Extendable Threshold Ring Signatures with Enhanced Anonymity

Gennaro Avitabile[1⋆], Vincenzo Botta[2⋆], and Dario Fiore[1]

[1] IMDEA Software Institute, Madrid, Spain.
{gennaro.avitabile,dario.fiore}@imdea.org
[2] University of Warsaw, Warsaw, Poland. v.botta@uw.edu.pl

Abstract. Threshold ring signatures are digital signatures that allow t parties to sign a message while hiding their identity in a larger set of n users called "ring". Recently, Aranha et al. [PKC 2022] introduced the notion of *extendable* threshold ring signatures (ETRS). ETRS allow one to update, in a non-interactive manner, a threshold ring signature on a certain message so that the updated signature has a greater threshold, and/or an augmented set of potential signers. An application of this primitive is anonymous count me in. A first signer creates a ring signature with a sufficiently large ring announcing a proposition in the signed message. After such cause becomes *public*, other parties can anonymously decide to support that proposal by producing an updated signature. Crucially, such applications rely on partial signatures being posted on a *publicly accessible* bulletin board since users may not know/trust each other.

In this paper, we first point out that even if anonymous count me in was suggested as an application of ETRS, the anonymity notion proposed in the previous work is insufficient in many application scenarios. Indeed, the existing notion guarantees anonymity only against adversaries who just see the last signature, and are not allowed to access the "full evolution" of an ETRS. This is in stark contrast with applications where partial signatures are posted in a public bulletin board. We therefore propose stronger anonymity definitions and construct a new ETRS that satisfies such definitions. Interestingly, while satisfying stronger anonymity properties, our ETRS asymptotically improves on the two ETRS presented in prior work [PKC 2022] in terms of both time complexity and signature size. Our ETRS relies on extendable non-interactive witness-indistinguishable proof of knowledge (ENIWI PoK), a novel technical tool that we formalize and construct, and that may be of independent interest. We build our constructions from pairing groups under the SXDH assumption.

Keywords: Threshold Ring Signatures · Anonymity · Malleable Proof Systems.

⋆ Work done mainly while working at University of Salerno, Italy.

1 Introduction

Anonymity is a central requirement in several privacy-preserving technologies. Notable examples are e-voting protocols [34], anonymous authentication [30], and privacy-protecting cryptocurrencies [35]. A central cryptographic primitive that can be used to provide anonymity in applications is ring signatures [33]. Ring signatures [33] are digital signatures which allow one user to sign a message while hiding her identity in a larger group called ring $\mathcal{R}$. In practice, the signing algorithm, aside the message, takes as input a set of public keys (i.e., the ring) and one of the corresponding secret keys. The produced signature guarantees that one of the public keys in the ring signed the message, while hiding which one of the secret keys was used to create the signature. Clearly, the larger is $\mathcal{R}$ the greater is the anonymity provided to the signer. Constructions for ring signatures are known from a variety of cryptographic tools such as RSA [17], pairing groups [10,16,37], non-interactive zero-knowledge proofs [11,3,22], and lattices [27,28,19,9]. A practical application of ring signature is whistleblowing. By signing a message, a member of a company can report a wrong practice of the company itself while hiding his identity among all the other employees.

Threshold ring signatures [12] enrich ring signatures by allowing t signers to hide their identity within the ring. The signature guarantees that t members of $\mathcal{R}$ signed the message without revealing which ones. Ring signatures can be seen as threshold ring signatures with $t = 1$. Some threshold ring signatures also enjoy a property called flexibility [31,29]. They allow new signers to join already produced signatures: a signature on a message m that was already created with threshold t for a ring $\mathcal{R}$ can be transformed into a new signature on message m with threshold $t + 1$ w.r.t. the same ring $\mathcal{R}$. The interesting aspect of flexible threshold ring signatures is that the update does not require the participation of any previous signer. Nevertheless, until recently, all known threshold ring signatures did not offer an analogous property that would allow extending the ring. In other words, all previous constructions required to fix the ring from the beginning and did not allow to modify it further.

This problem has been addressed for the first time in the recent work of Aranha et al. [2] which has put forth the notion of *extendable* threshold ring signatures (ETRS). ETRS, aside the join operation, also provide an *extend* operation: any signature with ring $\mathcal{R}$ can be transformed by anybody into a signature with ring $\mathcal{R}'$ s.t. $\mathcal{R} \subset \mathcal{R}'$. After the extend operation, all signers in $\mathcal{R}'$ can join the signature.

On count-me-in applications. Aranha et al. [2] observe how the richer flexibility of ETRS can enable more advanced forms of whistleblowing or anonymous petitions. The first signer could create a ring signature with a sufficiently large ring announcing a proposition in the signed message. After such cause becomes *public*, other parties could support the cause via extend and/or join operations. As also reported in [2], an observer who has seen signatures on an old ring $\mathcal{R}$ and on a new ring $\mathcal{R}'$ can always compute $\mathcal{R}' \setminus \mathcal{R}$, and this can help narrowing down the identity of the signers. This problem is inherent in the functionality provided by ETRS, and it worsens as t approaches the size of the ring. A clear example is the one of a signature w.r.t. ring $\mathcal{R}$ with threshold $t = n - 1$, where $n = |\mathcal{R}|$, which is transformed into a signature with threshold $t = n' - 1$ w.r.t. $\mathcal{R}', |\mathcal{R}'| = n' = n + 1$ (i.e., the threshold is increased by one and the final ring contains an additional public key of a user A). By looking at the two signatures, one can infer that one signer of the last signature either comes from $|\mathcal{R}|$ or it is A with probability $\frac{1}{2}$.

In [2], the authors address this issue by proposing an anonymity definition in which the adversary is restricted to see only the signature obtained eventually, after all the extend and join operations have been applied. However, this restriction hinders the use of ETRS in real-world count-me-in applications since it bears an implicit requirement: the signers should privately interact to incrementally produce the ETRS and then only the final signature can be made public to the outside world. This means that *all* the possible advocates of a proposal should be given access to a private bulletin board where partial signatures are posted. Additionally, the abstract of [2] informally mentions the importance of *fellow signer anonymity* (FSA), a property stating that "it is often crucial for signers to remain anonymous even from their fellow signers". Such requirement was previously formally modeled in [29], but it is not captured by the anonymity definitions of [2]. Indeed, it is unclear how such property could be guaranteed when anonymity is only formulated w.r.t. an adversary who cannot see intermediate signatures (as real signers would) and does not have the secret key of any of the signers (as in the definition of [2]).

