group signatures suffer from the drawback that they require trusted setup for every group. Ring signatures are a manager-free variant of group signatures. They enable individual users to sign messages anonymously on behalf of a dynamically chosen group of users, while hiding the exact identity of the signer(s) [25]. Traditionally, this is enabled by including a "ring" $\mathcal{R}$ of public keys (belonging to all possible signers, including the actual signer) as an input to the signing algorithm; a ring signature does not reveal which of the corresponding secret keys was used to produce it. There are many ways to construct ring signatures using different building blocks: classic RSA [13], bilinear pairings [5,12,30], composite-order groups [7,26], non-interactive zero knowledge [6,20], and, most recently, quantum-safe isogenies and lattices [4,14,18,19].

Threshold ring signatures are a threshold variant of this primitive [8], which allow some t signers to sign a message on behalf of a ring $\mathcal{R}$ of size larger than t. The signature reveals that t members of the ring signed the message, but not the identities of those members. Some threshold ring signature schemes are *flexible* [23], meaning that even after the threshold ring signature has been produced for a given ring $\mathcal{R}$, another signer from that ring can participate, resulting in a threshold ring signature for the same ring $\mathcal{R}$ but with a threshold of $t + 1$. However, if a signer from *outside* the ring wants to participate, existing constructions do not support this. All existing constructions of ring and threshold ring signatures have a common limitation: the ring of potential signers is fixed at the time of signature generation. In particular, it is not possible to have the added flexibility of publicly "adjusting" the ring, i.e., to extend the initial ring to a larger one, increasing the anonymity set. Increasing the size of the set of potential signers not only increases the anonymity provided by the signature, but also makes threshold systems easier to realize in practice.

To work in practice, standard threshold ring signatures need all of the signers to independently sign the same message μ with the same ring $\mathcal{R}$, which must

include the public keys of all t signers. We are interested in relaxing this implicit synchronization requirement.

1.1 Our Contributions

In this paper, we introduce a new property of (threshold) ring signatures which we call *extendability*. A (threshold) ring signature scheme is *extendable* if it allows anyone to enlarge the set of potential signers of a given signature. Extendable threshold ring signatures are fundamental for whistleblowing, where one party may want to "join the cause" after it becomes public. Extendability, together with flexibility, enables a signer A to join a threshold ring signature which was produced using an anonymity ring $\mathcal{R}$ that does not contain A. This can be done by first extending the existing signature to a new ring $\mathcal{R}' \supseteq \mathcal{R} \cup \{A\}$ which contains both the ring used by previous signers as well as the new signer. Then, thanks to flexibility, the new signer can add their own signature with respect to the new ring $\mathcal{R}'$ (using sk_A). (Of course, an observer who has seen signatures under the old ring $\mathcal{R}$ and under the new ring $\mathcal{R}'$ will be able to determine $\mathcal{R}'\backslash\mathcal{R}$; this is inherent—since an observer can always tell which ring a signature is meant for by attempting verification—and can help that observer narrow down possibilities for the identity of A. However, an observer who has *not* seen a signature under the old ring $\mathcal{R}$ will learn nothing additional about the identity of A.)

In addition to drawing formal models, we give the first constructions of *extendable ring signatures*, *same-message linkable* extendable ring signatures and extendable *threshold* ring signatures. We provide a proof of concept implementation of our construction, benchmark the signing and verification running times as well as the signature size.

Constructions from Signatures of Knowledge and Discrete Log. We build extendable ring signatures and same-message linkable extendable ring signatures using signatures of knowledge. Each signature will include several elements of a group, with the property that all of their discrete logs cannot be known. (This is because the product of the elements gives a discrete log challenge which is part of the public parameters.) A signer signs the message with a *signature of knowledge* that proves that she knows either her own secret key, or the discrete log of one of the elements. The signer uses her secret key for this (and so can use the element for which the discrete log is unknown), but for each of the other signers' public keys in the ring, she includes a signature of knowledge using the discrete log of one of the elements. Because all of the element discrete logs cannot be known, a verifier is convinced that at least one signature of knowledge is produced using a secret key, and that therefore the overall signature was produced by one of the members of the ring.

We build extendable threshold ring signatures similarly, but by choosing the elements in such a way that at least t of their discrete logs cannot be known without revealing the discrete log of a challenge element in the public parameters. We enforce this by placing the elements on a polynomial of appropriate degree.

A Generic Transformation. One might hope to build extendable threshold ring signatures by concatenating t extendable ring signatures; however, we would need to additionally prove to the verifier that the t signatures were produced by t different signers. Building such a proof would require interaction between the signers, and it would be challenging to maintain the proof as the ring is expanded. Instead, we solve this problem using a primitive which we call a *same-message linkable extendable ring signatures*, where, given two signatures on the same message, it is immediately clear whether they were produced by the same signer. Our realizations of this primitive provide linkability without revealing the signer's identity or resorting to additional zero knowledge proofs and can be used to construct extendable threshold ring signatures in a generic way.

Implementation. We provide an implementation that demonstrates the concrete efficiency of our schemes. The benchmarks place our constructions firmly within the realm of practicality: an extendable ring signature for a ring with 2048 members can be created in 0.45 s.

1.2   Related Work

Ring signatures were first introduced by Rivest, Shamir, and Tauman in [25] as a mechanism to leak secrets anonymously. This initial construction was based on trapdoor permutations, but other schemes quickly followed. A threshold version of their scheme was proposed the following year by Bresson *et al.* [8], together with a revised security analysis for the original scheme. By using RSA accumulators and the Fiat-Shamir transform, a ring signature scheme with signature sizes independent of the ring size was later constructed by Dodis *et al.* [13]. (A similar scheme in the threshold setting was described by Munch-Hansen *et al.* [22].) In addition to the hardness of integer factorization, pairing groups were used in early constructions to obtain ring signatures in the conventional [5] and identity-based [30] settings.

The first ring signature constructions were all based on the random oracle model, but alternatives proven secure in the common reference string model were later proposed [12,26], including constructions with sublinear [10] and constant signature size [7]. In the standard model, early constructions were based on 2-round public coin witness-indistinguishable protocols [1], but more recent constructions rely on non-interactive zero-knowledge proofs [6,20].

Threshold ring signature schemes come in many flavors, with many constructions based on RSA and bilinear maps and security based on number-theoretic assumptions [17,28,29]; and post-quantum schemes based both on lattices [3] and coding theory [21]. The post-quantum schemes have traditionally relied on the Fiat-Shamir transform, the quantum security of which is not fully determined. Recent work in threshold ring signatures has provided both improved security definitions [22] and constructions based on the quantum-safe Unruh's transform [15].

2   Background and Preliminaries

Notation. We denote the set of natural numbers by $\mathbb{N}$ and let the computational security parameter of our schemes to be $\lambda \in \mathbb{N}$. We say that a function is *negligible* (in λ), and we denote it by $\mathtt{negl}$, if $\mathtt{negl}(\lambda) = \Omega(\lambda^{-c})$ for any fixed constant $c > 1$. We also say that a probability is *overwhelming* (in λ) if it is greater than or equal to $1 - \mathtt{negl}$. Given two values $a < b$, we denote the list of integer numbers between a and b as $[a, \ldots, b]$. For compactness, when $a = 1$, we simply write $[b]$ for $[1, \ldots, b]$. We denote empty strings as ϵ. Unless otherwise specified, all the algorithms defined throughout this work are assumed to be probabilistic Turing machines that run in polynomial time (abbreviated as PPT). When sampling the value a uniformly at random from a set X, we employ the notation $a \leftarrow_R X$. In our constructions, we denote by $\mathsf{GroupGen}(1^\lambda)$ the algorithm that, given in input the security parameter, outputs the tuple $(p, g, \mathbb{G})$, where p is a 2λ-bit prime; g is a group generator and $\mathbb{G}$ is a description of a group of order p, $\mathbb{G} = \langle g \rangle$. Through out the paper, we assume solving the Discrete Logarithm Problem in $\mathbb{G}$ is computationally hard.

2.1   Main Primitives

Ring Signatures. A ring signature scheme is defined as a tuple of four probabilistic polynomial time algorithms $\mathbf{RS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

$\mathsf{Setup}(1^\lambda) \rightarrow \mathtt{pp}$: Takes a security parameter λ and outputs a set of public parameters $\mathtt{pp}$. The public parameters are implicitly input to all subsequent algorithms.

$\mathsf{KeyGen}() \rightarrow (\mathtt{pk}, \mathtt{sk})$: Produces a key pair $(\mathtt{pk}, \mathtt{sk})$.

$\mathsf{Sign}(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}}, \mathtt{sk}_i) \rightarrow \sigma$: Takes a message $\mu \in \{0, 1\}^*$ to be signed, the set of public keys of the users within the ring of identifiers $\mathcal{R}$, and the secret key $\mathtt{sk}_i$ of the signer $i \in \mathcal{R}$ (i.e., the signer's public key must appear in the set $\{\mathtt{pk}_j\}_{j \in \mathcal{R}}$). Outputs a signature σ.

$\mathsf{Verify}(\mu, \{\mathtt{pk}_i\}_{i \in \mathcal{R}}, \sigma) \rightarrow \mathtt{accept}/\mathtt{reject}$: Takes a message, a set of public keys of the users within a ring, and a signature σ. Outputs $\mathtt{accept}$ or $\mathtt{reject}$, reflecting the validity of the signature σ on the message μ with respect to the ring $\mathcal{R}$.

Naturally, a ring signature scheme should satisfy correctness, meaning that any signature generated by Sign should verify (against the signed message and the original ring). A *secure* ring signature scheme RS must additionally satisfy (a) unforgeability, meaning that no adversary should be able to produce a verifying signature without knowledge of at least one signing key corresponding to a public verification key in the ring, and (b) anonymity, meaning that no adversary should be able to tell from a signature which ring member produced it. We refer to prior work for the formal definitions of a ring signature scheme [8,13,16].

Threshold Ring Signatures. There are many different ways to formalize the threshold ring signature syntax, which force varying degrees of interaction between the t signers. A non-interactive threshold ring signature scheme is defined as a tuple of five probabilistic polynomial time algorithms (Setup, KeyGen, Sign, Combisign, Verify). The algorithms Setup, KeyGen, Sign and Verify are syntactically the same as in a ring signature scheme, with the exceptions that (1) Sign now outputs a *partial* signature σ_i for signer i, and (2) Verify now additionally takes the threshold t as input. The algorithm Combisign, described below, combines t partial signatures into a single threshold signature. It may be run by any third party, as it does not require any signers' secrets.

Combisign($\{\sigma_i\}_{i \in \mathcal{S} \subseteq \mathcal{R}}$) → σ: Takes partial signatures $\{\sigma_i\}_{i \in \mathcal{S}}$ from $|\mathcal{S}| = t$ signers, and outputs a combined signature σ.

There are also interactive threshold ring signature schemes. In this case Sign (which in this case also subsumes Combisign) is an interactive protocol run between the signers, which implicitly requires the signers to be aware of one another's identities.

Finally, there is a solution in between, where one signer produces the initial signature, and then the remaining signers pass the signature around, and each "joins" the signature before passing it on. In such a syntax, each signer must only receive (at most) one message from one other signer, and send (at most) one message to one other signer. Instead of Combisign, in such a syntax we have a Join algorithm, described below.

Join($\mu, \{\text{pk}_j\}_{j \in \mathcal{R}}, \text{sk}, \sigma$) → σ': Takes a message μ, a set of public keys $\{\text{pk}_j\}_{j \in \mathcal{R}}$, which includes the public key of the new signer, the new signer's secret key sk, and a signature σ produced by a subset of $\mathcal{R}$ (with threshold level t'). Outputs a modified threshold ring signature σ' with threshold $t' + 1$.

