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; 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 signatures 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 [30,5,12], composite-order groups [26,7], non-interactive zero knowledge [6,20], and, most recently, quantum-safe isogenies and lattices [14,19,18,4].

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 $\mathtt{sk}_A$). (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 0.45s.

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 $\mathbf{RS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

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

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

$\mathsf{Sign}(\mu, \{\mathtt{pk}_j\}_{j\in\mathcal{R}}, \mathtt{sk}_i) \to \sigma$**:** 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.

$\mathsf{Verify}(\mu, \{\mathtt{pk}_i\}_{i\in\mathcal{R}}, \sigma) \to \mathtt{accept}/\mathtt{reject}$**:** Takes a message, a set of public keys of the users within a ring, and a signature σ. Outputs `accept` or `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 $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Combisign}, \mathsf{Verify})$. The algorithms $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}$ 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.

$\mathsf{Combisign}(\{\sigma_i\}_{i\in\mathcal{S}\subseteq\mathcal{R}}) \to \sigma$**:** 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.

$\mathsf{Join}(\mu, \{\mathtt{pk}_j\}_{j\in\mathcal{R}}, \mathtt{sk}, \sigma) \to \sigma'$**:** Takes a message μ, a set of public keys $\{\mathtt{pk}_j\}_{j\in\mathcal{R}}$, which includes the public key of the new signer, the new signer's secret key $\mathtt{sk}$, 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 $\mathbf{SoK} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{SimSetup}, \mathsf{SimSign})$:

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

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

$\mathsf{Verify}(\mu, \phi, \sigma) \to \mathtt{accept/reject}$**:** Takes as input a message μ, a statement ϕ, and a signature σ. Outputs $\mathtt{accept}$ if the the signature is valid, $\mathtt{reject}$ 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[4] — 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 $\mathsf{Process}(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad})$, that we describe in Figure 1. As the name suggests, this algorithm processes a ladder $\mathtt{lad}$ on a given message μ using keys from $\mathsf{L}_{\mathsf{keys}}$ (the list of generated keys). $\mathsf{Process}$ signs μ using $\mathtt{sk}_i$ under the ring $\mathcal{R}^{(1)}$, and extends the signature to all the subsequent rings (using keys stored in the list $\mathsf{L}_{\mathsf{keys}}$). $\mathsf{Process}$ returns an extendable ring signature σ, which is the output of the last operation.
For correctness, we require that any — possibly extended — signature σ output by $\mathsf{Process}$ verifies for the given message, under the final ring $\mathcal{R}^{(l)}$.

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

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

Parse $\mathtt{lad}$ as $(i, \mathcal{R}^{(1)}, \mathcal{R}^{(2)}, \ldots, \mathcal{R}^{(l)})$
if $i \notin \mathcal{R}^{(1)}$ **return** $\perp$ // make sure all public keys are in $\mathsf{L}_{\mathsf{keys}}$
for $j \in \mathcal{R}^{(1)} \cup \cdots \cup \mathcal{R}^{(l)}$: **if** $(j, \mathtt{pk}_j, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$ **return** $\perp$
// make sure the signer's secret key is available in $\mathsf{L}_{\mathsf{keys}}$
if $\mathtt{sk}_i = \perp$: **return** $\perp$ // make sure $\mathtt{sk}_i$ is not corrupted
// process the instructions in the ladder
$\sigma^{(1)} \leftarrow \mathbf{ERS}.\mathsf{Sign}(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}^{(1)}}, \mathtt{sk}_i)$
for $l' \in [2, \ldots, l]$:
  $\mathcal{R}^{(l')} \leftarrow \mathcal{R}^{(l')} \cup \mathcal{R}^{(l'-1)}$ // enforce rings form an increasing chain
  $\sigma^{(l')} \leftarrow \mathbf{ERS}.\mathsf{Extend}(\mu, \{\mathtt{pk}_j\}_{j \in \mathcal{R}^{(l'-1)}}, \sigma^{(l'-1)}, \{\mathtt{pk}_j\}_{j \in \mathcal{R}^{(l')}})$
$\sigma \leftarrow \sigma^{(l)}$
return σ

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

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:*

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

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 requirement from regular ring signatures: no adversary should be able to produce a signature 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 $\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathbf{ERS}}(\lambda)$). Additionally, to account for the key duplication attack (where an adversary registers 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, Figure 2).