1.1 Our Contributions

In this work, we address the aforementioned shortcomings of ETRS. First, we propose a stronger security definition that guarantees anonymity even against adversaries that see the full "evolution" of a signature. Second, we propose a new ETRS construction that achieves our strong anonymity definition, and also improves in efficiency over previous work (cfr., Table 1). Our construction relies on extendable non-interactive witness indistinguishable proof of knowledge (ENIWI PoK), a novel technical tool that we formalize and construct, and that may be of independent interest. In what follows, we present our contributions in more detail.

Stronger anonymity for ETRS*.* Even though certain leaks are inherent when the adversary gets to see several ETRS, one should aim at building a scheme which leaks nothing more than that. To this regard, we start from the anonymity definition proposed in [2] and we make it stronger as follows. We allow the adversary $\mathcal{A}$ to see all the ETRS that led to the final signature. In a nutshell, $\mathcal{A}$ outputs two sequences of operations which at every step lead to an ETRS on the same message, with the same ring, and the same threshold in both sequences. The challenger $\mathcal{C}$ picks one of such sequences at random, executes it, and gives to $\mathcal{A}$ the corresponding outputs of each step. We then require that $\mathcal{A}$ only has a negligible advantage in guessing which sequence was applied. We also propose a security game that models fellow signer anonymity for ETRS.

Constructing ETRS. In [2], two constructions of ETRS are proposed: the first one is obtained from extendable same-message linkable ring signatures (SMLERS)[3], while the second one is constructed from signatures of knowledge (SoK) for the discrete log relation, public key encryption (PKE), and the discrete log assumption. The first scheme achieves our stronger anonymity notion but suffers quite high complexity; for instance, the signature size is $\mathcal{O}(tn)$. The second scheme in [2] is more compact but does not fulfill our stronger anonymity notion. Indeed, anyone who sees an ETRS before and after a join operation can easily pinpoint the exact signer who joined the signature (see App. A.1 of [5] for more details). It follows that such scheme is also not fellow-signer anonymous, since no secret key is required to carry out the above attack.

We construct an ETRS which fulfills our stronger anonymity definition and is also fellow-signer anonymous. As shown in Tab. 1, our ETRS also generally improves the constructions given in [2] in terms of both time complexity and signature size. In App. A.1 of [5], we give a high-level overview of both ETRS presented in [2]. To build our ETRS, we introduce the notion of ENIWI PoK, which may be of independent interest. We then show how to build an ETRS from an ENIWI PoK for a hard relation, and an IND-CPA homomorphic public key encryption scheme.

Scheme	**Size**	**Sign**	**Join**	**Extend**	**Verify**	**Anonymity**	**FSA**
SMLERS [2]	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$	$\mathcal{O}(n)$	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$	Strong	Yes
DL + SoK + PKE [2]	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	Weak	No
Ours	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Strong	Yes

Table 1. Comparison of signature size, time complexities, and anonymity guarantees of our ETRS and the ones presented in [2]. Let $|\mathcal{R}| = n$ and t be the threshold. In the DL + SoK + PKE construction of [2] signature size and time complexities both depend on a fixed upper bound on the ring size N. We say that a scheme achieves weak anonymity if it achieves the anonymity property of [2], and strong anonymity if our stronger anonymity definition is satisfied. FSA stands for fellow-signer anonymity.

ENIWI *PoKs.* In [14], Chase et al. examined notions of malleability for non-interactive proof systems. They defined the notion of allowable transformation $T = (T_x, T_w)$ w.r.t. a relation $R_{\mathcal{L}}$. A transformation is allowable w.r.t. $R_{\mathcal{L}}$ if on input $(x, w) \in R_{\mathcal{L}}$ it gives as output $(T_x(x) = x', T_w(w) = w') \in R_{\mathcal{L}}$. Then, a proof

[3] SMLERS were introduced in [2] as well. A SMLERS is a ring signature which is also extendable. In addition, it allows to link two signatures produced by the same signer on the same message, even on different rings. The SMLERS of [2] is obtained from signatures of knowledge for the discrete log relation, collision-resistant hash functions, and the discrete log assumption.

system is said to be malleable w.r.t. an allowable transformation $T = (T_x, T_w)$, if there exists a poly-time algorithm that on input the initial statement x, the transformation T, and an accepting proof Π, gives an accepting proof Π' for the transformed statement x'. They also considered more complex transformations including n statements and proofs. They showed that Groth-Sahai (GS) proofs [24] are malleable w.r.t. the language of sets of pairing product equations and they define a set of elementary allowable transformations which can be used to build more complex ones, including conjunctions and disjunctions. They also observed that since GS is re-randomizable, a transformation of a proof followed by its re-randomization is indistinguishable from a proof computed from scratch for statement x' using witness w'. They called this property derivation privacy.

In this paper, we further explore the notion of malleability for non-interactive witness indistinguishable (NIWI) proofs of knowledge (PoKs) in the context of threshold relations. A threshold relation $R_{\mathcal{L}^{tr}}$ is defined w.r.t. a relation $R_{\mathcal{L}}$ as $R_{\mathcal{L}^{tr}} = \{(x = (k, x_1, \dots, x_n), w = ((w_1, \alpha_1), \dots, (w_k, \alpha_k)))|1 \leq \alpha_1 < \dots < \alpha_k \leq n \wedge \ \forall\ j \in [k] : (x_{\alpha_j}, w_j) \in R_{\mathcal{L}}\}$. Let $\mathcal{L}^{tr}$ be the corresponding NP language. In words, the prover wants to prove it has k witnesses for k *different* statements out of n statements. The transformations we explore are extend and add operations:

- **Extend**: transform a proof for $(k, x_1, \dots, x_n) \in \mathcal{L}^{tr}$ into a proof for $(k, x_1, \dots, x_n, x_{n+1}) \in \mathcal{L}^{tr}$.
- **Add**: transform a proof for $(k, x_1, \dots, x_n) \in \mathcal{L}^{tr}$ into a proof for $(k + 1, x_1, \dots, x_n) \in \mathcal{L}^{tr}$.

While the extend operation can be realized without using any private input of the "previous" prover, as modelled in [14], the same does not hold for the add operation. Indeed, thanks to extractability, an accepting proof for $(k + 1, x_1, \dots, x_n) \in \mathcal{L}^{tr}$ can only be generated by the prover, except with negligible probability, using $k + 1$ witnesses for $k + 1$ different statements out of all the n statements. It follows that the add transformation must require a witness for statement x_i, with index $i \in [n]$ that was not previously used, and it cannot produce an accepting proof for the updated statement on input a witness for a previously used index. It is straightforward to notice that this fact could be used to check whether or not a given witness was used in the proof, thus violating witness indistinguishability.