2.2 Main Building Blocks

Signatures of Knowledge. Signatures of Knowledge (SoKs) [11] generalise digital signatures by replacing the public key with an *instance*, or *statement*, in a NP language. A signer can generate a valid signature for a message only if she has a valid witness for the statement.

Syntax. A SoK for an efficiently decidable binary relation $\mathscr{R}$ is defined as a tuple of PPT algorithms **SoK** = (Setup, Sign, Verify, SimSetup, SimSign):

Setup($1^\lambda, \mathscr{R}$) → pp: Takes a security parameter λ and a binary relation $\mathscr{R}$ and returns public parameters pp. The input pp is implicit to al subsequent algorithms.

Sign(μ, ϕ, w) → σ: Takes as input a message $\mu \in \{0,1\}^*$, a statement ϕ, and a witness w. Outputs a signature σ.

Verify(μ, ϕ, σ) → accept/reject: Takes as input a message μ, a statement ϕ, and a signature σ. Outputs accept if the the signature is valid, reject otherwise.

$\mathsf{SimSetup}(1^\lambda, \mathscr{R}) \to (\mathtt{pp}, \mathtt{td})$: A simulated setup which takes as input a relation $\mathscr{R}$ and returns public parameters $\mathtt{pp}$ and a trapdoor $\mathtt{td}$.

$\mathsf{SimSign}(\mathtt{td}, \mu, \phi) \to \sigma'$: A simulated signing algorithm that takes as input a trapdoor $\mathtt{td}$, a message μ and a statement ϕ and returns a simulated signature σ'.

A SoK scheme should satisfy correctness, simulatability and extractability as formally defined in the full version of this paper.

3   Extendable Ring Signatures

Ring signatures enable a signer to generate a signature while hiding her identity within a ring of potential signers. Even though the ring of potential signers $\mathcal{R}$ can be arbitrary[1]—realizing ad-hoc anonymity sets—existing constructions do not let a third party increase the size of $\mathcal{R}$ after the signature is produced. Once a signature is generated, it is not possible to "extend" it to a larger anonymity set; in other words, ring signatures do not allow one to modify a signature and obtain a new signature for the same message but with a wider set of potential signers. Our notion of extendability aims to allow exactly this, while preserving signer anonymity.

3.1   Syntax

An extendable ring signature scheme (ERS) is a ring signature scheme that has an additional algorithm, Extend, that allows any third party to enlarge the ring of potential signers of a given signature:

$\mathsf{Extend}(\mu, \{\mathtt{pk}_i\}_{i \in \mathcal{R}}, \sigma, \{\mathtt{pk}_j\}_{j \in \mathcal{R}'}) \to \sigma'$: Takes a message, a set of public keys (indexed by the ring $\mathcal{R}$), a signature σ, and a second ring of public keys (indexed by $\mathcal{R}'$). It outputs a modified signature σ' which verifies under $\mathcal{R} \cup \mathcal{R}'$.

Remark 1. Consider an ERS scheme where Extend can be repeatedly applied to extend a signature a polynomial number of times. In this case, we can have a very simple instantiation where Sign always produces a signature for the singleton ring $\{\mathtt{pk}\}$ containing only the signer's public key $\mathtt{pk}$, and Extend is called only on singleton extension rings, i.e., $|\mathcal{R}'| = 1$. A signature for the singleton ring can be extended to any ring by having the signer iteratively apply Extend with a single additional public key.

For the following definitions, we use ladders of rings, i.e., tuples $\mathtt{lad} = (i, \mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(l)})$, where i is a signer identity, and the rings $\mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(l)}$ are all sets of signer identifiers. In addition, we make use of an algorithm $\mathsf{Process}(\mu, \mathsf{L_{keys}}, \mathtt{lad})$, that we describe in Fig. 1. As the name suggests,

---

[1] The ring $\mathcal{R}$ should of course contain the signer's identity.

this algorithm processes a ladder `lad` on a given message μ using keys from $\mathsf{L}_{\mathsf{keys}}$ (the list of generated keys). Process signs μ using sk_i under the ring $\mathcal{R}^{(1)}$, and extends the signature to all the subsequent rings (using keys stored in the list $\mathsf{L}_{\mathsf{keys}}$). Process returns an extendable ring signature σ, which is the output of the last operation.

ERS.Process$(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad})$

$1:$ Parse $\mathtt{lad}$ as $(i, \mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(l)})$

$2:$ **if** $i \notin \mathcal{R}^{(1)}$**return** $\bot$ // make sure all public keys are in $\mathsf{L}_{\mathsf{keys}}$

$3:$ **for** $j \in \mathcal{R}^{(1)} \cup \cdots \cup \mathcal{R}^{(l)}$: **if** $(j, \mathtt{pk}_j, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$ **return** $\bot$

    // make sure the signer's secret key is available in $\mathsf{L}_{\mathsf{keys}}$

$4:$ **if** $\mathtt{sk}_i = \bot$:   **return** $\bot$ // make sure $\mathtt{sk}_i$ is not corrupted

    // process the instructions in the ladder

$5:$ $\sigma^{(1)} \leftarrow$ **ERS**.Sign$(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}^{(1)}}, \mathtt{sk}_i)$

$6:$ **for** $l' \in [2, \ldots, l]$:

$7:$ $\mathcal{R}^{(l')} \leftarrow \mathcal{R}^{(l')} \cup \mathcal{R}^{(l'-1)}$ // enforce rings form an increasing chain

$8:$ $\sigma^{(l')} \leftarrow$ **ERS**.Extend$(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}^{(l'-1)}}, \sigma^{(l'-1)}, \{\mathtt{pk}_j\}_{j \in \mathcal{R}^{(l')}})$

$9:$ $\sigma \leftarrow \sigma^{(l)}$

$10:$ **return** σ

Fig. 1. The Process algorithm for extendable ring signatures.

For correctness, we require that any—possibly extended—signature σ output by Process verifies for the given message, under the final ring $\mathcal{R}^{(l)}$.

Definition 1 (Correctness for ERS). *An extendable ring signature scheme ERS is said to be correct if, for all security parameters $\lambda \in \mathbb{N}$, for any message $\mu \in \{0,1\}^*$, for any ladder $\mathtt{lad} = (i, \mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \cdots, \mathcal{R}^{(l)})$ where $i \in \mathcal{R}^{(1)}$ and $l > 0$, it must hold that:*

$$\Pr\left[\begin{array}{c} \textbf{ERS}.\mathsf{Verify}(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}}, \sigma) \\ = \mathtt{accept}\ OR\ \sigma = \bot \end{array} \left| \begin{array}{l} \mathcal{R} = \mathcal{R}^{(1)} \cup \cdots \cup \mathcal{R}^{(l)} \\ \mathtt{pp} \leftarrow \textbf{ERS}.\mathsf{Setup}(1^\lambda) \\ \mathsf{L}_{\mathsf{keys}} \leftarrow \{(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \textbf{ERS}.\mathsf{KeyGen}()\}_{j \in \mathcal{R}} \\ \sigma \leftarrow \textbf{ERS}.\mathsf{Process}(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad}) \end{array} \right. \right] = 1$$

3.2   Security Model

Definition 2 (Secure ERS). *An extendable ring signature scheme is secure if it satisfies correctness (Definition 1), unforgeability (Definition 3), anonymity (Definition 4), and some notion of anonymous extendability (described below).*

Unforgeability. Extendable ring signatures inherit their unforgeability require-
ment from regular ring signatures: no adversary should be able to produce a sig-
nature unless they know at least one secret key belonging to a party in the ring.
Notably, the unforgeability experiment for ERS (cmEUF, detailed in Figure 2)
needs to take into account that the adversary can arbitrarily expand the ring
associated to a signature. To rule out trivial attacks derived with this strategy,
the adversary does not break unforgeability if the candidate forgery could be
generated by extending the outcome of a signing query (line 5 in $\mathsf{Exp}_{\mathcal{A},\mathbf{ERS}}^{\mathrm{cmEUF}}(\lambda)$).
Additionally, to account for the key duplication attack (where an adversary reg-
isters an existing public key to a new identity), instead of simply checking if
the identities in the output ring are among the corrupted ones, the experiment
checks if the *public keys* belonging to the parties involved in the adversary's
output ring are among the corrupted ones (line 7, Fig. 2).

$\mathsf{Exp}\ _{\mathcal{A},\mathbf{ERS}}^{\mathrm{cmEUF}}(\lambda)$

1 : $\mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}} \leftarrow \varnothing$

2 : $\mathsf{pp} \leftarrow \mathbf{ERS}.\mathsf{Setup}(1^\lambda)$

3 : $O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$

4 : $(\mu^*, \mathcal{R}^*, \sigma^*) \leftarrow \mathcal{A}^O(\mathsf{pp})$

// rule out trivial wins due to ring expansion

5 :   **if** $\exists\ (\mu^*, \mathcal{R}, \cdot) \in \mathsf{L}_{\mathsf{sign}}\ s.t.$
          $\{\mathsf{pk}_j\}_{j \in \mathcal{R}} \subseteq \{\mathsf{pk}_j\}_{j \in \mathcal{R}^*}$

6 :   **return** lose

// rule out trivial wins due to key duplication

7 :   **if** $\{\mathsf{pk}_j\}_{j \in \mathcal{R}^*} \cap \{\mathsf{pk}_j\}_{j \in \mathsf{L}_{\mathsf{corr}}} \neq \varnothing$

8 :   **return** lose

9 :   **if** $\mathsf{Verify}(\mu^*, \{\mathsf{pk}_j\}_{j \in \mathcal{R}^*}, \sigma^*) = \mathtt{reject}$

10 :   **return** lose

11 :   **return** win

$O\mathsf{Corrupt}(i)$

1 :   **if** $(i, \mathsf{pk}_i, \mathsf{sk}_i) \in \mathsf{L}_{\mathsf{keys}}$ **and** $\mathsf{sk}_i \neq \bot$

2 : $\mathsf{L}_{\mathsf{corr}} \leftarrow \mathsf{L}_{\mathsf{corr}} \cup \{i\}$

3 :   **return** $(\mathsf{pk}_i, \mathsf{sk}_i)$

4 :   **return** $\bot$ // if i has not been initialized.

$O\mathsf{KeyGen}(i, \mathsf{pk})$

// standard key generation for a new identifier i

1 :   **if** $\mathsf{pk} = \bot$

2 : $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathbf{ERS}.\mathsf{KeyGen}()$

3 : $\mathsf{L}_{\mathsf{keys}} \leftarrow \mathsf{L}_{\mathsf{keys}} \cup \{(i, \mathsf{pk}_i, \mathsf{sk}_i)\}$

4 :   **else**

// $\mathcal{A}$ over-writes an identifier with malicious keys

5 : $\mathsf{L}_{\mathsf{corr}} \leftarrow \mathsf{L}_{\mathsf{corr}} \cup \{i\}$

6 : $\mathsf{pk}_i \leftarrow \mathsf{pk}$

7 : $\mathsf{L}_{\mathsf{keys}} \leftarrow \mathsf{L}_{\mathsf{keys}} \cup \{(i, \mathsf{pk}_i, \bot)\}$

8 :   **return** pk_i

$O\mathsf{Sign}(\mu, \mathcal{R}, i)$

1 :   **if** $(i \in \mathsf{L}_{\mathsf{corr}} \vee i \notin \mathcal{R})$:   **return** $\bot$

// check that all keys in the query are initialized

2 :   **for all** $j \in \mathcal{R}$

3 :   **if** $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$

4 :   **return** $\bot$

5 : $\sigma \leftarrow \mathbf{ERS}.\mathsf{Sign}(\mu, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_i)$

6 : $\mathsf{L}_{\mathsf{sign}} \leftarrow \mathsf{L}_{\mathsf{sign}} \cup \{(\mu, \mathcal{R}, i)\}$

7 :   **return** σ

Fig. 2. Existential unforgeability under chosen message attack for (extendable) ring
signatures (security experiment and oracles). Our key generation oracle allows $\mathcal{A}$ to
register signers with arbitrary public keys (i.e., it also acts as a registration oracle).