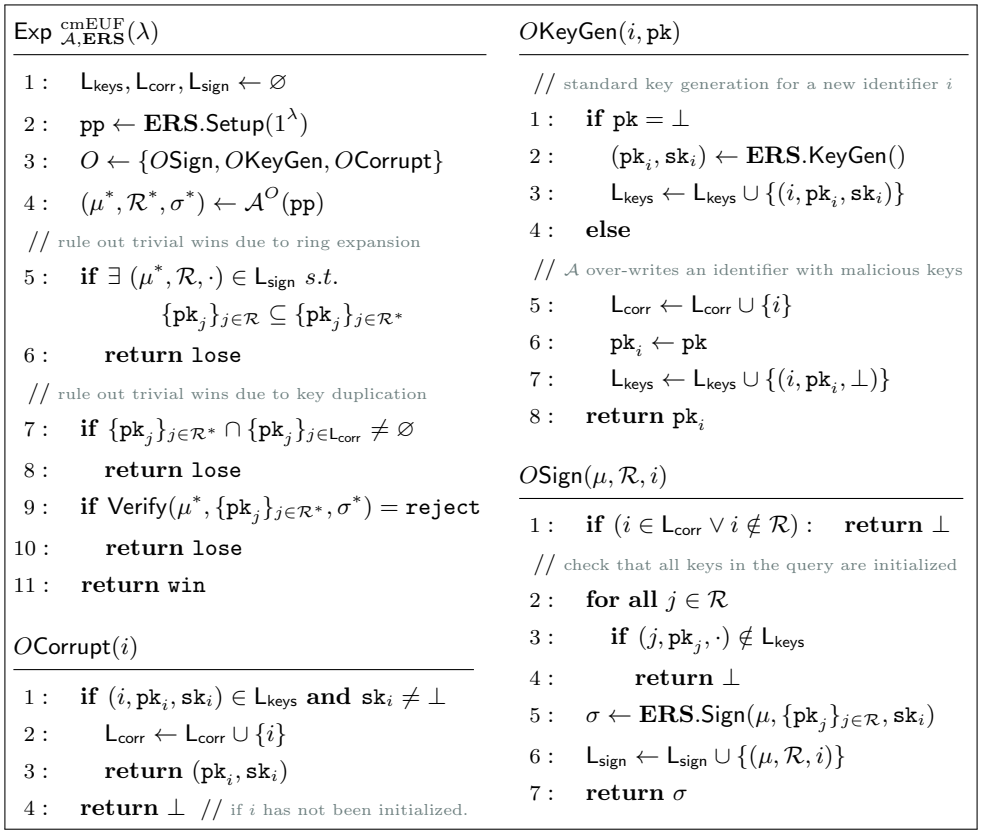

Fig. 2: Existential Unforgeability under Chosen Message Attack for (Extendable) Ring Signatures (security experiment and oracles). Our key generation oracle allows $\mathcal{A}$ to register signers with arbitrary public keys (i.e., it also acts as a registration oracle).

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 Figure 2), the success probability is negligible, i.e.:* $\Pr\left[\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathbf{ERS}}(\lambda) = \mathtt{win}\right] \leq \mathtt{negl}$.

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

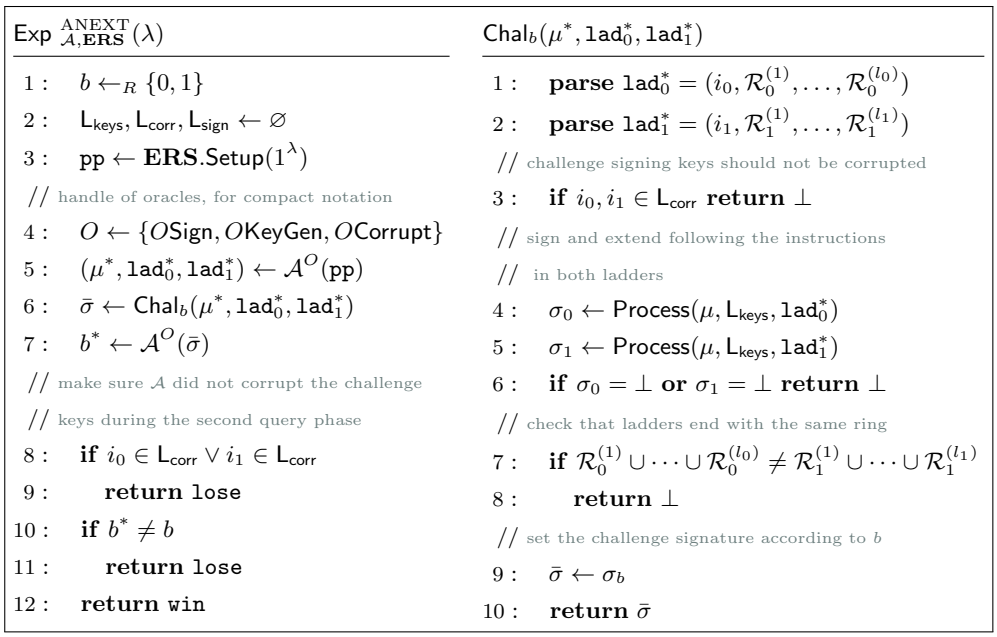

Fig. 3: Anonymity and Anonymous Extendability for Extendable Ring Signatures. The oracles $O\mathsf{Sign}$, $O\mathsf{KeyGen}$ and $O\mathsf{Corrupt}$ are defined in Figure 2.

For standard anonymity we consider adversaries that output ladders ($\mathtt{lad}_0^*, \mathtt{lad}_1^*$ in line 5 of $\mathsf{Exp}^{\text{ANEXT}}_{\mathcal{A},\mathbf{ERS}}$ in Figure 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.,:* $\Pr\left[\mathsf{Exp}^{\text{ANEXT}}_{\mathcal{A},\mathbf{ERS}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}$.

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 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 (Figure 3), it holds that:* $\Pr\left[\mathsf{Exp}^{\text{ANEXT}}_{\mathcal{A},\mathbf{ERS}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}$.

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 Figure 4.