Therefore, we put forth the new notion of ENIWI PoK. In an ENIWI PoK, when the prover computes a proof Π for a statement $x = (k, x_1, \dots, x_n)$, it also outputs a list of auxiliary values $\mathsf{AUX} = (\mathsf{aux}_1, \dots, \mathsf{aux}_n)$. The auxiliary value aux_i will be later used to perform the add operation via an additional algorithm called PrAdd. PrAdd, on input an accepting proof Π for $(k, x_1, \dots, x_n) \in \mathcal{L}^{tr}$, a witness w_i for a not previously used index i s.t. $(x_i, w_i) \in R_{\mathcal{L}}$, and the corresponding auxiliary value aux_i, outputs a proof Π' for $(k + 1, x_1, \dots, x_n) \in \mathcal{L}^{tr}$. Analogously, there is an additional algorithm $\mathsf{PrExtend}$ that is used to perform the extend operation. $\mathsf{PrExtend}$ does not require any auxiliary value. $\mathsf{PrExtend}$, on input an accepting proof for $(k, x_1, \dots, x_n) \in \mathcal{L}^{tr}$, and a statement x_{n+1}, outputs a proof Π' for $(k, x_1, \dots, x_{n+1}) \in \mathcal{L}^{tr}$ and the auxiliary value aux_{n+1} related to statement x_{n+1}. The auxiliary value aux_{n+1} can later be used to perform an add operation using witness w_{n+1} s.t. $(x_{n+1}, w_{n+1}) \in R_{\mathcal{L}}$. The verification algorithm is left unaltered and does not take any auxiliary value in input.

Similarly to derivation privacy, we require that the outputs of both the extend and add operations followed by a re-randomization are indistinguishable from proofs created using the regular prover algorithm. Regarding witness indistinguishability, we have to treat the auxiliary values in a special manner. Indeed, giving out all the auxiliary values would at least reveal the indices of the used witnesses. Therefore, we propose a new notion called extended witness indistinguishability. In this notion, the adversary $\mathcal{A}$ samples a $x = (k, x_1, \ldots, x_n)$ and two witnesses w^i as $((w_1^i, \alpha_1^i) \ldots, (w_k^i, \alpha_k^i))$, s.t. $(x, w^i) \in R_{\mathcal{L}^{tr}}$ for $i \in \{0, 1\}$. Recall that $\alpha_j \in [n]$, with $j \in [k]$, indicates that w_j is a witness s.t. $(x_{\alpha_j}, w_j) \in R_{\mathcal{L}}$. Then, the challenger $\mathcal{C}$ outputs a proof computed using one of the two witnesses, but it only gives to $\mathcal{A}$ a *subset* of all the auxiliary values. Such subset includes the auxiliary values only related to certain indices, namely $(\{1, \ldots, n\} \setminus (\{\alpha_1^0, \ldots, \alpha_n^0\} \cup \{\alpha_1^1, \ldots, \alpha_n^1\})) \cup (\{\alpha_1^0, \ldots, \alpha_n^0\} \cap \{\alpha_1^1, \ldots, \alpha_n^1\})$. In words, those are the auxiliary values related to the indices for which one of the following conditions holds: (i) the index was not used in either w^0 or w^1; (ii) the index was used in both w^0 and w^1. We require that $\mathcal{A}$ has negligible advantage in guessing whether w^0 or w^1 was used to create the proof. The idea is that if we build upon a NIWI and if the auxiliary values are only tied to the *indices* of the used witness and not to their concrete values, then giving the auxiliary values for the "irrelevant" positions to $\mathcal{A}$ does not give $\mathcal{A}$ any advantage. Although it could seem a cumbersome notion, ENIWI is enough to obtain strongly anonymous ETRS, and could possibly have other applications.

High-level overview of our ENIWI. We propose an ENIWI for the base relation $R_{\mathcal{L}}$ of pairing product equations (PPEs) in which all the variables are elements of group two, public constants are either paired with secret values or with the public generator, and the target element is the neutral element.

We build our ENIWI from GS proofs. GS is a commit-and-prove system where secret variables are first committed and the prover algorithm takes as input the committed values as well as the commitments randomnesses to create some proof elements. The proof can be verified on input the statement, the commitments, and proof elements. We first modify known techniques to get disjunctions of PPEs [23,13] into a technique to get proofs of partial satisfiability of k out of n PPEs. Such transformation modifies the starting PPEs via some additional variables $\hat{M}_i$ with $i \in [n]$ s.t. k of the PPEs are left unaltered while $n - k$ of them now admit the trivial solution, thus allowing for simulation. The value of $\hat{M}_i$ is constrained to two values, depending on whether or not the proof for the i-th equation should be simulated. We then observe that such proofs can be turned into an ENIWI provided with the extend and the add operations. The auxiliary values can be seen as the commitment openings related to such variables which enable to replace an $\hat{M}_i$ allowing for simulation (i.e., an $\hat{M}_i$ that makes the corresponding PPE admit the trivial solution) with a new one preventing simulation (i.e., an $\hat{M}_i$ that leaves the corresponding PPE unaltered). The idea is that to perform the add operation, the old commitment to a variable

$\hat{M}_i$ would be replaced with a fresh one. Then, aux_i would allow to erase from the proof element the contribution related to the old committed variable and to subsequently put in the contribution of the freshly committed variable. The extend operation is more straightforward since it does not need to erase any contribution, but only to add the contribution of a new variable. At a high level, extended witness indistinguishability is achieved since the $\hat{M}_i$ variables are only tied to the particular equation being simulated or not, but not to the actual value of any of the variables. Proofs can also be re-randomized leveraging the re-randomizability of GS and by appropriately updating the auxiliary values after the re-randomization.

High-level overview of our ETRS. To get an ETRS, we just need a way to turn an ENIWI in a signature scheme preserving its extendability properties. In [20], it is shown how to create a signature of knowledge (SoK) from a NIWI PoK in the random oracle model (ROM). In a nutshell, the message is hashed to produce the CRS which is then used to prove the statement of the SoK. The resulting proof constitutes the signature. We leverage their technique to create an ETRS starting from an ENIWI PoK. The idea is that since the transformation given in [20] just modifies how the CRS is generated, we are able to replace the NIWI PoK with an ENIWI PoK to get an ETRS instead of a regular signature. In our ETRS, the i-th signer has as public key a statement x_i for a hard relation $R_{\mathcal{L}}$ for which it exists an ENIWI, along with the public key $\mathsf{pk}_{\mathsf{e}}^i$ of an IND-CPA public key encryption scheme (PKE) which is homomorphic w.r.t. the update operation of the auxiliary values. The corresponding secret key is w_i s.t. $(x_i, w_i) \in R_{\mathcal{L}}$, along with the secret key of the encryption scheme $\mathsf{sk}_{\mathsf{e}}^i$. The first signer S hashes the message m to get the CRS, then S uses her own witness to create a proof for $(1, x_1, \ldots, x_n) \in R_{\mathcal{L}^{tr}}$. By creating such proof, the signer will also get auxiliary values $(\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$. Since publishing the auxiliary values in the clear would reveal the identity of the signer, each individual aux_i is encrypted using the public key of the i-th signer[4]. A new signer willing to join will decrypt aux_i and run PrAdd to update the proof. To extend the ring, it suffices to run $\mathsf{PrExtend}$ to update the proof. Finally, to ensure anonymity we exploit the fact that ENIWI PoKs are re-randomizable. We re-randomize all the proofs after running either PrAdd or $\mathsf{PrExtend}$. We additionally exploit the homomorphic property of the encryption scheme to update the auxiliary values after each re-randomization. We prove the security of our ETRS in the ROM.