Definition 3 (Unforgeability for ERS). *An extendable ring signature scheme ERS is said to be unforgeable if for all PPT adversaries $\mathcal{A}$ taking part in the unforgeability experiment (*cmEUF *in Fig. 2), the success probability is negligible, i.e.:* $\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbf{ERS}}^{\mathrm{cmEUF}}(\lambda) = \mathtt{win}\right] \leq \mathtt{negl}.$

Anonymous Extendability. For extendability, we consider security notions related to anonymity (thus the name anonymous extendability). We define an experiment that is general enough to support three different flavors of anonymous extendability: the standard *anonymity* notion, where no extension happens; *weak extendability*, where it is not possible to identify the original subring of an extended signature; and *strong extendability*, where it is not possible to tell what sequence of extensions a signature has undergone.

$\mathsf{Exp}_{\mathcal{A},\mathbf{ERS}}^{\mathrm{ANEXT}}(\lambda)$	$\mathsf{Chal}_b(\mu^*, \mathtt{lad}_0^*, \mathtt{lad}_1^*)$
$1: \quad b \leftarrow_R \{0,1\}$	$1: \quad \mathbf{parse}\ \mathtt{lad}_0^* = (i_0, \mathcal{R}_0^{(1)}, \ldots, \mathcal{R}_0^{(l_0)})$
$2: \quad \mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}} \leftarrow \varnothing$	$2: \quad \mathbf{parse}\ \mathtt{lad}_1^* = (i_1, \mathcal{R}_1^{(1)}, \ldots, \mathcal{R}_1^{(l_1)})$
$3: \quad \mathtt{pp} \leftarrow \mathbf{ERS}.\mathsf{Setup}(1^\lambda)$	// challenge signing keys should not be corrupted
// handle of oracles, for compact notation	$3: \quad \mathbf{if}\ i_0, i_1 \in \mathsf{L}_{\mathsf{corr}}\ \mathbf{return}\ \bot$
$4: \quad O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$	// sign and extend following the instructions
$5: \quad (\mu^*, \mathtt{lad}_0^*, \mathtt{lad}_1^*) \leftarrow \mathcal{A}^O(\mathtt{pp})$	// in both ladders
$6: \quad \bar{\sigma} \leftarrow \mathsf{Chal}_b(\mu^*, \mathtt{lad}_0^*, \mathtt{lad}_1^*)$	$4: \quad \sigma_0 \leftarrow \mathsf{Process}(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad}_0^*)$
$7: \quad b^* \leftarrow \mathcal{A}^O(\bar{\sigma})$	$5: \quad \sigma_1 \leftarrow \mathsf{Process}(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad}_1^*)$
// make sure $\mathcal{A}$ did not corrupt the challenge	$6: \quad \mathbf{if}\ \sigma_0 = \bot\ \mathbf{or}\ \sigma_1 = \bot\ \mathbf{return}\ \bot$
// keys during the second query phase	// check that ladders end with the same ring
$8: \quad \mathbf{if}\ i_0 \in \mathsf{L}_{\mathsf{corr}} \vee i_1 \in \mathsf{L}_{\mathsf{corr}}$	$7: \quad \mathbf{if}\ \mathcal{R}_0^{(1)} \cup \cdots \cup \mathcal{R}_0^{(l_0)} \neq \mathcal{R}_1^{(1)} \cup \cdots \cup \mathcal{R}_1^{(l_1)}$
$9: \quad\quad \mathbf{return}\ \mathtt{lose}$	$8: \quad\quad \mathbf{return}\ \bot$
$10: \quad \mathbf{if}\ b^* \neq b$	// set the challenge signature according to b
$11: \quad\quad \mathbf{return}\ \mathtt{lose}$	$9: \quad \bar{\sigma} \leftarrow \sigma_b$
$12: \quad \mathbf{return}\ \mathtt{win}$	$10: \quad \mathbf{return}\ \bar{\sigma}$

Fig. 3. Anonymity and anonymous extendability for extendable ring signatures. The oracles $O\mathsf{Sign}$, $O\mathsf{KeyGen}$ and $O\mathsf{Corrupt}$ are defined in Fig. 2.

For standard anonymity we consider adversaries that output ladders ($\mathtt{lad}_0^*, \mathtt{lad}_1^*$ in line 5 of $\mathsf{Exp}_{\mathcal{A},\mathbf{ERS}}^{\mathrm{ANEXT}}$ in Fig. 3) each containing only one ring. To avoid making the game trivial to win, the two rings need to be identical (line 7 of Chal_b). Moreover since the extension algorithm is never called ($l_0 = l_1 = 1$ in this case), it is clear that—with this restriction on the adversary's input to the challenger—our ANEXT experiment is the same as the standard anonymity one.

Definition 4 (Anonymity for ERS). *An extendable ring signature scheme is said to be anonymous if for all PPT adversaries $\mathcal{A}$ taking part in the anonymous extendability experiment (*ANEXT *in Figure 3) and submitting to the challenger ladders of the type* $\mathtt{lad}_0^* = (i_0, \mathcal{R}), \mathtt{lad}_1^* = (i_1, \mathcal{R})$, *it holds that the success probability of $\mathcal{A}$ is negligibly close to random guessing. i.e.,:*
$\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbf{ERS}}^{\mathrm{ANEXT}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}.$

For strong anonymous extendability, we consider adversaries that output any type of ladders that culminate in the same ring. In particular, we could have $l_0 \neq l_1$. Notice that strong anonymous extendability implies both weak anonymous extendability and standard anonymity.

Definition 5 (Strong Anonymous Extendability for ERS). *An extendable ring signature scheme is said to be strongly anonymous extendable if for all PPT adversaries $\mathcal{A}$ taking part in the anonymous extendability experiment (Fig. 3), it holds that:* $\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbf{ERS}}^{\mathrm{ANEXT}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}.$

We remark that strong extendability implies that the act of extending a ring signature is seamless, i.e., an adversary is not able to distinguish between a fresh ring signature (returned by Sign), and an extension of it (returned by Extend). This is covered in the strong extendability game for $l_0 = 1$ and $l_1 > 1$.

3.3   ERS from Signatures of Knowledge and Discrete Log

In what follows, we exhibit an efficient realization of extendable ring signature scheme from prime order groups and signatures of knowledge.

Our Construction in a Nutshell. The setup generates a prime-order group $\mathbb{G} = \langle g \rangle$, a random group element $H \leftarrow_R \mathbb{G}$ and public parameters for a SoK scheme for the relation

$$\mathscr{R}_\mathbb{G}\left(\phi = (h, \mathtt{pk}), w = x\right) = \left\{g^x = h \vee g^x = \mathtt{pk}\right\}.$$

Intuitively, $\mathscr{R}_\mathbb{G}$ requires that the witness be either the discrete log of $\mathtt{pk}$ (which is the corresponding secret key), or the element h. The signing procedure simply samples a random value $\mathtt{td} \leftarrow_R \mathbb{Z}_p$, creates an element $h := H \cdot g^{-\mathtt{td}}$ (which implies that $h \cdot g^{\mathtt{td}} = H$), and computes a signature of knowledge π for $(h, \mathtt{pk})$ using her secret key $\mathtt{sk}$. The signature σ contains $\mathtt{td}$, and a set $P = \{(h, \mathtt{pk}, \pi)\}$. Extending works essentially like signing, except that the extender uses the other kind of witness. Concretely, the extender samples a new $\mathtt{td}'$, computes $h' = g^{\mathtt{td}'}$ and a signature of knowledge π' for the $\mathtt{pk}'$ she wishes to add to the ring, using $\mathtt{td}'$ as the witness. The tuple $(h', \mathtt{pk}', \pi')$ is added to P, and $\mathtt{td}$ is replaced by $\mathtt{td} - \mathtt{td}'$. The verification checks that $H = g^{\mathtt{td}} \cdot \prod h_i$ for all h_i present in P, and that all π_i verify. This ensures that at least one of the π_i was produced using $\mathtt{sk}_i$ as a witness (otherwise we would be able to extract $dlog(H)$). A formal description of this construction is given in Fig. 4.

ERS.Setup$(1^\lambda) \mapsto$ pp

1: $(p, g, \mathbb{G}) \leftarrow$ GroupGen(1^λ)

2: $\mathbf{SoK}.\mathrm{pp} \leftarrow \mathbf{SoK}.\mathsf{Setup}(1^\lambda, \mathscr{R}_\mathbb{G})$

3: $H \leftarrow_R \mathbb{G}$

4: **return** pp := $(\mathbf{SoK}.\mathrm{pp}, g, H)$

ERS.KeyGen$() \mapsto$ (pk, sk)

1: sk $\leftarrow \mathbb{Z}_p$

2: pk := g^{sk}

3: **return** (sk, pk)

ERS.Sign$(\mu, \mathrm{sk}) \mapsto \sigma$

1: td $\leftarrow_R \mathbb{Z}_p$

2: $h := H \cdot g^{-\mathrm{td}}$

// signer does not know $dlog(h)$

// compute the signature

3: nonce $\leftarrow_R \{0,1\}^\lambda$

4: $\phi := (h, \mathrm{pk})$

5: $w := \mathrm{sk}$

6: $\pi \leftarrow \mathbf{SoK}.\mathsf{Sign}((\mathrm{nonce}, \mu), \mathscr{R}, \phi, w)$

7: $P := \{(h, \mathrm{pk}, \pi)\}$

8: **return** $\sigma := (\mathrm{nonce}, \mathrm{td}, P)$

ERS.Verify$(\mu, \{\mathrm{pk}_j\}_{j \in \mathcal{R}}, \sigma) \mapsto$ accept/reject

1: **parse** $\sigma = (\mathrm{nonce}, \mathrm{td}, P = \{(h_i, \mathrm{pk}_i, \pi_i)\}_{i \in \mathcal{R}'})$

2: **if** $H \neq g^{\mathrm{td}} \cdot \prod_{i \in \mathcal{R}'} h_i$: **return** reject

3: **if** $\{\mathrm{pk}_j\}_{j \in \mathcal{R}} \neq \{\mathrm{pk}_i\}_{i \in \mathcal{R}'}$:

      return reject

4: **for** $i \in \mathcal{R}'$:

      $\phi_i := (h_i, \mathrm{pk}_i)$

      if $\mathbf{SoK}.\mathsf{Verify}((\mathrm{nonce}, \mu), \mathscr{R}, \phi_i, \pi_i) =$ reject :

         return reject

5: **return** accept

ERS.Extend$(\mu, \{\mathrm{pk}_j\}_{j \in \mathcal{R}}, \sigma, \mathrm{pk}) \mapsto \sigma'$

1: **if** pk $\in \{\mathrm{pk}_j\}_{j \in \mathcal{R}}$: **return** $\perp$

2: **parse** $\sigma = (\mathrm{nonce}, \mathrm{td}, P = \{(h_i, \mathrm{pk}_i, \pi_i)\}_{i \in \mathcal{R}'})$

3: $\mathrm{td}' \leftarrow_R \mathbb{Z}_p$ // pick a trapdoor

4: $\mathrm{td} \leftarrow \mathrm{td} - \mathrm{td}' \pmod{p}$

5: $h := g^{\mathrm{td}'}$ // to simulate using the trapdoor

6: $\phi := (h, \mathrm{pk}), w := \mathrm{td}'$

7: $\pi \leftarrow \mathbf{SoK}.\mathsf{Sign}((\mathrm{nonce}, \mu), \mathscr{R}, \phi, w)$

8: add (h, pk, π) to P // update the signature

9: **return** $\sigma' := (\mathrm{nonce}, \mathrm{td}, P)$

Fig. 4. Extendable ring signatures from signature of knowledge and discrete log. The relation used by the SoK scheme is $\mathscr{R}_\mathbb{G} = \{(\phi, w) = (h, \mathrm{pk}, x) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p : g^x = h \vee g^x = \mathrm{pk}\}$.