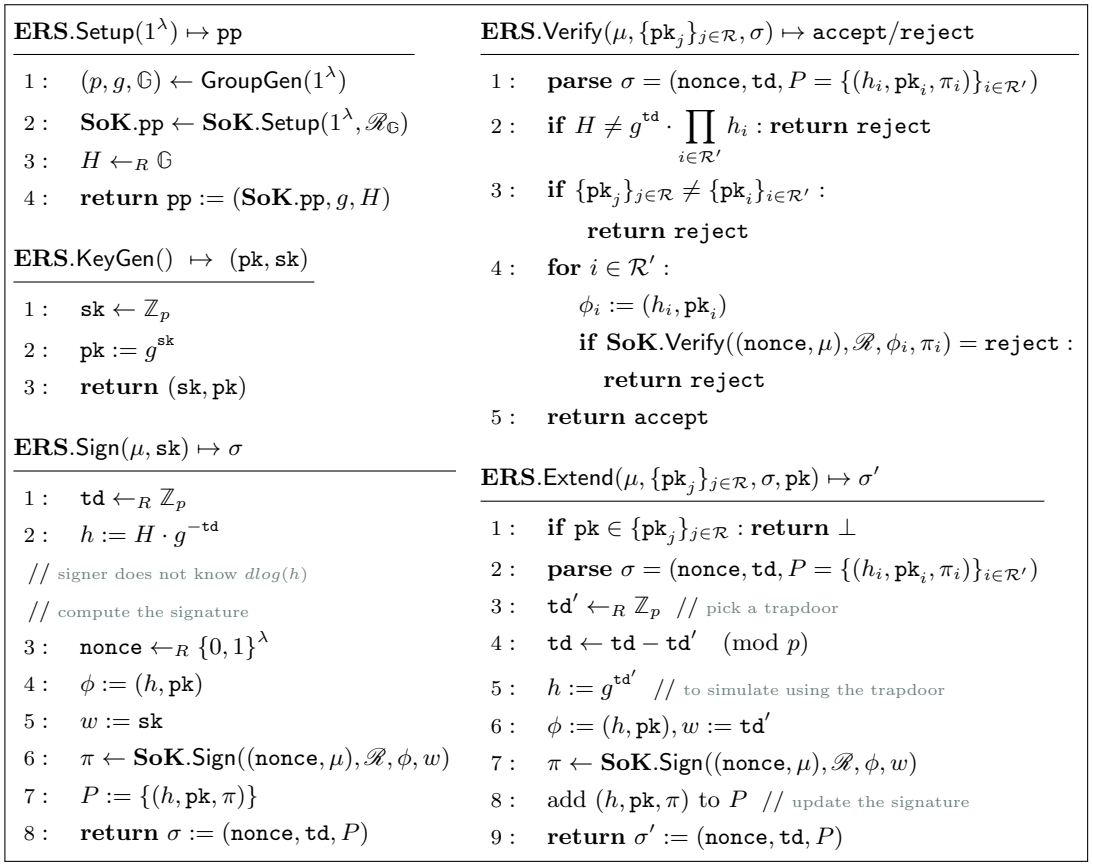

Fig. 4: Extendable Ring Signatures from Signature of Knowledge and Discrete Log. The relation used by the SoK scheme is $\mathscr{R}_\mathbb{G} = \{(\phi, w) = (h, \mathtt{pk}, x) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p : g^x = h \vee g^x = \mathtt{pk}\}$.

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 Figure 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. □

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 (SMLERS). We give a security model for this new primitive, and describe an instantiation that builds on our ERS construction from Section 3.3.

4.1 Syntax

A same-message linkable extendable ring signature scheme is a tuple of six algorithms $\mathbf{SMLERS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extend}, \mathsf{Link})$. 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.

$\mathsf{Link}(\mu, (\sigma_0, \{\mathtt{pk}_j\}_{j\in\mathcal{R}_0}), (\sigma_1, \{\mathtt{pk}_j\}_{j\in\mathcal{R}_1})) \to \{\mathtt{linked}, \mathtt{unlinked}\}$**:** 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 `linked` if σ_0 and σ_1 were produced by the same signer, and `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 $\mathtt{lad}_0 = (i_0, \mathcal{R}_0^{(1)}, \ldots, \mathcal{R}_0^{(l_0)})$, $\mathtt{lad}_1 = (i_1, \mathcal{R}_1^{(1)}, \ldots, \mathcal{R}_1^{(l_0)})$ 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:*

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

where $\mathsf{Process}$ *is the algorithm described in Figure 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).

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

$\mathtt{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}} \leftarrow \varnothing$
$O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$
$(\mu^*, \{(\sigma_k^*, \mathcal{R}_k^*)\}_{k\in[1\ldots,t]}) \leftarrow \mathcal{A}^O(\mathtt{pp})$
// $\mathcal{A}$ has never seen a signature for the message and a subring of the forgery rings
if $\exists\ (\mu^*, \mathcal{R}, \cdot) \in \mathsf{L}_{\mathsf{sign}}$ *s.t.* $\mathcal{R} \subseteq \mathcal{R}_k^*$ for some $k \in [1, \ldots, t]$ **return** `lose`
// $\mathcal{A}$ holds at most $t-1$ secret keys, among the keys identified by the forgery rings
if $|(\mathcal{R}_1^* \cup \cdots \cup \mathcal{R}_t^*) \cap \mathsf{L}_{\mathsf{corr}}| \geq t$ **return** `lose`
// all the signatures in the forgery verify (for the same message)
if $\exists\ k \in [1\ldots,t]$ s.t. $\mathsf{Verify}(\mu^*, \{\mathtt{pk}_j\}_{j\in\mathcal{R}_k^*}, \sigma_k^*) = \mathtt{reject}$ **return** `lose`
// all signatures in the forgery are unlinked (here $k, l \in [1, \ldots, t]$)
if $\exists\ k \neq l$ s.t. $\mathsf{Link}(\mu^*, (\sigma_k^*, \{\mathtt{pk}_j\}_{j\in\mathcal{R}_k^*}), (\sigma_l^*, \{\mathtt{pk}_j\}_{j\in\mathcal{R}_l^*})) = \mathtt{linked}$
  return `lose`
return `win`

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

Definition 7 (Secure SMLERS). *A same-message linkable extendable ring signature scheme (SMLERS) 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 SMLERS is said to be one-more linkable if for all PPT adversaries $\mathcal{A}$ taking part in the same-message one-more linkability experiment* ($\mathsf{Exp}^{\text{omlink}}_{\mathcal{A},\mathbf{SMLERS}}(\lambda)$ *depicted in Figure 5), it holds that:* $\Pr[\mathsf{Exp}^{\text{omlink}}_{\mathcal{A},\mathbf{SMLERS}}(\lambda) = \mathtt{win}] \leq \mathtt{negl}$.