Both the constructions presented in [2] use SoKs for the discrete log relation as a building block without specifying a concrete instantiation. Whether they require the ROM or not depends on whether there exists a practical[5] SoK without

[4] Notice that for anonymity to hold, it is crucial that the witness indistinguishability property holds even if the auxiliary values related to unused positions are accessible by the adversary. Indeed, in our anonymity notion the adversary is allowed to corrupt all non-signers, thus getting their decryption keys and the related auxiliary values.

[5] Chase and Lysyanskaya [15] proposed a generic construction under standard complexity assumptions in the common random string model, but it is not practical since it uses general non-interactive zero-knowledge (NIZK) proofs.

random oracles for that relation. The authors also provide an implementation in which they use the Schnorr identification scheme with the Fiat-Shamir transform as a SoK. Such SoK requires the ROM. In our ETRS, all operations require linear time in n as the number of equations to be proved linearly depends on n. Additionally, GS proofs have constant size for each type of equation, therefore the size of the ETRS is $\mathcal{O}(n)$. Note that both time complexity and signature size do not depend on t.

2 Related Work

Threshold ring signatures were introduced by Bresson et al. [12]. They provided a construction based on RSA. The size of the signature is $\mathcal{O}(n \log n)$, where n is the size of the ring. Subsequent works proposed new constructions from a variety of assumptions focused on either relaxing the setup assumptions, reducing the signature size, or getting rid of the ROM.

Several works have signatures of size linear in n [1,32,26], while some others proposed constructions with signature size that can be sub-linear in n [36,4,6][6], or even $\mathcal{O}(t)$ [29,25]. Some works have also focused on providing post-quantum security [1,8,26].

In [31], the concept of flexibility was introduced. A flexible threshold ring signature scheme allows one to modify an already created signature on a message m with threshold t and ring $\mathcal{R}$ into a new signature on message m with threshold $t+1$ w.r.t. $\mathcal{R}$, without the intervention of the previous signers.

Usually, threshold ring signatures are formulated as an interactive protocol run among the signers. Some schemes have a weaker requirement [4,6], where the signers just have to interact with one party called the aggregator. After having interacted with all the signers, the aggregator just compiles all the received contributions into one threshold ring signatures which can then be publicly posted. Munch-Hansen et al. [29] presented a threshold ring signature based on RSA accumulators with size $\mathcal{O}(t)$. Their scheme also achieves flexibility. Moreover, they introduce a stronger anonymity property that demands that a signer cannot be deanonymized even by their fellow signers. In this scenario, having non-interactive signing is crucial since the deanonymization could be done by exploiting communication meta-data such as the IP address. The same applies to signatures using an aggregator, unless the aggregator is trusted. Recently, Aranha et al. [2] have further enhanced the functionality of threshold ring signature by proposing extendable threshold ring signatures ETRS. ETRS are flexible and they also allow to extend the ring of a given signature without the need of any secret.

3 Preliminaries

In this section, we introduce the assumptions and the cryptographic tools our constructions rely on. We defer to the full version [5] for more widely known defi-

[6] In particular, [36] has size $\mathcal{O}(t\sqrt{n})$, [6] is $\mathcal{O}(t \log n)$, and [4] is $\mathcal{O}(\log n)$.

nitions and assumptions. When referring to an NP language $\mathcal{L}$ we call $R_{\mathcal{L}}$ the corresponding NP relation. We work over bilinear groups $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$. $\mathcal{G}(1^\lambda)$ is a generator algorithm that on input the security parameter, outputs the description of a bilinear group. We call such description group key gk. $\hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}$ are prime p order groups, $\hat{g}, \check{h}$ are generators of $\hat{\mathbb{G}}, \check{\mathbb{H}}$ respectively, and $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \rightarrow \mathbb{T}$ is a non-degenerate bilinear map. In this paper, we will use additive notation for the group operations and multiplicative notation for the bilinear map e.

Assumption 1 (Double Pairing Fixed Term Assumption) *We say the double pairing fixed term assumption holds relative to $\hat{\mathbb{G}}$ if for $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$, and for all PPT adversaries $\mathcal{A}$ we have*

$$\Pr\Big[\hat{a}, \hat{b} \leftarrow\!\!\$\, \hat{\mathbb{G}} \setminus (\hat{0}, \hat{0}); \check{b}' \leftarrow \mathcal{A}(\mathsf{gk}, \hat{a}, \hat{b}) : \check{b}' \in \check{\mathbb{H}}, \hat{a} \cdot \check{h} + \hat{b} \cdot \check{b}' = 0_{\mathbb{T}}\Big] \leq \mathsf{negl}(\lambda).$$

Lemma 1. *If the double pairing fixed term assumption holds for* gk*, then the Decisional Diffie-Hellman assumption holds for* $\hat{\mathbb{G}}$*.*

See [5] for the proof.

3.1 Groth-Sahai Proofs

The Groth-Sahai proof system [24] is a proof system for the language of satisfiable equations (of types listed below) over a bilinear group $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$. The prover wants to show that there is an assignment of all the variables that satisfies the equation. Such equations can be of four types:

Pairing-product equations (PPE): For public constants $\hat{a}_j \in \hat{\mathbb{G}}$, $\check{b}_i \in \check{\mathbb{H}}$, $\gamma_{ij} \in \mathbb{Z}_p$, $t_{\mathbb{T}} \in \mathbb{T}$: $\sum_i \hat{x}_i \cdot \check{b}_i + \sum_j \hat{a}_j \cdot \check{y}_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i \cdot \check{y}_j = t_{\mathbb{T}}$.