Theorem 1. *Assuming that* **SoK** *is a secure signature of knowledge scheme, and that the discrete log problem is hard in the group* $\mathbb{G}$, *then the scheme* **ERS** $=$ (Setup, KeyGen, Sign, Verify, Extend) *described in Fig. 4 is an extendable ring signature scheme that satisfies* correctness *(Definition 1),* unforgeability *(Definition 3), and* strong anonymous extendability *(Definition 5).*

Proof. The correctness of the construction follows by inspection.

Unforgeability. To prove unforgeability, we present a sequence of hybrid games at the end of which the reduction is able to extract a solution to a discrete logarithm challenge from $\mathcal{A}$'s forgery with high-enough probability. Essentially this involves: embedding a discrete logarithm into H; moving to the simulatable setup for the SoK; replacing all signatures of knowledge with simulated ones; and using the witness extracted from π^* to learn $dlog(H)$. Due to space limitation all details are deferred to the full version of this paper.

Anonymous Extendability. To prove the strong anonymous extendability of our construction it suffices to show that if an adversary $\mathcal{A}$ can successfully break

anonymous extendability, we can build a reduction $\mathcal{B}$ that breaks the security of the signature of knowledge. Imagine that $\mathcal{B}$, playing the role of the challenger, runs the simulated setup for the signature of knowledge, instead of the real setup. This gives $\mathcal{B}$ a trapdoor that allows it to simulate signatures without knowledge of a witness. $\mathcal{B}$ uses this trapdoor to simulate all signatures of knowledge in response to signing queries from $\mathcal{A}$. $\mathcal{B}$ generates the challenge signature with no reference to the ladders. It simply chooses $\mathtt{td}$ at random, generates the h_i's as random values such that $g^{\mathtt{td}} \cdot \prod h_i = H$, and uses the trapdoor to simulate all signatures of knowledge. If $\mathcal{A}$ can distinguish $\mathcal{B}$ from an honest challenger, $\mathcal{B}$ can use $\mathcal{A}$ to break the simulatability property of the signature of knowledge. If $\mathcal{A}$ cannot distinguish $\mathcal{B}$ from an honest challenger, since $\mathcal{B}$'s behavior does not depend on choice of b, $\mathcal{A}$ cannot possibly win the anonymous extendability game with probability non-negligibly more than half. $\square$

4   Same-Message Linkable Extendable Ring Signatures

A same-message linkable ring signature scheme (SMLRS) is a ring signature scheme that additionally allows any third party to publicly identify (link) whether two signatures were generated by the same signer for the same message. This means that if the same party signs the same message twice, even for different rings, the two signatures can be linked by any third party.

In what follows, we introduce the notion of *extendable* same-message linkable ring signatures (ESMLRS). We give a security model for this new primitive, and describe an instantiation that builds on our ERS construction from Sect. 3.3.

4.1   Syntax

A same-message linkable extendable ring signature scheme is a tuple of six algorithms **SMLERS** $=$ (Setup, KeyGen, Sign, Verify, Extend, Link). The first five algorithms are inherited from extendable ring signatures. The Link algorithm (described below) allows any verifier to determine whether two signatures on a particular message were produced by the same signer.

Link$(\mu, (\sigma_0, \{\mathtt{pk}_j\}_{j \in \mathcal{R}_0}), (\sigma_1, \{\mathtt{pk}_j\}_{j \in \mathcal{R}_1})) \to \{\mathtt{linked}, \mathtt{unlinked}\}$: An algorithm that takes a message μ, two signatures (σ_0, σ_1) and two sets of public keys belonging to members of the rings $\mathcal{R}_0, \mathcal{R}_1$. It outputs $\mathtt{linked}$ if σ_0 and σ_1 were produced by the same signer, and $\mathtt{unlinked}$ otherwise.

We remark that Link does not necessarily reveal the identity of the common signer if signatures are linked. Next we discuss correctness for extendable same-message linkable ring signature schemes, which encompasses two statements: *extended signatures verify*, which is inherited from correctness for extendable ring signatures (Definition 1); and *extended signatures from different signers are unlinked*, which we formalize in the following definition.

Definition 6 (Cross-Signer Correctness for SMLERS). *For all security parameters $\lambda \in \mathbb{N}$, for any message $\mu \in \{0,1\}^*$, for any two ladders $\texttt{lad}_0 = (i_0, \mathcal{R}_0^{(1)}, \ldots, \mathcal{R}_0^{(l_0)})$, $\texttt{lad}_1 = (i_1, \mathcal{R}_1^{(1)}, \ldots, \mathcal{R}_1^{(l_0)})$ where $i_0 \in \mathcal{R}_0^{(1)}$, $i_1 \in \mathcal{R}_1^{(1)}$, $l_0 > 0$, $l_1 > 0$ and $i_0 \neq i_1$, it must hold that:*

$$\Pr \left[\begin{array}{c} \mathsf{Link}(\mu, (\sigma_0, \{\texttt{pk}_j\}_{j \in \mathcal{R}_0}), \\ (\sigma_1, \{\texttt{pk}_j\}_{j \in \mathcal{R}_1})) \to \texttt{unlinked} \end{array} \middle| \begin{array}{l} \mathcal{R}_0 = \mathcal{R}_0^{(1)} \cup \cdots \cup \mathcal{R}_0^{(l_0)} \\ \mathcal{R}_1 = \mathcal{R}_0^{(1)} \cup \cdots \cup \mathcal{R}_1^{(l_1)} \\ \texttt{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{L_{keys}} \leftarrow \{\mathsf{KeyGen}()\}_{j \in \mathcal{R}_0 \cup \mathcal{R}_1} \\ \sigma_0 \leftarrow \mathsf{Process}(\mu, \mathsf{L_{keys}}, \texttt{lad}_0) \\ \sigma_1 \leftarrow \mathsf{Process}(\mu, \mathsf{L_{keys}}, \texttt{lad}_1) \end{array} \right] = 1 - \texttt{negl}$$

where $\mathsf{Process}$ *is the algorithm described in Fig. 1 except that the ERS algorithms are replaced with the corresponding* **SMLERS** *ones.*

Remark 2. To build some intuition that may come in handy for understanding the security model, the reader might consider the following natural strategy for constructing an extendable same-message linkable ring signature scheme: ensuring that (part of) the signature is unique for every public key and message pair. In other words, the signer's public key and the signed message uniquely determine a part of the ring signature; we will refer to this part as the *linkability tag*. This tag is not modified by ring extensions and can be used to identify if two ring signatures, on the same message, were produced by the same signer simply by checking whether they share the same tag.

4.2   Security Model

A same-message linkable extendable ring signature scheme is an extendable ring signature that additionally satisfies the following properties:

Same-Message One-More Linkability: no set of $(t-1)$ corrupt signers can produce t signatures for the same message which appear pairwise unlinked. (We present this property in Definition 8).

Cross-Message Unlinkability: no adversary can determine whether two signatures for different messages were produced by the same signer. (We present this property in Definition 9).

Definition 7 (Secure SMLERS). *A same-message linkable extendable ring signature scheme (ESMLRS) is* secure *if it satisfies correctness, same-message one-more linkability (Definition 8, which implies unforgeability), and cross-message unlinkability (Definition 9).*

Definition 8 (Same-Message One-more Linkability for SMLERS). *A same-message linkable extendable ring signature scheme ESMLRS is said to be one-more linkable if for all PPT adversaries $\mathcal{A}$ taking part in the same-message one-more linkability experiment ($\mathsf{Exp}_{\mathcal{A}, \textbf{SMLERS}}^{\text{omlink}}(\lambda)$ depicted in Fig. 5), it holds that:* $\Pr[\mathsf{Exp}_{\mathcal{A}, \textbf{SMLERS}}^{\text{omlink}}(\lambda) = \texttt{win}] \leq \texttt{negl}.$

$\mathsf{Exp}_{\mathcal{A},\mathbf{SMLERS}}^{\mathrm{omlink}}(\lambda)$

1 : $\quad$ pp $\leftarrow$ Setup(1^λ)

2 : $\quad$ L$_{\mathsf{keys}}$, L$_{\mathsf{corr}}$, L$_{\mathsf{sign}} \leftarrow \varnothing$

3 : $\quad$ $O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$

4 : $\quad$ $(\mu^*, \{(\sigma_k^*, \mathcal{R}_k^*)\}_{k \in [1,\ldots,t]}) \leftarrow \mathcal{A}^O(\mathsf{pp})$

$\quad$ // $\mathcal{A}$ has never seen a signature for the message and a subring of the forgery rings

5 : $\quad$ **if** $\exists\, (\mu^*, \mathcal{R}, \cdot) \in$ L$_{\mathsf{sign}}$ $s.t.$ $\mathcal{R} \subseteq \mathcal{R}_k^*$ for some $k \in [1,\ldots,t]$ $\quad$ **return lose**

$\quad$ // $\mathcal{A}$ holds at most $t-1$ secret keys, among the keys identified by the forgery rings

6 : $\quad$ **if** $|(\mathcal{R}_1^* \cup \cdots \cup \mathcal{R}_t^*) \cap$ L$_{\mathsf{corr}}| \geq t$ $\quad$ **return lose**

$\quad$ // all the signatures in the forgery verify (for the same message)

7 : $\quad$ **if** $\exists\, k \in [1\ldots,t]$ s.t. Verify$(\mu^*, \{\mathrm{pk}_j\}_{j \in \mathcal{R}_k^*}, \sigma_k^*) =$ reject $\quad$ **return lose**

$\quad$ // all signatures in the forgery are unlinked (here $k, l \in [1,\ldots,t]$)

8 : $\quad$ **if** $\exists\, k \neq l$ s.t. Link$(\mu^*, (\sigma_k^*, \{\mathrm{pk}_j\}_{j \in \mathcal{R}_k^*}), (\sigma_l^*, \{\mathrm{pk}_j\}_{j \in \mathcal{R}_l^*})) =$ linked

9 : $\quad\quad$ **return lose**

10 : $\quad$ **return win**

Fig. 5. Security experiment for same-message one-more linkability. The signing, key generation and corruption oracles are as defined in Fig. 2, except that the algorithms for **ERS** are replaced with the corresponding algorithms for **SMLERS**. We recall that the list L$_{\mathsf{sign}}$ of sign-queries contains elements of the form $(\mu, \mathcal{R}, i)$.

Definition 9 (Cross-Message Unlinkability for SMLERS). *A same-message linkable extendable ring signature scheme ESMLRS is said to be cross-message unlinkable if for all PPT adversaries $\mathcal{A}$ taking part in the cross-message unlinkability experiment $(\mathsf{Exp}_{\mathcal{A},\mathbf{SMLERS}}^{\mathrm{cmunlink}}(\lambda)$ depicted in Fig. 6), it holds that the success probability of $\mathcal{A}$ is negligibly close to random guessing, i.e.,:*
$\Pr[\mathsf{Exp}_{\mathcal{A},\mathbf{SMLERS}}^{\mathrm{cmunlink}}(\lambda) = \mathtt{win}] \leq \frac{1}{2} + \mathtt{negl}.$

4.3 SMLERS from Signatures of Knowledge and Discrete Log

Our ESMLRS construction builds on the ERS construction in Fig. 4. Since the nuance is limited, we only briefly describe the tweaks needed to transform our ERS into an ESMLRS.