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

$\mathsf{Exp}^{\text{cmunlink}}_{\mathcal{A},\mathbf{LRS}}(\lambda)$

1 : $b \leftarrow_R \{0,1\}, \mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}} \leftarrow \varnothing$
2 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
3 : $O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$
4 : $(\{\mu_0, \mathcal{R}_0, i_0\}, \{\mu_1, \mathcal{R}_1, i_1\}) \leftarrow \mathcal{A}^O(\mathsf{pp})$
5 : $(\bar{\sigma}_0, \bar{\sigma}_1) \leftarrow \mathsf{Chal}_b(\{\mu_0, \mathcal{R}_0, i_0\}, \{\mu_1, \mathcal{R}_1, i_1\})$
6 : $b^* \leftarrow \mathcal{A}^O(\bar{\sigma}_0, \bar{\sigma}_1)$
// Rule out corruption of challenge identities
7 : **if** $i_0 \in \mathsf{L}_{\mathsf{corr}} \vee i_1 \in \mathsf{L}_{\mathsf{corr}}$ **return** `lose`
// Rule out trivial attacks using Link
8 : **if** $\mu_0 = \mu_1$ **return** `lose`
// Rule out trivial attacks using Link
9 : **if** $(\mu_0, \cdot, i_0) \in \mathsf{L}_{\mathsf{sign}} \vee (\mu_0, \cdot, i_1) \in \mathsf{L}_{\mathsf{sign}}$
$\vee\ (\mu_1, \cdot, i_0) \in \mathsf{L}_{\mathsf{sign}} \vee (\mu_1, \cdot, i_1) \in \mathsf{L}_{\mathsf{sign}}$
10 : **return** `lose`
11 : **if** $b^* \neq b$ **return** `lose`
12 : **return** `win`

$\mathsf{Chal}_b(\{\mu_0, \mathcal{R}_0, i_0\}, \{\mu_1, \mathcal{R}_1, i_1\})$

// the challenge identities must be uncorrupted
1 : **if** $i_0 \in \mathsf{L}_{\mathsf{corr}} \vee i_1 \in \mathsf{L}_{\mathsf{corr}}$
2 : **return** $\perp$
// one identity needs to be in both rings
3 : **if** $i_0 \notin \mathcal{R}_0 \cap \mathcal{R}_1 \vee i_1 \notin \mathcal{R}_1$
4 : **return** $\perp$
// signing keys must exist
5 : **if** $\nexists\ (i_0, \mathtt{pk}_{i_0}, \mathtt{sk}_{i_0}) \in \mathsf{L}_{\mathsf{keys}}$ **return** $\perp$
6 : **if** $\nexists\ (i_1, \mathtt{pk}_{i_1}, \mathtt{sk}_{i_1}) \in \mathsf{L}_{\mathsf{keys}}$ **return** $\perp$
// generate a signature
7 : $\bar{\sigma}_0 \leftarrow \mathbf{LRS}.\mathsf{Sign}(\mu_0, \{\mathtt{pk}_i\}_{i \in \mathcal{R}_0}, \mathtt{sk}_{i_0})$
// generate the second signature according to
// the experiment's bit b
8 : $\bar{\sigma}_1 \leftarrow \mathbf{LRS}.\mathsf{Sign}(\mu_1, \{\mathtt{pk}_i\}_{i \in \mathcal{R}_1}, \mathtt{sk}_{i_b})$
9 : **return** $(\bar{\sigma}_0, \bar{\sigma}_1)$

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

4.3 SMLERS from Signatures of Knowledge and Discrete Log

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

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

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

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 SMLERS are with respect to the new relation $\mathscr{R}_{\mathbf{SMLERS}}$.

Second, we modify the Sign algorithm of our ERS in Figure 4 so that it additionally computes $g' := \mathsf{H}(\mu)$ and $\tau := (g')^{\mathtt{sk}}$ 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 $\mathsf{Combine}$ algorithm (below).

Extendability: Given a signature σ on a message μ for the ring $\mathcal{R}$ with threshold 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 $\mathbf{ETRS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Combine}, \mathsf{Extend})$, where the public parameters $\mathtt{pp}$ produced by Setup are implicitly available to all other algorithms:

$\mathsf{Setup}(1^\lambda) \to \mathtt{pp}$**:** Takes a security parameter λ and outputs a set of public parameters $\mathtt{pp}$.

$\mathsf{KeyGen}() \to (\mathtt{pk}, \mathtt{sk})$**:** Generates a new public and secret key pair.

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

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

$\mathsf{Combine}(\mu, \sigma_0, \sigma_1, \{\mathtt{pk}_i\}_{i\in\mathcal{R}}) \mapsto \sigma'$**:** 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.

$\mathsf{Extend}(\mu, \sigma, \{\mathtt{pk}_i\}_{i\in\mathcal{R}}, \{\mathtt{pk}_i\}_{i\in\mathcal{R}'}) \mapsto \sigma'$**:** 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 '$\mathsf{Sign}\&\mathsf{Combine}$' executions with a Join operation (described in Section 2.1). For the sake of formalism, we present our security model only for schemes with $\mathsf{Combine}$ and defer the discussion on how to handle Join operations to the Section 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 $\mathtt{lad}$ in a slightly different way than we did in the context of extendable ring signatures (Section 3). We generalize $\mathtt{lad}$ to support arbitrary sequences of actions that could lead to a valid threshold ring signature (on some fixed message). $\mathtt{lad}$ will contain a sequence of tuples of the form $(\mathtt{action}, \mathtt{input})$. The first component, $\mathtt{action}$, can take on the values $\mathsf{Sign}, \mathsf{Combine}$, or Extend. If $\mathtt{action} = \mathsf{Sign}$, we expect $\mathtt{input} = (\mathcal{R}, i)$, where $\mathcal{R}$ and i are the ring and signer identity with which the signature should be produced. If $\mathtt{action} = \mathsf{Combine}$, we expect $\mathtt{input} = (l_1, l_2, \mathcal{R})$, where l_1 and l_2 are indices of two signatures under the same ring $\mathcal{R}$. If $\mathtt{action} = \mathsf{Extend}$, we expect $\mathtt{input} = (l', \mathcal{R})$, 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 $\mathsf{Process}(\mu, \mathsf{L}_{\mathsf{keys}}, \mathtt{lad})$, which processes all of the operations in $\mathtt{lad}$ on the message μ (using keys stored in the list $\mathsf{L}_{\mathsf{keys}}$) and returns $(\sigma, t, \mathcal{R})$: the signature returned by the last operation of $\mathtt{lad}$, the corresponding threshold, and the ring that σ verifies under. We define $\mathtt{lad.sr}$ to be the union of all identities and rings in $\mathtt{lad}$. ($\mathtt{sr}$ stands for super-ring.) We give a formal description of $\mathsf{Process}$ in the full version of this paper.