Multi-scalar multiplication equation in $\hat{\mathbb{G}}$ (ME$_{\hat{\mathbb{G}}}$): For public constants $\hat{a}_j \in \hat{\mathbb{G}}$, $b_i \in \mathbb{Z}_p$, $\gamma_{ij} \in \mathbb{Z}_p$, $\hat{t} \in \hat{\mathbb{G}}$: $\sum_i \hat{x}_i b_i + \sum_j \hat{a}_j y_j + \sum_i \sum_j \gamma_{ij} \hat{x}_i y_j = \hat{t}$.

Multi-scalar multiplication equation in $\check{\mathbb{H}}$ (ME$_{\check{\mathbb{H}}}$): For public constants $a_j \in \mathbb{Z}_p$, $\check{b}_i \in \check{\mathbb{H}}$, $\gamma_{ij} \in \mathbb{Z}_p$, $\check{t} \in \check{\mathbb{H}}$: $\sum_i x_i \check{b}_i + \sum_j a_j \check{y}_j + \sum_i \sum_j \gamma_{ij} x_i \check{y}_j = \check{t}$.

Quadratic equation in $\mathbb{Z}_p$ (QE): For public constants $a_j \in \mathbb{Z}_p$, $b_i \in \mathbb{Z}_p$, $\gamma_{ij} \in \mathbb{Z}_p$, $t \in \mathbb{Z}_p$: $\sum_i x_i b_i + \sum_j a_j y_j + \sum_i \sum_j \gamma_{ij} x_i y_j = t$.

Here, we formalize the GS proof system as in [18]. The GS proof system is a commit-and-prove system. Each committed variable is also provided with a public label that specifies the type of input (i.e., scalar or group element). Accordingly, the prover algorithm takes as input a label L which indicates the type of equation to be proved (i.e., $L \in \{\mathrm{PPE}, \mathrm{ME}_{\hat{\mathbb{G}}}, \mathrm{ME}_{\check{\mathbb{H}}}, \mathrm{QE}\}$). GS features the following PPT algorithms, the common reference string crs and the group key gk are considered as implicit input of all the algorithms.

- $\mathsf{crs} \leftarrow_{\$} \mathsf{CRSSetup}(\mathsf{gk})$: on input the group key, output the common reference string.
- $(l, c) \leftarrow \mathsf{Com}(l, w; r)$: return a commitment (l, c) to message w according to the label l and randomness r.
- $\pi \leftarrow \mathsf{Prove}(L, x, (l_1, w_1, r_1), \ldots, (l_n, w_n, r_n))$: consider statement x as an equation of type specified by L, and on input a list of commitment openings produce a proof π.
- $0/1 \leftarrow \mathsf{PrVerify}(x, (l_1, c_1), \ldots, (l_n, c_n), \pi)$: given committed variables, statement x, and proof π, output 1 to accept and 0 to reject.
- $((l_1, c'_1), \ldots, (l_n, c'_n), \pi') \leftarrow \mathsf{RandPr}(L, (l_1, c_1), \ldots, (l_n, c_n), \pi; r)$: on input equation type specified by L, a list of commitments, a proof π, and a randomness r, output a re-randomized proof along with the corresponding list of re-randomized commitments.

GS can be also used to prove that a set of equations S, with possibly shared variables across the equations, has a satisfying assignment. To do so, the prover has to reuse the same commitments for the shared variables while executing the Prove algorithm for each individual equation. The above description can also fit the interface of NIWI PoK (see App. A.2 of [5]). Indeed, the Prove algorithm can just launch the Com and the Prove algorithm above with the appropriate labels, and return as a proof both the commitments and the proof elements. Similarly, the PrVerify and RandPr algorithms of the NIWI PoK interface have just to appropriately parse their inputs and call the PrVerify and RandPr algorithms described above.

The GS proof system is proved to be a NIWI for all types of the above equations under the SXDH assumption. In addition, it is a NIWI PoK for all equations involving solely group elements. To be more specific, Escala and Groth formulated the notion of F-knowledge[18] (i.e., a variation of adaptive extractable soundness, see Def. 14 of [5]) for a commit-and-prove system. In a nutshell, it requires the existence of an Ext_2 algorithm that, on input a valid commitment and the extraction key produced by Ext_1, outputs a function F of the committed value. They prove that GS enjoys F-knowledge. For commitments to group elements, F is identity function. Regarding commitments to scalars, F is a one-way function that uniquely determines the committed value.

Internals of GS proofs. In [18], the authors provide a very fine-grained description of GS proofs. In this description, we report only the aspects that are relevant to our constructions. It is possible to write the equations of Sec. 3.1 in a more compact way. Consider $\hat{\boldsymbol{x}} = (\hat{x}_1, \ldots, \hat{x}_m)$ and $\check{\boldsymbol{y}} = (\check{y}_1, \ldots, \check{y}_n)$, which may be both public constants (i.e., written before as $\hat{a}_j, \check{b}_i$) or secret values. Let $\Gamma = \{\gamma_{ij}\}_{i=1,j=1}^{m,n} \in \mathbb{Z}_p^{m \times n}$. We can now write a PPE as $\hat{\boldsymbol{x}}\Gamma\check{\boldsymbol{y}} = t_{\mathbb{T}}$. Similarly, a $\mathrm{ME}_{\hat{\mathbb{G}}}$, a $\mathrm{ME}_{\check{\mathbb{H}}}$, and a QE can be written as $\hat{\boldsymbol{x}}\Gamma\boldsymbol{y} = \hat{t}$, $\boldsymbol{x}\Gamma\check{\boldsymbol{y}} = \check{t}$, and $\boldsymbol{x}\Gamma\boldsymbol{y} = t$. This holds for $\hat{\boldsymbol{x}} \in \hat{\mathbb{G}}^{1\times m}, \check{\boldsymbol{y}} \in \check{\mathbb{H}}^{n\times 1}, \boldsymbol{x} \in \mathbb{Z}_p^{1\times m}, \boldsymbol{y} \in \mathbb{Z}_p^{n\times 1}$. Additionally, for equations of type $\mathrm{ME}_{\hat{\mathbb{G}}}$, $\mathrm{ME}_{\check{\mathbb{H}}}$, and QE, we can, without loss of generality, assume the target element to be the neutral element. For PPE we will restrict ourselves to the case in which $t_{\mathbb{T}} = 0_{\mathbb{T}}$, and no public constants are paired with each other, unless one of the two is a generator specified in the public parameters. The structure of the crs is clear from Fig. 1, where the Ext_1 algorithm is shown.

```
$(\mathsf{crs}, xk) \leftarrow \mathsf{Ext}_1(\mathsf{gk})$
Parse $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$
$\rho \leftarrow\!\!\$\ \mathbb{Z}_p, \xi \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and $\sigma \leftarrow\!\!\$\ \mathbb{Z}_p, \psi \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$\hat{\boldsymbol{v}} = (\xi\hat{g}, \hat{g})^\top$ and $\check{\boldsymbol{v}} = (\psi\check{h}, \check{h})$
$\hat{\boldsymbol{w}} = \rho\hat{\boldsymbol{v}}$ and $\check{\boldsymbol{w}} = \sigma\check{\boldsymbol{v}}$
$\hat{\boldsymbol{u}} = \hat{\boldsymbol{w}} + (\hat{0}, \hat{g})^\top$ and $\check{\boldsymbol{u}} = \check{\boldsymbol{w}} + (\check{0}, \hat{g})$
$\boldsymbol{\xi} = (-\xi^{-1} \mod p, 1)$ and
$\boldsymbol{\psi} = (-\psi^{-1} \mod p, 1)^\top$
$\mathsf{crs} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$
$xk = (\boldsymbol{\xi}, \boldsymbol{\psi})$
return $(\mathsf{crs}, xk)$
```