First, we adopt a slightly different relation $\mathscr{R}_{\mathbf{SMLERS}}$:

$$\mathscr{R}_{\mathbf{SMLERS}}\,(\phi = (h, \mathrm{pk}, g', \tau), w = x) = \{g^x = h \vee (g^x = \mathrm{pk} \wedge (g')^x = \tau)\}$$

Notably, the last *AND* not only requires a signer to prove knowledge of the secret key, but it also enforces that the same secret key is used to generate the linkability tag τ. The signatures of knowledge for ESMLRS are with respect to the new relation $\mathscr{R}_{\mathbf{SMLERS}}$.

$\mathsf{Exp}_{\mathcal{A},\mathbf{LRS}}^{\text{cmunlink}}(\lambda)$	$\mathsf{Chal}_b(\{\mu_0, \mathcal{R}_0, i_0\}, \{\mu_1, \mathcal{R}_1, i_1\})$
1 : $\quad b \leftarrow_R \{0,1\}, \mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}} \leftarrow \varnothing$	// the challenge identities must be uncorrupted
2 : $\quad \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	1 : $\quad$ **if** $i_0 \in \mathsf{L_{corr}} \vee i_1 \in \mathsf{L_{corr}}$
3 : $\quad O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$	2 : $\quad\quad$ **return** $\bot$
4 : $\quad (\{\mu_0, \mathcal{R}_0, i_0\}, \{\mu_1, \mathcal{R}_1, i_1\}) \leftarrow \mathcal{A}^O(\mathsf{pp})$	// one identity needs to be in both rings
5 : $\quad (\bar{\sigma}_0, \bar{\sigma}_1) \leftarrow \mathsf{Chal}_b(\{\mu_0, \mathcal{R}_0, i_0\}, \{\mu_1, \mathcal{R}_1, i_1\})$	3 : $\quad$ **if** $i_0 \notin \mathcal{R}_0 \cap \mathcal{R}_1 \vee i_1 \notin \mathcal{R}_1$
6 : $\quad b^* \leftarrow \mathcal{A}^O(\bar{\sigma}_0, \bar{\sigma}_1)$	4 : $\quad\quad$ **return** $\bot$
// Rule out corruption of challenge identities	// signing keys must exist
7 : $\quad$ **if** $i_0 \in \mathsf{L_{corr}} \vee i_1 \in \mathsf{L_{corr}}$ **return** lose	5 : $\quad$ **if** $\nexists\, (i_0, \mathsf{pk}_{i_0}, \mathsf{sk}_{i_0}) \in \mathsf{L_{keys}}$ **return** $\bot$
// Rule out trivial attacks using Link	6 : $\quad$ **if** $\nexists\, (i_1, \mathsf{pk}_{i_1}, \mathsf{sk}_{i_1}) \in \mathsf{L_{keys}}$ **return** $\bot$
8 : $\quad$ **if** $\mu_0 = \mu_1$ **return** lose	// generate a signature
// Rule out trivial attacks using Link	7 : $\quad \bar{\sigma}_0 \leftarrow \mathbf{LRS}.\mathsf{Sign}(\mu_0, \{\mathsf{pk}_i\}_{i \in \mathcal{R}_0}, \mathsf{sk}_{i_0})$
9 : $\quad$ **if** $(\mu_0, \cdot, i_0) \in \mathsf{L_{sign}} \vee (\mu_0, \cdot, i_1) \in \mathsf{L_{sign}}$	// generate the second signature according to
$\quad\quad \vee (\mu_1, \cdot, i_0) \in \mathsf{L_{sign}} \vee (\mu_1, \cdot, i_1) \in \mathsf{L_{sign}}$	// the experiment's bit b
10 : $\quad\quad$ **return** lose	8 : $\quad \bar{\sigma}_1 \leftarrow \mathbf{LRS}.\mathsf{Sign}(\mu_1, \{\mathsf{pk}_i\}_{i \in \mathcal{R}_1}, \mathsf{sk}_{i_b})$
11 : $\quad$ **if** $b^* \neq b$ **return** lose	9 : $\quad$ **return** $(\bar{\sigma}_0, \bar{\sigma}_1)$
12 : $\quad$ **return** win	

Fig. 6. Cross-message unlinkability. The signing, key generation and corruption oracles are as defined in Fig. 2, except that the **ERS** algorithms are substituted with the respective **SMLERS** variants.

Second, we modify the Sign algorithm of our ERS in Figure 4 so that it additionally computes $g' := \mathsf{H}(\mu)$ and $\tau := (g')^{\mathsf{sk}}$ for some hash function H, and it includes the linkability tag τ as part of the signature. Finally, the algorithm Link simply compares the linkability tags in the two signatures. It returns `linked` if they are equal, and `unlinked` otherwise.

This scheme can be shown to be same-message one-more linkable (resp. cross-message unlinkable) with only minor modifications to the proof of unforgeability (resp. anonymous extendability) of the extendable ring signature scheme.

5   Extendable Threshold Ring Signatures

Like a traditional threshold ring signature scheme, an *extendable* threshold ring signature scheme enables parties to produce a signature on a message μ for a ring $\mathcal{R}$ showing that at least t of the $|\mathcal{R}|$ potential signers in the ring participated, without revealing which. An extendable threshold ring signature scheme additionally has the following properties:

Flexibility: Given any two threshold signatures σ_0 and σ_1 that verify for the same message μ and for the same ring $\mathcal{R}$, anyone can *non-interactively* combine the signatures to obtain σ. The new signature σ is also a threshold ring signature and its threshold is equal to the total number of unique signers

who contributed to at least one of the two signatures. This functionality is
provided by the Combine algorithm (below).

Extendability: Given a signature σ on a message μ for the ring $\mathcal{R}$ with thresh-
old t, anyone can *non-interactively* transform σ into a signature σ' on the
same message with the same threshold , but for a larger ring $\mathcal{R}' \supseteq \mathcal{R}$. This
functionality is provided by Extend (see below).

5.1 Syntax

A non-interactive extendable threshold ring signature scheme (**ETRS**) is defined
as a tuple of six PPT algorithms **ETRS** = (Setup, KeyGen, Sign, Verify, Combine,
Extend), where the public parameters pp produced by Setup are implicitly avail-
able to all other algorithms:

Setup$(1^\lambda) \to$ pp: Takes a security parameter λ and outputs a set of public param-
eters pp.

KeyGen$() \to$ (pk, sk): Generates a new public and secret key pair.

Sign$(\mu, \{\mathtt{pk}_i\}_{i \in \mathcal{R}}, \mathtt{sk}) \to \sigma$: Returns a signature with threshold $t = 1$ using the
secret key sk corresponding to a public key $\mathtt{pk}_i$ with $i \in \mathcal{R}$.

Verify$(t, \mu, \{\mathtt{pk}_i\}_{i \in \mathcal{R}}, \sigma) \to$ accept/reject: Verifies a signature σ for the message
μ against the public keys $\{\mathtt{pk}_i\}_{i \in \mathcal{R}}$ with threshold t.

Combine$(\mu, \sigma_0, \sigma_1, \{\mathtt{pk}_i\}_{i \in \mathcal{R}}) \mapsto \sigma'$: Combines two signatures σ_0, σ_1 for the same
ring $\mathcal{R}$ into a signature σ' with threshold $t = |S_0 \cup S_1|$ where S_0, S_1 is the
set of (hidden) signers for σ_0 and σ_1 respectively.

Extend$(\mu, \sigma, \{\mathtt{pk}_i\}_{i \in \mathcal{R}}, \{\mathtt{pk}_i\}_{i \in \mathcal{R}'}) \mapsto \sigma'$: Extends the signature σ with threshold
t for the ring $\mathcal{R}$ into a new signature σ' with threshold t for the larger ring
$\mathcal{R} \cup \mathcal{R}'$.

For a somewhat more interactive syntax, we can replace 'Sign&Combine' exe-
cutions with a Join operation (described in Sect. 2.1). For the sake of formalism,
we present our security model only for schemes with Combine and defer the dis-
cussion on how to handle Join operations to the Sect. 5.4, where we present a
construction that uses the Join operation from signatures of knowledge and the
discrete log problem.

For the following definitions, we use ladders lad in a slightly different way
than we did in the context of extendable ring signatures (Sect. 3). We generalize
lad to support arbitrary sequences of actions that could lead to a valid threshold
ring signature (on some fixed message). lad will contain a sequence of tuples of
the form (action, input). The first component, action, can take on the values
Sign, Combine, or Extend. If action = Sign, we expect input = $(\mathcal{R}, i)$, where
$\mathcal{R}$ and i are the ring and signer identity with which the signature should be
produced. If action = Combine, we expect input = $(l_1, l_2, \mathcal{R})$, where l_1 and l_2
are indices of two signatures under the same ring $\mathcal{R}$. If action = Extend, we
expect input = $(l', \mathcal{R})$, where l' is the index of an existing signature which we
will extended to $\mathcal{R}$.

For use in our definitions, we define an algorithm Process$(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad})$, which
processes all of the operations in lad on the message μ (using keys stored in the

list $\mathsf{L_{keys}}$) and returns $(\sigma, t, \mathcal{R})$: the signature returned by the last operation of lad, the corresponding threshold, and the ring that σ verifies under. We define lad.sr to be the union of all identities and rings in lad. (sr stands for super-ring.)

We give a formal description of Process in the full version of this paper.

Definition 10 (Correctness for ETRS). *For correctness, we require that for all ladders* lad*, the signature returned by* Process(lad) *verifies. Formally: for all security parameters* $\lambda \in \mathbb{N}$*, for any message* $\mu \in \{0,1\}^*$*, for any ladder* lad *of polynomial size identifying a ring* $\mathcal{R} := \text{lad.sr}$ *of public-key identifiers, for any chosen threshold value* $1 \leq t \leq |\mathcal{R}|$*, it holds:*

$$\Pr\left[\begin{array}{l}\mathsf{Verify}(t, \mu, \{\mathsf{pk}_i\}_{i \in \mathcal{R}}, \sigma) \\ = \mathtt{accept} \ OR \ \sigma = \bot\end{array} \middle| \begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{L_{keys}} \leftarrow \{\mathsf{KeyGen}()\}_{j \in \text{lad.sr}} \\ (\sigma, t, \mathcal{R}) \leftarrow \mathsf{Process}(\mu, \mathsf{L_{keys}}, \text{lad})\end{array}\right] = 1.$$

5.2 Security Model

Our security definitions are loosely based on the ones given for threshold ring signatures by Munch-Hansen *et al.* [22].

Definition 11 (Secure ETRS). *An extendable threshold ring signature scheme is secure if it satisfies correctness (Definition 10), unforgeability (Definition 12), anonymity (Definition 13), and some notion of anonymous extendability.*

$\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathbf{ETRS}}(\lambda)$

1 : $\mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}} \leftarrow \varnothing$

2 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$

3 : $O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$

4 : $(t^*, \mu^*, \mathcal{R}^*, \sigma^*) \leftarrow \mathcal{A}^O(\mathsf{pp})$

5 : $q \leftarrow |\{(\mu^*, \mathcal{R}, \cdot) \in \mathsf{L_{sign}} \ s.t. \ \mathcal{R} \subseteq \mathcal{R}^*)\}|$

// rule out attacks if $\mathcal{A}$ knows too many sk:s or honestly generated signatures for μ^*

6 :   **if** $|\mathcal{R}^* \cap \mathsf{L_{corr}}| + q \geq t$ **return** lose

// rule out outputs that do not verify

7 :   **if** $\mathsf{Verify}(t, \mu^*, \{\mathsf{pk}_j\}_{j \in \mathcal{R}^*}, \sigma^*) = \mathtt{reject}$ **return** lose

8 :   **return** win

Fig. 7. Existential unforgeability under chosen message attack for (extendable) threshold ring signatures. The key generation, corruption and signing oracles are as in Fig. 2, with the difference that the **ERS** algorithms are substituted with the **ETRS** variants, and the signing oracle now returns partial signatures.