Definition 10 (Correctness for ETRS). *For correctness, we require that for all ladders* $\mathtt{lad}$*, the signature returned by* $\mathsf{Process}(\mathtt{lad})$ *verifies. Formally: for all security parameters* $\lambda \in \mathbb{N}$*, for any message* $\mu \in \{0,1\}^*$*,*

for any ladder $\texttt{lad}$ *of polynomial size identifying a ring* $\mathcal{R} := \texttt{lad.sr}$ *of public-key identifiers, for any chosen threshold value* $1 \leq t \leq |\mathcal{R}|$*, it holds:*

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

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.*

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

1 : $\mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}} \leftarrow \varnothing$
2 : $\texttt{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
3 : $O \leftarrow \{O\mathsf{Sign}, O\mathsf{KeyGen}, O\mathsf{Corrupt}\}$
4 : $(t^*, \mu^*, \mathcal{R}^*, \sigma^*) \leftarrow \mathcal{A}^O(\texttt{pp})$
5 : $q \leftarrow |\{(\mu^*, \mathcal{R}, \cdot) \in \mathsf{L}_{\mathsf{sign}}\ s.t.\ \mathcal{R} \subseteq \mathcal{R}^*)\}|$
// rule out attacks if $\mathcal{A}$ knows too many sk:s or honestly generated signatures for μ^*
6 : **if** $|\mathcal{R}^* \cap \mathsf{L}_{\mathsf{corr}}| + q \geq t$ **return** $\texttt{lose}$
// rule out outputs that do not verify
7 : **if** $\mathsf{Verify}(t, \mu^*, \{\texttt{pk}_j\}_{j\in\mathcal{R}^*}, \sigma^*) = \texttt{reject}$ **return** $\texttt{lose}$
8 : **return** $\texttt{win}$

Fig. 7: Existential Unforgeability under Chosen Message Attack for (Extendable) Threshold Ring Signatures. The key generation, corruption and signing oracles are as in Figure 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 Figure 7 is* $\Pr\left[\mathsf{Exp}^{\text{cmEUF}}_{\mathcal{A},\mathbf{ETRS}}(\lambda) = \texttt{win}\right] \leq \texttt{negl}$.

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.

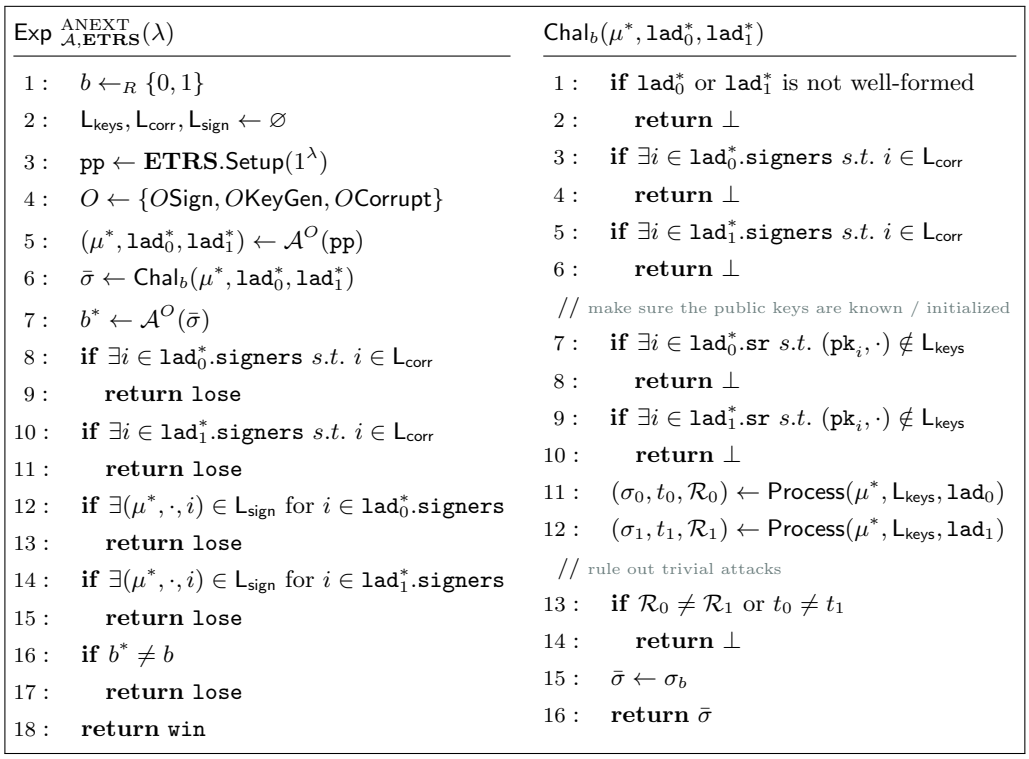

Fig. 8: Anonymity and Anonymous Extendability for Extendable Threshold Ring Signatures. The key generation, corruption and signing oracles are exactly as described in the unforgeability experiment (Figure 7).