Fig. 1. Generation of the CRS along with the extraction key in the GS proof system.

In Fig. 2, we report the commitment labels and corresponding commit algorithm that are of interest for this work.

Input	Randomness	Output
$\mathsf{pub}_{\hat{\mathbb{G}}}, \hat{x}$	$r = 0, s = 0$	$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x}$
$\mathsf{com}_{\hat{\mathbb{G}}}, \hat{x}$	$r, s \leftarrow\!\!\$\ \mathbb{Z}_p$	$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{x} + \hat{\boldsymbol{v}}r + \hat{\boldsymbol{w}}s$
$\mathsf{base}_{\hat{\mathbb{G}}}, \hat{g}$	$r = 0, s = 0$	$\hat{\boldsymbol{c}} = \boldsymbol{e}^\top \hat{g}$
$\mathsf{sca}_{\hat{\mathbb{G}}}, x$	$r \leftarrow\!\!\$\ \mathbb{Z}_p, s = 0$	$\hat{\boldsymbol{c}} = \hat{\boldsymbol{u}}x + \hat{\boldsymbol{v}}r$

Input	Randomness	Output
$\mathsf{pub}_{\check{\mathbb{H}}}.\check{y}$	$r = 0, s = 0$	$\check{\boldsymbol{d}} = \check{y}\boldsymbol{e}$
$\mathsf{com}_{\check{\mathbb{H}}}, \check{x}$	$r, s \leftarrow\!\!\$\ \mathbb{Z}_p$	$\check{\boldsymbol{d}} = \check{y}\boldsymbol{e} + r\check{\boldsymbol{v}} + s\check{\boldsymbol{w}}$
$\mathsf{base}_{\check{\mathbb{H}}}, \check{h}$	$r = 0, s = 0$	$\check{\boldsymbol{d}} = \check{h}\boldsymbol{e}$
$\mathsf{sca}_{\check{\mathbb{H}}}, y$	$r \leftarrow\!\!\$\ \mathbb{Z}_p, s = 0$	$\check{\boldsymbol{d}} = y\check{\boldsymbol{u}} + r\check{\boldsymbol{v}}$

Fig. 2. GS commit labels and corresponding commit algorithm, $\boldsymbol{e} = (0, 1)$.

In Fig. 3 and in Fig. 4, we report the prover and verifier algorithm respectively. We defer to App. A.4 of [5] for more details on GS internals.

4 Extendable Threshold Ring Signature

A non-interactive extendable threshold ring signature scheme ETRS is defined as a tuple of six PPT algorithms $\mathsf{ETRS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Join}, \mathsf{Extend})$, where the public parameters pp produced by Setup are implicitly available to all the other algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: on input the security parameter, outputs public parameters pp.

```
$\mathsf{Prove}(L, \varGamma, \{(l_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^{m}, \{(l_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^{n})$
--------------------------------------------------------------------------------
**if** $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ define $\hat{C} = \boldsymbol{e}^\top \boldsymbol{x} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x$ **else if** $\boldsymbol{x} \in \mathbb{Z}_p^m$ define $\hat{C} = \hat{\boldsymbol{u}}\boldsymbol{x} + \hat{\boldsymbol{v}}\boldsymbol{r}_x$
**if** $\boldsymbol{y} \in \check{\mathbb{H}}^n$ define $\check{D} = \boldsymbol{e}^\top \boldsymbol{y} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}}$ **else if** $\boldsymbol{y} \in \mathbb{Z}_p^n$ define $\check{D} = \check{\boldsymbol{u}}\boldsymbol{y} + \boldsymbol{r}_y\check{\boldsymbol{v}}$
Set $\alpha = \beta = \gamma = \delta = 0$
**if** $L = \mathrm{PPE}$ $\alpha, \beta, \gamma, \delta \leftarrow\!\$\, \mathbb{Z}_p$
**if** $L = \mathrm{ME}_{\hat{\mathbb{G}}}$ $\alpha, \beta \leftarrow\!\$\, \mathbb{Z}_p$
**if** $L = \mathrm{ME}_{\check{\mathbb{H}}}$ $\alpha, \gamma \leftarrow\!\$\, \mathbb{Z}_p$
**if** $L = \mathrm{QE}$ $\alpha \leftarrow\!\$\, \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x \varGamma \check{D} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$        $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} = (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\varGamma \boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x \varGamma \check{D} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$        $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} = (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\varGamma \boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$
**return** $\boldsymbol{\pi} = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}})$
```

Fig. 3. Prover algorithm of the GS proof system.

```
$\mathsf{PrVerify}(L, \varGamma, \{(l_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^{m}, \{(l_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^{n}), \boldsymbol{\pi})$
--------------------------------------------------------------------------------
Check that the equation has a valid format.
Check $\hat{C} = (\hat{\boldsymbol{c}}_1 \dots \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2\times m}$ and $\check{D} = (\check{\boldsymbol{d}}_1 \dots \check{\boldsymbol{d}}_n)^\top \in \check{\mathbb{H}}^{n\times 2}$
Check $\boldsymbol{\pi} = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}) \in \check{\mathbb{H}}^{2\times 1} \times \check{\mathbb{H}}^{2\times 1} \times \hat{\mathbb{G}}^{1\times 2} \times \hat{\mathbb{G}}^{1\times 2}$
Check $\hat{C}\varGamma\check{D} = \hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + \hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}\check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}\check{\boldsymbol{w}}$
**return** 1 if and only if all checks pass and 0 otherwise.
```

Fig. 4. Verifier algorithm of the GS proof system.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$: generates a new public and secret key pair.
- $\sigma \leftarrow \mathsf{Sign}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk})$: returns a signature with threshold $t = 1$ using the secret key sk corresponding to a public key pk_i with $i \in \mathcal{R}$.
- $0/1 \leftarrow \mathsf{Verify}(t, m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \sigma)$: verifies a signature σ for the message m against the public keys $\{\mathsf{pk}_i\}_{i\in\mathcal{R}}$ with threshold t. Outputs 1 to accept, and 0 to reject.
- $\sigma' \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk}, \sigma)$: it takes as input a signature σ for message m produced w.r.t. ring $\mathcal{R}$ with threshold t, and the new signer secret key sk (whose corresponding pk is included in $\mathcal{R}$). It outputs a new signature σ' with threshold $t + 1$.
- $\sigma' \leftarrow \mathsf{Extend}(m, \sigma, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \{\mathsf{pk}_i\}_{i\in\mathcal{R}'})$: extends the signature σ with threshold t for the ring $\mathcal{R}$ into a new signature σ' with threshold t for the larger ring $\mathcal{R} \cup \mathcal{R}'$.