Definition 12 (Unforgeability for ETRS). *An extendable threshold ring signature scheme* **ETRS** *is said to be unforgeable if for all thresholds t, for all PPT adversaries $\mathcal{A}$ the success probability in the* cmEUF *experiment in Fig. 7 is* $\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbf{ETRS}}^{\mathrm{cmEUF}}(\lambda) = \mathtt{win}\right] \leq \mathtt{negl}.$

Just like for extendable ring signatures, the notion of anonymity for extendable threshold ring signatures captures scenarios where the adversary distinguishes fresh (not-extended) signatures, i.e., the challenge will be a threshold ring signature which has not be extended.

$\mathsf{Exp}_{\mathcal{A},\mathbf{ETRS}}^{\mathrm{ANEXT}}(\lambda)$	$\mathsf{Chal}_b(\mu^*, \mathtt{lad}_0^*, \mathtt{lad}_1^*)$
$1: \quad b \leftarrow_R \{0,1\}$	$1: \quad \mathbf{if}\ \mathtt{lad}_0^*\ \text{or}\ \mathtt{lad}_1^*\ \text{is not well-formed}$
$2: \quad \mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}} \leftarrow \varnothing$	$2: \quad\quad \mathbf{return}\ \bot$
$3: \quad \mathsf{pp} \leftarrow \mathbf{ETRS}.\mathsf{Setup}(1^\lambda)$	$3: \quad \mathbf{if}\ \exists i \in \mathtt{lad}_0^*.\mathsf{signers}\ s.t.\ i \in \mathsf{L}_{\mathsf{corr}}$
$4: \quad O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$	$4: \quad\quad \mathbf{return}\ \bot$
$5: \quad (\mu^*, \mathtt{lad}_0^*, \mathtt{lad}_1^*) \leftarrow \mathcal{A}^O(\mathsf{pp})$	$5: \quad \mathbf{if}\ \exists i \in \mathtt{lad}_1^*.\mathsf{signers}\ s.t.\ i \in \mathsf{L}_{\mathsf{corr}}$
$6: \quad \bar{\sigma} \leftarrow \mathsf{Chal}_b(\mu^*, \mathtt{lad}_0^*, \mathtt{lad}_1^*)$	$6: \quad\quad \mathbf{return}\ \bot$
$7: \quad b^* \leftarrow \mathcal{A}^O(\bar{\sigma})$	// make sure the public keys are known / initialized
$8: \quad \mathbf{if}\ \exists i \in \mathtt{lad}_0^*.\mathsf{signers}\ s.t.\ i \in \mathsf{L}_{\mathsf{corr}}$	$7: \quad \mathbf{if}\ \exists i \in \mathtt{lad}_0^*.\mathsf{sr}\ s.t.\ (\mathsf{pk}_i, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$
$9: \quad\quad \mathbf{return}\ \mathtt{lose}$	$8: \quad\quad \mathbf{return}\ \bot$
$10: \quad \mathbf{if}\ \exists i \in \mathtt{lad}_1^*.\mathsf{signers}\ s.t.\ i \in \mathsf{L}_{\mathsf{corr}}$	$9: \quad \mathbf{if}\ \exists i \in \mathtt{lad}_1^*.\mathsf{sr}\ s.t.\ (\mathsf{pk}_i, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$
$11: \quad\quad \mathbf{return}\ \mathtt{lose}$	$10: \quad\quad \mathbf{return}\ \bot$
$12: \quad \mathbf{if}\ \exists(\mu^*, \cdot, i) \in \mathsf{L}_{\mathsf{sign}}\ \text{for}\ i \in \mathtt{lad}_0^*.\mathsf{signers}$	$11: \quad (\sigma_0, t_0, \mathcal{R}_0) \leftarrow \mathsf{Process}(\mu^*, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad}_0)$
$13: \quad\quad \mathbf{return}\ \mathtt{lose}$	$12: \quad (\sigma_1, t_1, \mathcal{R}_1) \leftarrow \mathsf{Process}(\mu^*, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad}_1)$
$14: \quad \mathbf{if}\ \exists(\mu^*, \cdot, i) \in \mathsf{L}_{\mathsf{sign}}\ \text{for}\ i \in \mathtt{lad}_1^*.\mathsf{signers}$	// rule out trivial attacks
$15: \quad\quad \mathbf{return}\ \mathtt{lose}$	$13: \quad \mathbf{if}\ \mathcal{R}_0 \neq \mathcal{R}_1\ \text{or}\ t_0 \neq t_1$
$16: \quad \mathbf{if}\ b^* \neq b$	$14: \quad\quad \mathbf{return}\ \bot$
$17: \quad\quad \mathbf{return}\ \mathtt{lose}$	$15: \quad \bar{\sigma} \leftarrow \sigma_b$
$18: \quad \mathbf{return}\ \mathtt{win}$	$16: \quad \mathbf{return}\ \bar{\sigma}$

Fig. 8. Anonymity and anonymous extendability for extendable threshold ring signatures. The key generation, corruption and signing oracles are exactly as described in the unforgeability experiment (Fig. 7).

Definition 13 (Anonymity for ETRS). *An extendable threshold ring signature scheme is said to be anonymous if for all PPT adversaries $\mathcal{A}$ taking part in the anonymous extendability experiment (ANEXT in Figure 8) and submitting to the challenger two ladders with the structure explained below, it holds that the success probability of $\mathcal{A}$ is negligibly close to random guessing, i.e.:* $\Pr\left[\mathsf{Exp}_{\mathcal{A},\mathbf{ETRS}}^{\mathrm{ANEXT}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}.$

For anonymity, the ladders submitted by the adversary to the challenger have the following structure (here t denotes the threshold of the scheme): the first t

instructions are of the type $(\mathsf{Sign}, (\mathcal{R}, i))$, *where* $\mathcal{R}$ *is the same for all instructions in both ladders, and the signer indexes* i *are all distinct within the same ladder; the last* $(t-1)$ *instructions are of the type* $(\mathsf{Combine}, (l_1, l_2, \mathcal{R}))$, *where* $\mathcal{R}$ *is the same for all instructions in both ladders,* $l_1 = 1, 2, \ldots, t-1$, *and* $l_2 = t, t+1, \ldots, 2t-2$.

Our notion of anonymous extendability follows the gist of strong anonymity introduced in Sect. 3.2 for extendable ring signatures, but adapted to the threshold setting.

Definition 14 (Strong Anonymous Extendability for ETRS). *An extendable threshold ring signature scheme ETRS is said to be strongly anonymous extendable if for all PPT adversaries* $\mathcal{A}$ *taking part in the anonymous extendability experiment (*ANEXT *in Fig. 8) and submitting to the challenger ladders with the structure specified below, it holds that the success probability of* $\mathcal{A}$ *is negligibly close to random guessing, i.e.:* $\Pr\left[\mathsf{Exp}^{\mathrm{ANEXT}}_{\mathcal{A}_{\mathrm{sAnon}}, \mathbf{ERS}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + $ negl.

For strong anonymous extendability the adversary submits ladders that have the with the following structure: the first t *instructions are of the type* $(\mathsf{Sign}, (i, \mathcal{R}))$, *where the signer identities are pairwise distinct within a ladder, and the ring* $\mathcal{R}$ *is the same within the ladder (but possibly different for each ladder); the subsequent* $t-1$ *instructions are of the form* $(\mathsf{Combine}, (l_1, l_2, \mathcal{R}))$ *or* $(\mathsf{Extend}, (l', \mathcal{R}'))$, *where* l_1, l_2 *and* l' *denote indexes.*

Notably, in strong anonymous extendability each ladder may contain an arbitrary (polynomial, and possibly different for each ladder) number of subsequent Extend *instructions, so long the final one of each ladder culminates in the same ring.*

5.3   A Generic Compiler for ETRS from SMLERS

In what follows, we formalize the intuition given in Remark 2 (Section 4.1) on how to generically derive an extendable threshold ring signature scheme from any given same-message linkable extendable ring signature scheme. The compiler is detailed in Fig. 9.

Theorem 2. *Assuming that* **SMLERS** *is a secure same-message linkable extendable ring signature scheme, then the scheme* **ETRS** = $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extend}, \mathsf{Combine})$ *described in Fig. 9 is an extendable threshold ring signature scheme that satisfies* correctness *(Definition 10),* unforgeability *(Definition 12), and* anonymity *(Definition 13).*

We prove Theorem 2 in the full version of this paper.

5.4   ETRS from Signatures of Knowledge and Discrete Log

In what follows we present a somewhat more interactive Extendable Threshold Ring Signature Scheme that supports Join operations and enjoys more compact

Setup$(1^\lambda) \mapsto$ pp   KeyGen$() \mapsto$ (pk, sk)

---

return SMLERS.Setup(1^λ) **return SMLERS**.KeyGen$()$

Sign$(\mu, \text{sk}, \{\text{pk}_i\}_{i \in \mathcal{R}}) \mapsto \sigma$

---

return SMLERS.Sign$(\mu, \text{sk}, \{\text{pk}_i\}_{i \in \mathcal{R}})$

Extend$(\mu, \sigma, \{\text{pk}_i\}_{i \in \mathcal{R}_0}, \{\text{pk}_i\}_{i \in \mathcal{R}_1},) \mapsto \sigma'$

---

return $\{\textbf{SMLERS}.\text{Extend}(\{\text{pk}_i\}_{i \in \mathcal{R}_1}, \{\text{pk}_j\}_{j \in \mathcal{R}_2}, \sigma_i)\}_{\sigma_i \in \sigma}$

Combine$(\mu, \sigma_0, \sigma_1, \{\text{pk}_i\}_{i \in \mathcal{R}}) \mapsto \sigma'$

---

1 : $\sigma'_0 \leftarrow \{s_0 \in \sigma_0 \mid \forall s_1 \in \sigma_1 : \text{Link}(\mu, (s_0, \{\text{pk}_i\}_{i \in \mathcal{R}}), (s_1, \{\text{pk}_i\}_{i \in \mathcal{R}})) = \texttt{unlinked}\}$

2 :   **return** $\sigma'_0 \cup \sigma_1$

Verify$(t, \mu, \{\text{pk}_i\}_{i \in \mathcal{R}}, \sigma) \mapsto \texttt{accept/reject}$

---

1 :   Parse $\sigma = \{s_0, ..., s_\ell\}$ as a set of signatures // removing duplicates

2 :   **if** $|\sigma| < t$ **return reject**

3 :   **if** $\exists\, s_i \in \sigma : \text{Verify}(\mu, \{\text{pk}_i\}_{i \in \mathcal{R}}, s_i) = \texttt{reject}$ **return reject**

4 :   **if** $\exists\, (s_i, s_j) \in \sigma \times \sigma : s_i \neq s_j \,\wedge$

        $\text{Link}(\mu, (s_i, \{\text{pk}_i\}_{i \in \mathcal{R}}), (s_j, \{\text{pk}_i\}_{i \in \mathcal{R}})) = \texttt{linked}$ **return reject**

5 :   **return accept**

Fig. 9. Generic compiler for extendable threshold ring signatures from extendable same-message linkable ring signatures.

signatures. Concretely, the size of extended threshold signatures is independent of the threshold t, instead it grows linearly with n' (an upper bound on the ring size). This is an improvement compared to the compiler presented in Fig. 9, which if instantiated using our **SMLERS** from Signatures of Knowledge and Discrete Log of Sect. 4.3, returns signatures of size linear in $t \cdot |\mathcal{R}|$.