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.:* $\Pr\left[\mathsf{Exp}^{\text{ANEXT}}_{\mathcal{A},\mathbf{ETRS}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}$.

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 Section 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 Figure 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.:* $\Pr\left[\mathsf{Exp}^{\text{ANEXT}}_{\mathcal{A}_{\text{sAnon}},\mathbf{ERS}}(\lambda) = \mathtt{win}\right] \leq \frac{1}{2} + \mathtt{negl}$.
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 Figure 9.

Theorem 2. *Assuming that* **SMLERS** *is a secure same-message linkable extendable ring signature scheme, then the scheme* $\mathbf{ETRS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Extend}, \mathsf{Combine})$ *described in Figure 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 signatures. Concretely, the size of extended threshold

$\mathsf{Setup}(1^\lambda) \mapsto \mathtt{pp}$

return SMLERS.$\mathsf{Setup}(1^\lambda)$

$\mathsf{KeyGen}() \mapsto (\mathtt{pk}, \mathtt{sk})$

return SMLERS.$\mathsf{KeyGen}()$

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

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

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

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

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

1 : $\sigma'_0 \leftarrow \{s_0 \in \sigma_0 \mid \forall s_1 \in \sigma_1 : \mathsf{Link}(\mu, (s_0, \{\mathtt{pk}_i\}_{i\in\mathcal{R}}), (s_1, \{\mathtt{pk}_i\}_{i\in\mathcal{R}})) = \mathtt{unlinked}\}$
2 : **return** $\sigma'_0 \cup \sigma_1$

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

1 : Parse $\sigma = \{s_0, ..., s_\ell\}$ as a set of signatures // removing duplicates
2 : **if** $|\sigma| < t$ **return** `reject`
3 : **if** $\exists\ s_i \in \sigma : \mathsf{Verify}(\mu, \{\mathtt{pk}_i\}_{i\in\mathcal{R}}, s_i) = \mathtt{reject}$ **return** `reject`
4 : **if** $\exists\ (s_i, s_j) \in \sigma \times \sigma : s_i \neq s_j \wedge$
$\mathsf{Link}(\mu, (s_i, \{\mathtt{pk}_i\}_{i\in\mathcal{R}}), (s_j, \{\mathtt{pk}_i\}_{i\in\mathcal{R}})) = \mathtt{linked}$ **return** `reject`
5 : **return** `accept`

Fig. 9: Generic Compiler for Extendable Threshold Ring Signatures from Extendable Same-Message Linkable Ring Signatures.

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 Figure 9, which if instantiated using our SMLERS from Signatures of Knowledge and Discrete Log of Section 4.3, returns signatures of size linear in $t \cdot |\mathcal{R}|$.

Our Construction in a Nutshell Similarly to the ERS construction of Figure 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 $\mathtt{pk}$.

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 $(x_i, \mathtt{td}_i) \in \mathbb{Z}_p \times \mathbb{G}$. These pairs of values define a unique polynomial $f(x)$ of degree n' such that $f(0) = \mathrm{dlog}_g(H)$ and $f(x_i) = \mathtt{td}_i$ for every

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

// compute values for the Lagrange interpolation

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

// compute evaluation points for the Lagrange interpolation

$\mathcal{X} \coloneqq \{x\}_{(x,y)\in\mathcal{Z}}$

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

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

$\phi \coloneqq (\hat{y}, \mathtt{pk}_s)$ // include new poly. value in the statement

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

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

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

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

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

return $(\hat{y}, \pi)$

Fig. 10: Subroutine used in our ETRS construction depicted in Figure 11.

$i \in [n']$. Of course, since $\mathrm{dlog}_g(H)$ 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 $(x_i, g^{\mathtt{td}_i})$), 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 $(x_i, \mathtt{td}_i)$, and generate a proof for $\mathscr{R}_{\mathbb{G}}$ by satisfying the first clause (proving knowledge of $\mathtt{td}_i$), to include any $\mathtt{pk}$ in the ring. The pair $(x_i, \mathtt{td}_i)$ is then removed from the list of trapdoors. (In case the owner of $\mathtt{pk}$ later wants to join the signature, the Extend algorithm encrypts $\mathtt{td}_i$ to $\mathtt{pk}$; later, the owner of $\mathtt{pk}$ can recover $\mathtt{td}_i$ 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 Figure 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 $L_{(\mathcal{X},j)}(X) \coloneqq \prod_{m \in \mathcal{X} \setminus \{j\}} \frac{X-m}{j-m} \in \mathbb{F}[X]$.

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* **ETRS** $=$ (Setup, KeyGen, Sign, Verify, Extend, Join) *described in Figure 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 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 Sections 3, 4.3 and 5, at the 128-bit security level within the RELIC [5] 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 @ 4GHz, with HyperThreading and TurboBoost 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 Figures 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.