To formalize the properties of ETRS, we use the notion of ladder as in [2]. A ladder lad is a sequence of tuples $(\mathsf{action}, \mathsf{input})$, where action takes a value in the set $\{\mathsf{Sign}, \mathsf{Join}, \mathsf{Extend}\}$ and the value of input depends on the value of action. If $\mathsf{action} = \mathsf{Sign}$, then input is a pair $(\mathcal{R}, i)$, where $\mathcal{R}$ is the ring for the signature and i is the signer's identity. If $\mathsf{action} = \mathsf{Join}$, then input is an identifier i that identifies the signer that joins the signature. If $\mathsf{action} = \mathsf{Extend}$, then input is a ring $\mathcal{R}$ that is the ring to use to extend the previous ring. We notice that a ladder unequivocally determines a sequence of ETRS, each one with a specific ring and threshold value. In Fig. 7, the algorithm Proc is described. Proc takes as input a message, a ladder, and a corresponding list of keys, and outputs the sequence of all the signatures that correspond to each step of the ladder. It outputs $\perp$ whenever the ladder has an inconsistent sequence of actions or is incompatible with the list of keys provided in the input.

Definition 1 (Correctness for ETRS). *For all $\lambda \in \mathbb{N}$, for any message $m \in \{0,1\}^*$, for any ladder lad of polynomial size identifying a ring $\mathcal{R}$, it holds that:*

$$\Pr\left[\begin{array}{c|c} \left(\bigwedge_{j=1}^{\ell} \mathsf{Verify}(t, m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \sigma_j) = 1\right) & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ & \mathsf{L_{keys}} \leftarrow \{\mathsf{KeyGen}()\}_{i\in\mathcal{R}}; \\ \vee(\Sigma, t, \mathcal{R}) = \perp & (\Sigma, t, \mathcal{R}) \leftarrow \mathsf{Proc}(m, \mathsf{L_{keys}}, \mathsf{lad}); \\ & \{\sigma_1, \ldots, \sigma_\ell\} = \Sigma \end{array}\right] = 1.$$

Definition 2 (Unforgeability for ETRS). *An extendable threshold ring signature scheme ETRS is said to be unforgeable if for all PPT adversaries $\mathcal{A}$, the success probability in the experiment of Fig. 5 is*

$$\Pr\left[\mathsf{Exp}^{cmEUF}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\right] \leq \mathsf{negl}(\lambda).$$

Definition 3 (Anonymity for ETRS). *An extendable threshold ring signature scheme ETRS is said to provide anonymity if for all PPT adversaries $\mathcal{A}$, the success probability in the anonymous extendability experiment of Fig. 6 is $\Pr\left[\mathsf{Exp}^{ANEXT}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. In this experiment, the ladders submitted by $\mathcal{A}$ are said to be well-formed if all the actions in the ladders are pairwise of the same type, and they have the same ring as input.*

Remarks on anonymity and unforgeability for ETRS. We modify the definition of anonymity for ETRS in [2] making it stronger. The difference is that the adversary now gets to see all the intermediate ETRS instead of just the final one (see lines 11 and 12 of Chal in Fig. 6). This modification enables count-me-in applications where partial signatures get publicly posted. In addition, in the experiment, we add the checks of lines 15 and 17 to rule out a trivial attack inherent to any ETRS. Indeed, since the Join operation cannot increase the threshold of an ETRS when using a secret key that was already used before, $\mathcal{A}$ could use this fact to distinguish between the ladders.

The $\mathsf{Combine}$ algorithm is introduced in [2] as a procedure to combine together two signatures on the same message with two different (not necessarily disjoint) rings. The output is a signature having as ring the union of the two rings and as threshold the cardinality of the union of the signers sets of the starting signatures. The $\mathsf{Combine}$ algorithm can be run without knowing any secret key. In [2], the authors showed that the Join operation can be obtained

```
$\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
1 : $\mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}}, \mathsf{L_{join}} \leftarrow \emptyset$
2 : $\mathsf{pp} \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
3 : $\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$
4 : $(t^*, m^*, \mathcal{R}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
5 : $q_1 \leftarrow |\{(m^*, \mathcal{R}, \cdot) \in \mathsf{L_{sign}} : \mathcal{R} \subseteq \mathcal{R}^*\}|$
6 : $q_2 \leftarrow |\{(m^*, i, \cdot) \in \mathsf{L_{join}} : i \in \mathcal{R}^*\}|$
7 : $q \leftarrow q_1 + q_2$
8 : if $|\mathcal{R}^* \cap \mathsf{L_{corr}}| + q \geq t^*$
9 :     return $lose$
10 : if $\mathsf{Verify}(t^*, m^*, \{\mathsf{pk}_j\}_{j\in\mathcal{R}^*}, \sigma^*) = 0$
11 :     return $lose$
12 : return $win$

$\mathsf{OSign}(m, \mathcal{R}, i)$
1 : if $(i \in \mathsf{L_{corr}} \vee i \notin \mathcal{R})$ return $\perp$
2 : for $j \in \mathcal{R}$
3 :     if $(j, pk_j, \cdot) \notin \mathsf{L_{keys}}$
4 :         return $\perp$
5 : $\sigma \leftarrow \mathsf{ETRS.Sign}(m, \{\mathsf{pk}_j\}_{j\in\mathcal{R}}, \mathsf{sk}_i)$
6 : $\mathsf{L_{sign}} \leftarrow \mathsf{L_{sign}} \cup \{(m, \mathcal{R}, i)\}$
7 : return $\sigma$

$\mathsf{OKey}(i, \mathsf{pk})$
1 : if $\mathsf{pk} = \perp$
2 :     $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{ETRS.KeyGen}()$
3 :     $\mathsf{L_{keys}} \leftarrow \mathsf{L_{keys}} \cup \{(i, \mathsf{pk}_i, \mathsf{sk}_i)\}$
4 : else
5 :     $\mathsf{L_{corr}} \leftarrow \mathsf{L_{corr}} \cup \{i\}$
6 :     $\mathsf{pk}_i \leftarrow \mathsf{pk}$
7 :     $\mathsf{L_{keys}} \leftarrow \mathsf{L_{keys}} \cup \{(i, \mathsf{pk}_i, \perp)\}$
8 : return $\mathsf{pk}_i$

$\mathsf{OCorr}(i)$
1 : if $(i, \mathsf{pk}_i, \mathsf{sk}_i) \in \mathsf{L_{keys}} \wedge \mathsf{sk}_i \neq \perp$
2 :     $\mathsf{L_{corr}} \cup \{i\}$
3 :     return $(\mathsf{pk}_i, \mathsf{sk}_i)$
4 : return $\perp$

$\mathsf{OJoin}(m, \mathcal{R}, i, \sigma)$
1 : if $i \in \mathsf{L_{corr}}$ return $\perp$
2 : for $j \in \mathcal{R}$
3 :     if $((j, pk_j, \cdot) \notin \mathsf{L_{keys}})$
4 :         return $\perp$
5 : $\sigma' \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_j\}_{j\in\mathcal{R}}, \mathsf{sk}_i, \sigma)$
6 : $\mathsf{L_{join}} \leftarrow \mathsf{L_{join}} \cup \{(m, i, \sigma)\}$
7 : return $\sigma'$
```