Our Construction in a Nutshell. Similarly to the ERS construction of Fig. 4, we work with a prime order group $\mathbb{G}$, with two public elements $g, H \in \mathbb{G}$ and a signature of knowledge for a relation $\mathscr{R}_\mathbb{G}$ for knowledge of the discrete logarithm either of a given value h or of a pk.

Let $n' \in \mathbb{N}$ be an upper bound on the ring size. We achieve the threshold functionality by leveraging features of polynomials in a similar way to Shamir secret sharing. Intuitively, the signer samples $n' > 0$ pairs of values $(x_i, \text{td}_i) \in \mathbb{Z}_p \times \mathbb{G}$. These pairs of values define a unique polynomial $f(x)$ of degree n' such that $f(0) = \text{dlog}_g(H)$ and $f(x_i) = \text{td}_i$ for every $i \in [n']$. Of course, since $\text{dlog}_g(H)$ is unknown, our signers don't know the coefficients of this polynomial. However, since polynomial interpolation involves only linear operations (when the

x-coordinates are fixed and known), the signers can interpolate this polynomial *in the exponent* to learn additional points $(\hat{x}, y = g^{f(\hat{x})})$ for any given $\hat{x}$. In order to sign, and later to endorse a statement (Join a signature), the signer is required to produce a signature of knowledge for $\mathscr{R}_{\mathbb{G}}$ for a random point $(\hat{x}, y = g^{f(\hat{x})})$ on the polynomial such that $\hat{x} \notin \{x_i\}_{i \in [n']}$. Crucially, the signer does not know the discrete log of y (i.e., $(\hat{x}, y)$ is not among the 'trapdoored' values (x_i, g^{td_i})), and thus must satisfy the second clause of the relation (proving knowledge of their secret key). On the other hand, to extend a signature, anyone can pick one of the (remaining) 'trapdoored' points (x_i, td_i), and generate a proof for $\mathscr{R}_{\mathbb{G}}$ by satisfying the first clause (proving knowledge of td_i), to include any pk in the ring. The pair (x_i, td_i) is then removed from the list of trapdoors. (In case the owner of pk later wants to join the signature, the Extend algorithm encrypts td_i to pk; later, the owner of pk can recover td_i and return it to the list of trapdoors before producing a fresh signature of knowledge using her secret key.)

The key idea of our construction is detailed in Fig. 10 (the PolySign subroutine employed in Sign and Join–where this is called using the signer's secret key as w and on a random value $\hat{x}$– and in Extend–where an evaluation point and its corresponding trapdoor are used as $\hat{x}$and w respectively).

For any field $\mathbb{F}$ (often implicit) and $\mathcal{X} \subseteq \mathbb{F}$, $j \in \mathcal{X}$, define the degree $|\mathcal{X}| - 1$ Lagrange polynomial $L_{(\mathcal{X}, j)}(X) := \prod_{m \in \mathcal{X} \setminus \{j\}} \frac{X - m}{j - m} \in \mathbb{F}[X]$.

Theorem 3. *Assuming that* **SoK** *is a secure signature of knowledge scheme, and that the discrete log problem is hard in the group* $\mathbb{G}$*, then the scheme*

PolySign$(P, T, \hat{x}, w, \mathsf{pk}, \mu) \to (y_{\hat{x}}, \pi)$

// compute values for the Lagrange interpolation

1 : $\mathcal{Z} := \{(0, H)\} \cup \{(x, g^{\mathsf{td}})\}_{(x, \mathsf{td}) \in T} \cup \{(x, y)\}_{(x, y, \mathsf{pk}, c, \pi) \in P}$

// compute evaluation points for the Lagrange interpolation

2 : $\mathcal{X} := \{x\}_{(x, y) \in \mathcal{Z}}$

// evaluate the polynomial on the input point $\hat{x}$

3 : $y_{\hat{x}} \leftarrow \prod_{(x, y) \in \mathcal{Z}} y^{L_{(\mathcal{X}, x)}(\hat{x})}$ // note: $dlog(y_{\hat{x}})$ is unknown because $dlog(H)$ is

4 : $\phi := (\hat{y}, \mathsf{pk}_s)$ // include new poly. value in the statement

// Lagrange interpolation in the exponent over the standard set $\{1, ..., n'\}$

5 : **for** $i \in [n']$: $V_i \leftarrow \prod_{(x, y) \in \mathcal{Z}} y^{L_{(\mathcal{X}, x)}(i)}$

6 : $\hat{\mu} := (\mu, \{V_i\}_{i \in [n']})$ // include a 'commitment' to the polynomial in the message

7 : $\pi \leftarrow \mathbf{SoK}.\mathsf{Sign}(\hat{\mu}, \mathscr{R}_{\mathbb{G}}, \phi, w)$

// note: w is given in input to the algorithm,$w = \mathsf{sk}$ for Sign & Join, otherwise $w = \mathsf{td}$

8 : **return** $(\hat{y}, \pi)$

Fig. 10. Subroutine used in our **ETRS** construction depicted in Fig. 11.

KeyGen() $\mapsto$ (pk, sk)

1 : $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathbf{ERS}.\mathsf{KeyGen}()$

2 : $(\mathrm{pk}_e, \mathrm{sk}_e) \leftarrow \mathbf{PKE}.\mathsf{KeyGen}()$

3 :   **return** $(\mathrm{pk} = (\mathrm{pk}_s, \mathrm{pk}_e), \mathrm{sk} = (\mathrm{sk}_s, \mathrm{sk}_e))$

Sign$(\mu, \mathrm{sk}) \mapsto \sigma$

1 : $X \leftarrow_R \binom{\mathbb{Z}_p^*}{n'}$ // pick n' distinct evaluation points

2 : $T := \varnothing; \quad P := \varnothing$

3 :   **for** $x \in X$

4 : $\mathrm{td} \leftarrow_R \mathbb{Z}_p$ // generate trapdoors for poly. values

5 : $T \leftarrow T \cup \{(x, \mathrm{td})\}$ // populate trapdoor set

6 : $c \leftarrow \mathsf{Enc}(\mathrm{pk}_e, \bot)$ // no info to pass on

7 : $\hat{x} \leftarrow_R \mathbb{Z}_p^* \setminus X$ // pick a new evaluation point

8 : $(y, \pi) \leftarrow \mathsf{PolySign}(P, T, \hat{x}, w := \mathrm{sk}, \mathrm{pk}, \mu)$

9 : $P := \{(\hat{x}, y, \mathrm{pk}_s, \pi, c)\}$

10 :   **return** $\sigma := (T, P)$

Join$(\mu, \{\mathrm{pk}_j\}_{j \in \mathcal{R}}, \mathrm{sk}, \sigma) \mapsto \sigma'$

// check if current signer's pk_s is in P

1 :   **if** $\exists (x, y, \mathrm{pk}, \pi, c) \in P$ s.t. $\mathrm{pk} = \mathrm{pk}_s$

// remove simulated proof for the signer who wants to join

2 : $P \leftarrow P \setminus \{(x, y, \mathrm{pk}_s, \pi, c)\}$

// retrieve trapdoor value

3 : $\mathrm{td} \leftarrow \mathsf{Dec}(\mathrm{sk}_e, c)$

// add eval. point and td to the set of available trapdoors

4 : $T \leftarrow T \cup \{(x, \mathrm{td})\}$

5 : $c' \leftarrow \mathsf{Enc}(\mathrm{pk}_e, \bot)$ // no info to pass on

6 : $\hat{x} \leftarrow_R \mathbb{Z}_p^* \setminus X$ // pick a new evaluation point

// interpolate a unique representation of the polynomial

7 : $(y', \pi') \leftarrow \mathsf{PolySign}(P, T, \hat{x}, w := \mathrm{sk}, \mathrm{pk}, \mu)$

8 : $P \leftarrow P \cup \{(\hat{x}, y', \mathrm{pk}_s, \pi', c')\}$

9 :   Randomly permute P

10 :   **return** $\sigma := (T, P)$

Extend$(\mu, \{\mathrm{pk}_j\}_{j \in \mathcal{R}}, \sigma, \mathrm{pk}) \mapsto \sigma'$

1 :   **if** $\mathrm{pk} \in \{\mathrm{pk}_j\}_{j \in \mathcal{R}} :$ **return** $\bot$

2 : $(\hat{x}, \hat{\mathrm{td}}) \leftarrow_R T$ // Pick eval-point and trapdoor

3 : $c' \leftarrow \mathsf{Enc}(\mathrm{pk}_e, \hat{\mathrm{td}})$ // enable future endorsing

// interpolate a unique representation of the polynomial

4 : $(y', \pi') \leftarrow \mathsf{PolySign}(P, T, \hat{x}, w := \hat{x}, \mathrm{pk}, \mu)$

5 : $T \leftarrow T \setminus \{(\hat{x}, \hat{\mathrm{td}})\}$ // erase used trapdoor

// Add simulated signature to the set of proofs

6 : $P \leftarrow P \cup \{(\hat{x}, y', \mathrm{pk}_s, \pi', c')\}$

7 :   Randomly permute P

8 :   **return** $\sigma' := (T, P)$

Verify$(t, \mu, \{\mathrm{pk}_j\}_{j \in \mathcal{R}}, \sigma) \mapsto \mathtt{accept}/\mathtt{reject}$

1 :   **if** $\{\mathrm{pk}_j\}_{j \in \mathcal{R}} \neq \{\mathrm{pk}_i\}_{(\cdot, \cdot, \mathrm{pk}_i, \cdot, \cdot) \in P} :$

2 :   **return** reject

// check y's are consistent with a degree n' polynomial

3 : $\mathcal{Z} := \{(0, H)\} \cup \{(x, g^{\mathrm{td}})\}_{(x, \mathrm{td}) \in T}$

4 : $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(x, y)\}_{(x, y, \mathrm{pk}, c, \pi) \in P}$

5 :   Pick $\hat{\mathcal{Z}} \subseteq \mathcal{Z}$ s.t. $|\hat{\mathcal{Z}}| = n' + 1$

6 : $\mathcal{X} := \{x\}_{(x, y) \in \mathcal{Z}}; \hat{\mathcal{X}} := \{x\}_{(x, y) \in \hat{\mathcal{Z}}}$

7 :   **for** $(x, y) \in \mathcal{Z} :$

8 :   **if** $y \neq \prod_{(\hat{x}, \hat{y}) \in \hat{\mathcal{Z}}} \hat{y}^{L_{(\hat{\mathcal{X}}, \hat{x})}(x)} :$ **return** reject

// Interpolation over the standard set $\{1, \ldots, n'\}$

9 :   **for** $i \in [n'] : \quad V_i \leftarrow \prod_{(x, y) \in \mathcal{Z}} y^{L_{(\mathcal{X}, x)}(i)}$

10 : $\hat{\mu} := (\mu, \{V_i\}_{i \in [n']})$

11 :   **for** $(x, y, \mathrm{pk}_s, \pi, c) \in P$ // check proofs individually

12 : $\phi := (y, \mathrm{pk}_s)$

13 :   **if** $\mathbf{SoK}.\mathsf{Verify}(\hat{\mu}, \mathscr{R}_{\mathbb{G}}, \phi, \pi) = \mathtt{reject}$

14 :   **return** reject

15 :   **if** $|T| + |P| \geq t + n'$ **return** accept

16 :   **else return** reject

Fig. 11. Extendable threshold ring signatures from signature of knowledge and hardness of discrete log. The Setup algorithm is the same as in the **ERS** construction of Fig. 4 (with $\mathscr{R}_{\mathbb{G}} = \{(\phi, w) = (h, \mathrm{pk}, x) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p : g^x = h \vee g^x = \mathrm{pk}\}$). In the description, $n' > 0$ denotes the maximum amount of times a signature can be extended (it can be set in pp, or chosen upon signing). We always let pk denote the public key corresponding to sk; any algorithm that is given sk as input implicitly has access to pk. The parsing of pk into $(\mathrm{pk}_s, \mathrm{pk}_e)$ (or of pk_i into $(\mathrm{pk}_{s,i}, \mathrm{pk}_{e,i})$), of sk into $(\mathrm{sk}_s, \mathrm{sk}_e)$ and of σ into (T, P) is done implicitly.