[5] `https://github.com/relic-toolkit/relic`

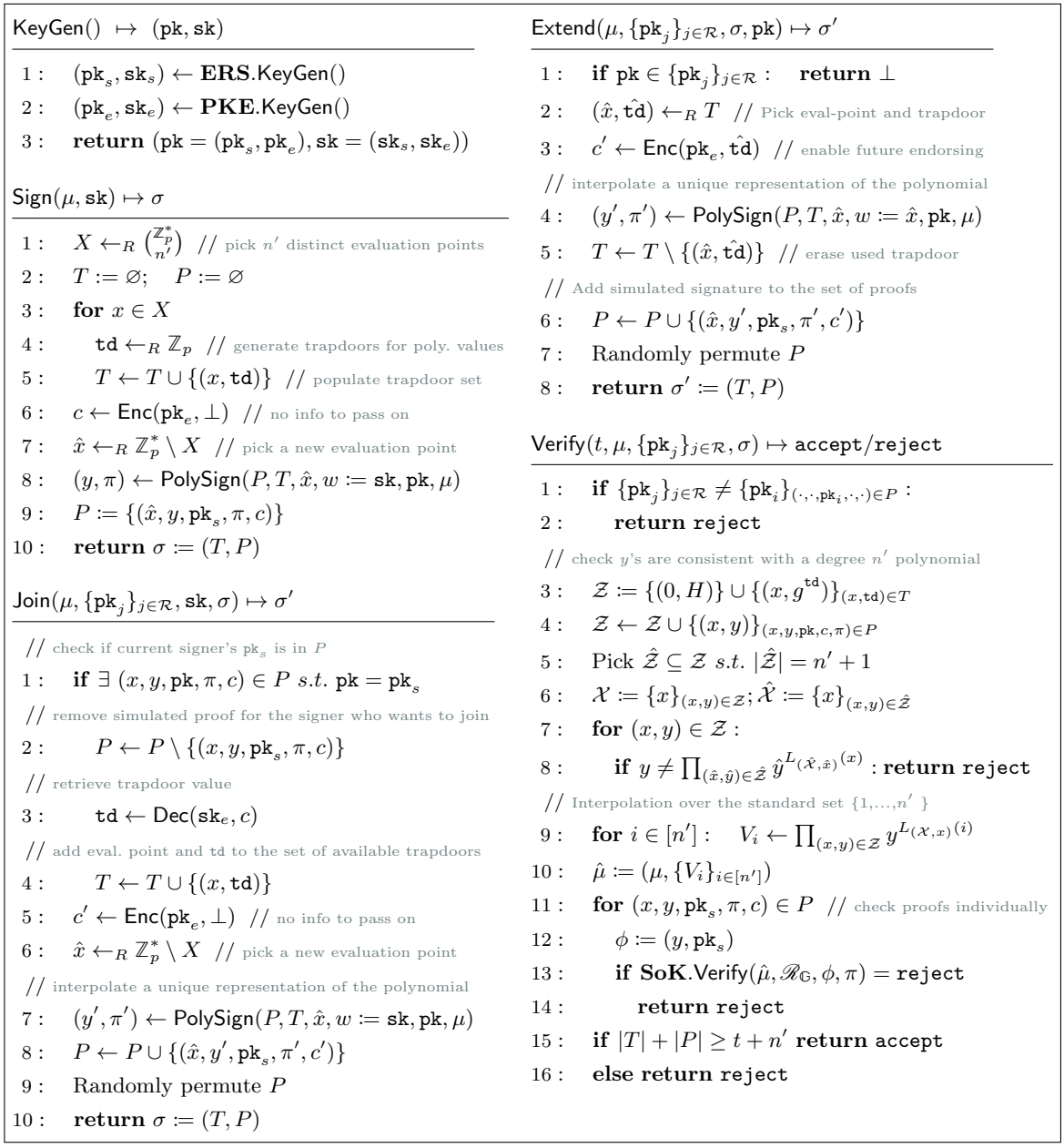

Fig. 11: Extendable Threshold Ring Signatures from Signature of Knowledge and Hardness of Discrete Log. The Setup algorithm is the same as in the ERS construction of Figure 4 (with $\mathscr{R}_{\mathbb{G}} = \{(\phi, w) = (h, \mathtt{pk}, x) \in \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_p : g^x = h \vee g^x = \mathtt{pk}\}$). 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 $(\mathtt{pk}_s, \mathtt{pk}_e)$ (or of $\mathtt{pk}_i$ into $(\mathtt{pk}_{s,i}, \mathtt{pk}_{e,i})$), of sk into $(\mathtt{sk}_s, \mathtt{sk}_e)$ and of σ into (T, P) is done implicitly.

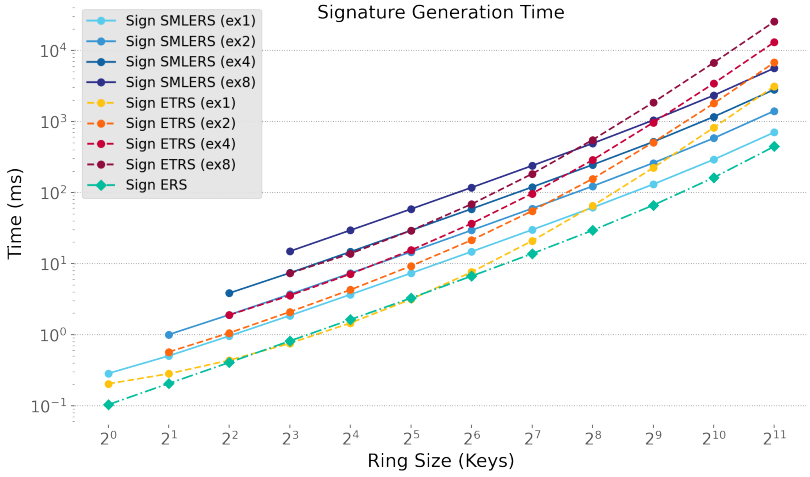

Fig. 12: Clock time for Sign in the three implemented schemes for different thresholds. The signature generation time includes the initial signature generation and subsequent joinings/extensions.

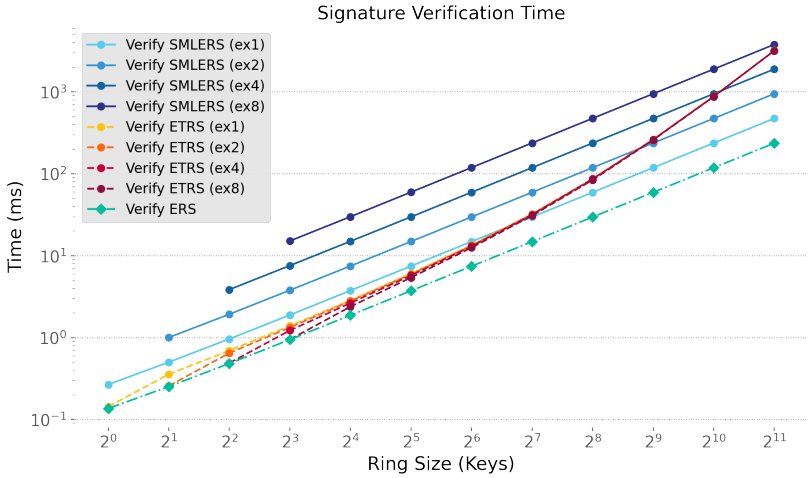

Fig. 13: Clock time for Verify in the three implemented schemes for different thresholds. The verification time is that of verifying the final extended signature.

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 $\{(g^x = h \vee g^x = \mathtt{pk}) \wedge (g^x = h \vee (g')^x = \tau)\}$, which allows us to share code with 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.

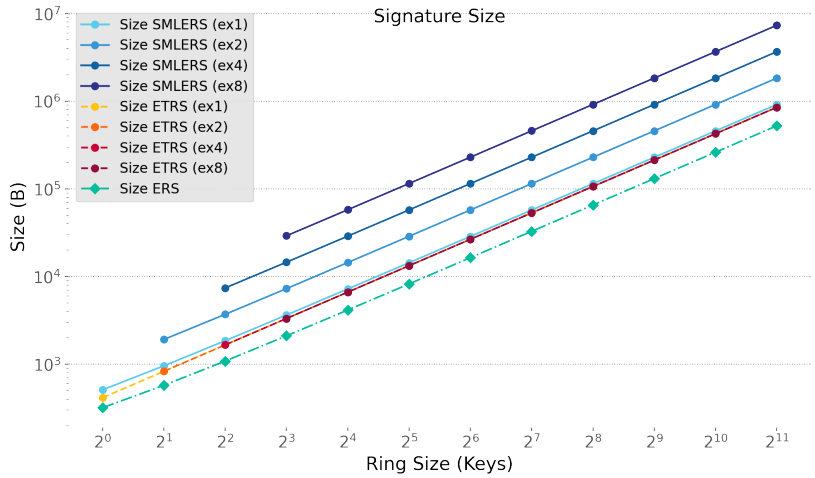

Fig. 14: Signature sizes for all the three implemented schemes, with varying thresholds. In the ETRS scheme, the signature size is independent of the threshold, while in SMLERS there is a linear dependence.

Acknowledgments This work was partially funded by ELLIIT; the Swedish Foundation for Strategic Research grant RIT17-0035; the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

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 (Mar 2006)
2. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of Cryptographic Engineering **2**(2), 77–89 (Sep 2012)
3. Bettaieb, S., Schrek, J.: Improved lattice-based threshold ring signature scheme. In: Gaborit, P. (ed.) Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013. pp. 34–51. Springer, Heidelberg (Jun 2013)
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 (May 2003)
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.R. (eds.) ESORICS 2015, Part I. LNCS, vol. 9326, pp. 243–265. Springer, Heidelberg (Sep 2015)
7. Bose, P., Das, D., Rangan, C.P.: Constant size ring signature without random oracle. In: Foo, E., Stebila, D. (eds.) ACISP 15. LNCS, vol. 9144, pp. 230–247. Springer, Heidelberg (Jun / Jul 2015)
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 (Aug 2002)
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 (Oct 2004)
10. Chandran, N., Groth, J., Sahai, A.: Ring signatures of sub-linear size without random oracles. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 423–434. Springer, Heidelberg (Jul 2007)
11. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (Aug 2006)
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 06. pp. 297–302. ACM Press (Mar 2006)
13. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (May 2004)
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 19. LNCS, vol. 11464, pp. 67–88. Springer, Heidelberg (Jun 2019)
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, Part II. LNCS, vol. 12111, pp. 423–452. Springer, Heidelberg (May 2020)
16. Liu, J.K.: Ring signature. In: Advances in Cyber Security: Principles, Techniques, and Applications, pp. 93–114. Springer (2019)

17. Liu, J.K., Wong, D.S.: On the security models of (threshold) ring signature schemes. In: Park, C., Chee, S. (eds.) ICISC 04. LNCS, vol. 3506, pp. 204–217. Springer, Heidelberg (Dec 2005)
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, Part I. LNCS, vol. 11735, pp. 726–746. Springer, Heidelberg (Sep 2019)
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 19. LNCS, vol. 11464, pp. 110–130. Springer, Heidelberg (Jun 2019)
20. Malavolta, G., Schröder, D.: Efficient ring signatures in the standard model. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part II. LNCS, vol. 10625, pp. 128–157. Springer, Heidelberg (Dec 2017)
21. Melchor, C.A., Cayrel, P.L., Gaborit, P., Laguillaumie, F.: A new efficient threshold ring signature scheme based on coding theory. IEEE Transactions on Information 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, S., Schürmann, C.: Eos a universal verifiable and coercion resistant voting protocol. In: E-VOTE-ID. Lecture Notes in Computer Science, vol. 10615, pp. 210–227. Springer (2017)
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 (Dec 2001)
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 (Apr 2007)
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, Part II. LNCS, vol. 10493, pp. 456–474. Springer, Heidelberg (Sep 2017)
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 (Dec 2004)
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 11. pp. 261–267. ACM Press (Mar 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 (Dec 2002)