ETRS $=$ (Setup, KeyGen, Sign, Verify, Extend, Join) *described in Fig. 11 is an extendable threshold ring signature scheme that satisfies* correctness *(Definition 10),* unforgeability *(Definition 12), and* strong anonymous extendability *(Definition 13).*

We give a proof of Theorem 3 in the full version of this paper. We also describe how we modify the security model to account for the Join there.

Remark 3. Note that a malicious extender can prevent the newly added members of the ring from later joining a signature, simply by not encrypting the correct trapdoor under that new member's public key. This is not captured by our security definitions, but precluding such attacks would be an interesting and valuable extension. We can modify our construction to disallow this by adding a zero knowledge proof that the encrypted value is in fact the discrete log of the h in question.

6   Implementation Results

We have implemented the **ERS**, **SMLERS** and **ETRS** constructions, respectively from Sects. 3, 4.3 and 5, at the 128-bit security level within the RELIC[2] library. The choice of underlying group is the conservative `edwards25519` elliptic curve used in the Ed25519 signature scheme [2]. The benchmarking platform is an Intel Core i7-6700K Skylake @ 4 GHz, with HyperThreading and Turbo-Boost disabled. Each operation was executed 10^4 times for the smaller rings and 10^2 times for the larger ones. The average times for signature generation and verification, and signature sizes (without point compression) are shown in Figs. 12, 13 and 14, respectively. For ease of exposition, we combined the wall time for the initial signature generation and subsequent joinings or extensions in the plots. A specific binary built by running `make` in `relic/demo/ers-etrs` allows to reproduce our results.

ERS Benchmark. We benchmark our ERS implementation for ring sizes of 1 to 2^{11}. The performance depends on the ring size only, so the number of extensions is always the number of keys. We instantiate the SoK for the relation $\mathscr{R}_{\mathbf{ERS}}$ as a non-interactive Sigma protocol combining an OR-proof with proof of knowledge of the discrete logarithm embedding the message to be signed in the challenge computation.

ETRS Benchmark. We benchmark our ETRS implementation for thresholds of $1, 2, 4, 8$ and ring sizes of 1 to 2^{11}. For the ETRS construction, the quadratic cost of interpolation clearly dominates the signing, joining and verification steps; and explains the additional computational overhead in comparison to the ERS scheme.

ETRS from SMLERS Benchmark. Finally, we include the benchmarks of our generic compiler applied to our SMLERS scheme. We instantiate the SoK for the relation $\mathscr{R}_{\mathbf{SMLERS}}$ as another non-interactive Sigma protocol combining OR-proofs and discrete logarithm proofs by slightly rewriting the statement as $\{(g^x = h \vee g^x = \mathrm{pk}) \wedge (g^x = h \vee (g')^x = \tau)\}$, which allows us to share code with

---
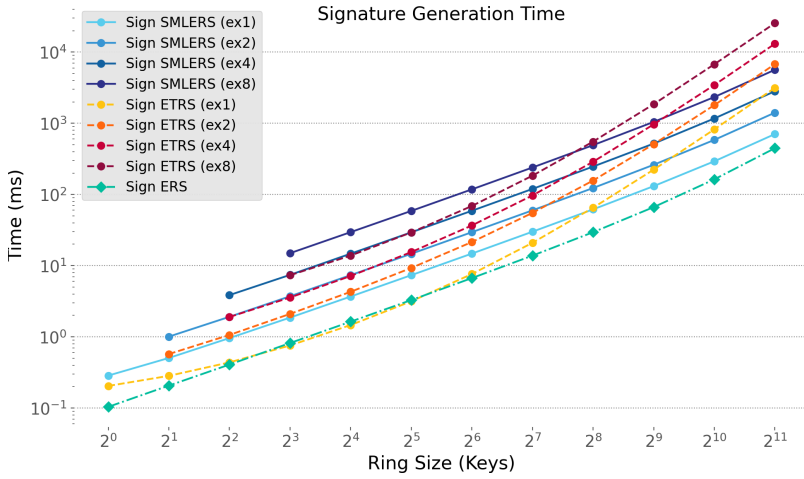
[2] https://github.com/relic-toolkit/relic

Fig. 12. Clock time for Sign in the three implemented schemes for different thresholds. The signature generation time includes the initial signature generation and subsequent joinings/extensions.
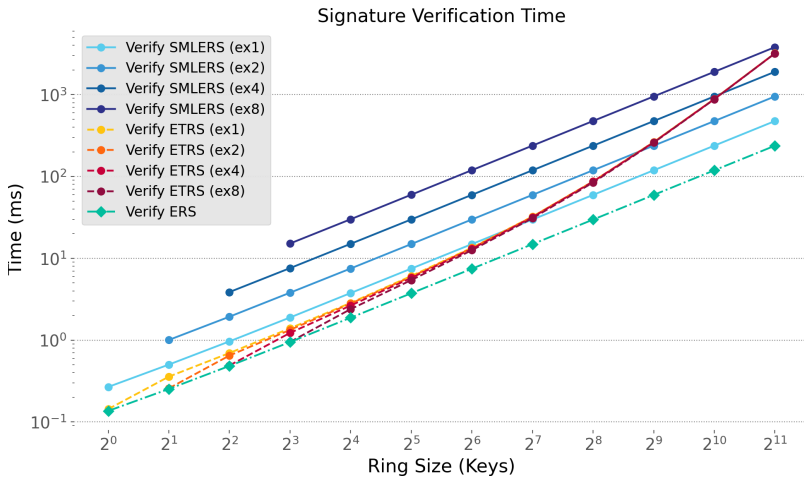

Fig. 13. Clock time for Verify in the three implemented schemes for different thresholds. The verification time is that of verifying the final extended signature.

the ERS implementation. In comparison with the ETRS scheme, the signature sizes are much larger; but the signature and verification times are more efficient for larger rings due to the cost of interpolation in the ETRS scheme.
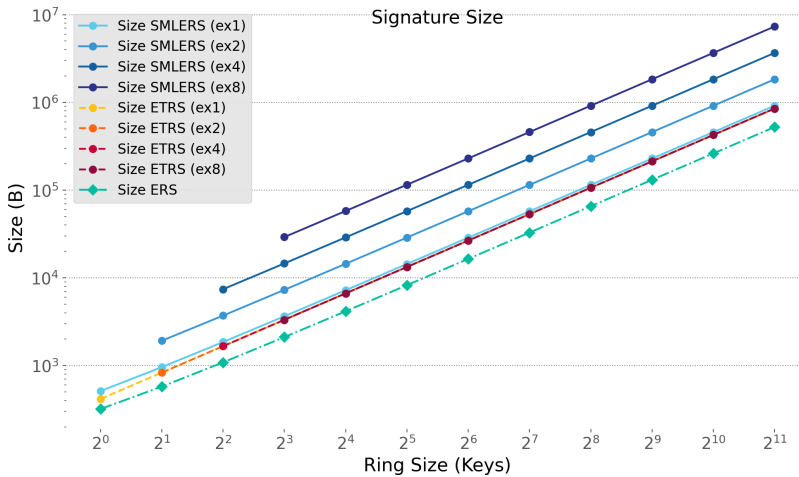
Fig. 14. Signature sizes for all the three implemented schemes, with varying thresholds. In the ETRS scheme, the signature size is independent of the threshold, while in ESMLRS there is a linear dependence.

References

1. Bender, A., Katz, J., Morselli, R.: Ring signatures: stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_4
2. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. J. Cryptogr. Eng. **2**(2), 77–89 (2012)
3. Bettaieb, S., Schrek, J.: Improved lattice-based threshold ring signature scheme. In: Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp. 34–51. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38616-9_3
4. Beullens, W., Katsumata, S., Pintore, F.: Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. Cryptology ePrint Archive, Report 2020/646 (2020). https://eprint.iacr.org/2020/646
5. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_26
6. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 243–265. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_13

7. Bose, P., Das, D., Rangan, C.P.: Constant size ring signature without random oracle. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, vol. 9144, pp. 230–247. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19962-7_14

8. Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_30

9. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 132–145. ACM Press (2004)

10. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73420-8_38

11. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006). https://doi.org/10.1007/11818175_5

12. Chow, S.S.M., Wei, V.K.W., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: Lin, F.C., Lee, D.T., Lin, B.S., Shieh, S., Jajodia, S. (eds.) ASIACCS 2006, pp. 297–302. ACM Press (2006)

13. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in *Ad Hoc* groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_36

14. Esgin, M.F., Steinfeld, R., Sakzad, A., Liu, J.K., Liu, D.: Short lattice-based one-out-of-many proofs and applications to ring signatures. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 67–88. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_4

15. Haque, A., Scafuro, A.: Threshold ring signatures: new definitions and post-quantum security. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 423–452. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_15

16. Liu, J.K.: Ring signature. In: Li, K.-C., Chen, X., Susilo, W. (eds.) Advances in Cyber Security: Principles, Techniques, and Applications, pp. 93–114. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-1483-4_5

17. Liu, J.K., Wong, D.S.: On the security models of (threshold) ring signature schemes. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 204–217. Springer, Heidelberg (2005). https://doi.org/10.1007/11496618_16

18. Liu, Z., Nguyen, K., Yang, G., Wang, H., Wong, D.S.: A lattice-based linkable ring signature supporting stealth addresses. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11735, pp. 726–746. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29959-0_35

19. Lu, X., Au, M.H., Zhang, Z.: Raptor: a practical lattice-based (linkable) ring signature. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 110–130. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_6

20. Malavolta, G., Schröder, D.: Efficient ring signatures in the standard model. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_5

21. Melchor, C.A., Cayrel, P.L., Gaborit, P., Laguillaumie, F.: A new efficient threshold ring signature scheme based on coding theory. IEEE Trans. Inf. Theory **57**(7), 4833–4842 (2011)

22. Munch-Hansen, A., Orlandi, C., Yakoubov, S.: Stronger notions and a more efficient construction of threshold ring signatures. Cryptology ePrint Archive, Report 2020/678 (2020). https://eprint.iacr.org/2020/678
23. Okamoto, T., Tso, R., Yamaguchi, M., Okamoto, E.: A k-out-of-n ring signature with flexible participation for signers. Cryptology ePrint Archive, Report 2018/728 (2018). https://eprint.iacr.org/2018/728
24. Patachi, Ş, Schürmann, C.: Eos a universal verifiable and coercion resistant voting protocol. In: Krimmer, R., Volkamer, M., Braun Binder, N., Kersting, N., Pereira, O., Schürmann, C. (eds.) E-Vote-ID 2017. LNCS, vol. 10615, pp. 210–227. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68687-5_13
25. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
26. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_12
27. Sun, S.-F., Au, M.H., Liu, J.K., Yuen, T.H.: RingCT 2.0: a compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10493, pp. 456–474. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66399-9_25
28. Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 384–398. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30556-9_30
29. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Threshold ring signature without random oracles. In: Cheung, B.S.N., Hui, L.C.K., Sandhu, R.S., Wong, D.S. (eds.) ASIACCS 2011, pp. 261–267. ACM Press (2011)
30. Zhang, F., Kim, K.: ID-based blind signature and ring signature from pairings. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_33