Fig. 5. Unforgeability game for ETRS (security experiment and oracles). This notion is reported from [2].

as the concatenation of the Sign operation and the Combine operation. In order to avoid the same attack described before, the checks in lines 11 and 13 of the experiment of Fig. 6 are needed. We notice that our ETRS only provides a weaker form of Combine in which the starting rings are disjoint (cfr., Sec. 6). A similar discussion holds for lines $5-8$ of the unforgeability experiment in Fig. 5. In particular, they rule out trivial attacks due to $\mathcal{A}$ asking too many sign, join, or corruption queries.

Fellow-signer anonymity. We also define a stronger version of anonymity called fellow-signer anonymity. This game models the requirement that even a signer cannot determine any of the other signers by just looking at all the signatures that were produced. It is straightforward to notice that fellow-signer anonymity implies anonymity for ETRS.

Definition 4 (Fellow Signer Anonymity for ETRS). *An extendable threshold ring signature scheme ETRS is said to provide fellow signer anonymity if for all PPT adversaries $\mathcal{A}$, the success probability in the experiment of Fig. 8 is* $\Pr\left[\mathsf{Exp}^{ANFS}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

5 Extendable Non-interactive Witness Indistinguishable Proof of Knowledge

Given an NP language $\mathcal{L}$ with associated poly-time relation $R_{\mathcal{L}}$, we define the related threshold relation $R_{\mathcal{L}^{tr}}$ as follows. We name the corresponding language $\mathcal{L}^{tr}$.

```
$\mathsf{Exp}^{\mathrm{ANEXT}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
 1 : $b \leftarrow\!\$\ \{0,1\}, \mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}}, \mathsf{L_{join}} \leftarrow \emptyset$
 2 : $pp \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
 3 : $\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$
 4 : $(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
 5 : $\Sigma \leftarrow \mathsf{Chal}_b(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*)$
 6 : $b^* \leftarrow \mathcal{A}^{\mathsf{O}}(\Sigma)$
 7 : if $\exists\, i \in \mathsf{lad}_0^*.\mathcal{S}\ s.t.\ i \in \mathsf{L_{corr}}$
 8 :   return $lose$
 9 : if $\exists\, i \in \mathsf{lad}_1^*.\mathcal{S}\ s.t.\ i \in \mathsf{L_{corr}}$
10 :   return $lose$
11 : if $\exists\, (m^*, \cdot, i) \in \mathsf{L_{sign}}$ for $i \in \mathsf{lad}_0^*.\mathcal{S}$
12 :   return $lose$
13 : if $\exists\, (m^*, \cdot, i) \in \mathsf{L_{sign}}$ for $i \in \mathsf{lad}_1^*.\mathcal{S}$
14 :   return $lose$
15 : if $\exists\, (m^*, i, \cdot) \in \mathsf{L_{join}}$ for $i \in \mathsf{lad}_0^*.\mathcal{S}$
16 :   return $lose$
17 : if $\exists\, (m^*, i, \cdot) \in \mathsf{L_{join}}$ for $i \in \mathsf{lad}_1^*.\mathcal{S}$
18 :   return $lose$
19 : if $b^* \neq b$ return $lose$
20 : return $win$
```

```
$\mathsf{Chal}_b(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*)$
 1 : if $(\mathsf{lad}_0^*, \mathsf{lad}_1^*)$ is not well-formed
 2 :   return $\perp$
 3 : if $\exists\, i \in \mathsf{lad}_0^*.\mathcal{S}\ s.t.\ i \in \mathsf{L_{corr}}$
 4 :   return $\perp$
 5 : if $\exists\, i \in \mathsf{lad}_1^*.\mathcal{S}\ s.t.\ i \in \mathsf{L_{corr}}$
 6 :   return $\perp$
 7 : $val_0 \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}_0^*)$
 8 : $val_1 \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}_1^*)$
 9 : if $val_0 = \perp \vee val_1 = \perp$
10 :   return $\perp$
11 : Parse $val_0$ as $(\Sigma_0, t_0, \mathcal{R}_0)$
12 : Parse $val_1$ as $(\Sigma_1, t_1, \mathcal{R}_1)$
13 : $\Sigma \leftarrow \Sigma_b$
14 : return $\Sigma$
```

Fig. 6. Anonymous extendability game. We use $\mathsf{lad}.\mathcal{S}$ to indicate the set of signers of a ladder lad. We propose a stronger notion compared to [2]. Indeed, in our definition, the adversary gets to see all the intermediate signatures instead of only the final ETRS.

```
Proc($m$, $\mathsf{L}_{\mathsf{keys}}$, $\mathsf{lad}$)
 1 : $\Sigma \leftarrow \emptyset, t = 0$
 2 : Parse $\mathsf{lad}$ as $((\mathsf{action}^1, \mathsf{input}^1), \ldots, (\mathsf{action}^l, \mathsf{input}^l))$
 3 : if $\mathsf{action}^1 \neq \mathsf{Sign}$ return $\perp$
 4 : else
 5 :   Parse $\mathsf{input}^1$ as $(\mathcal{R}^1, i^1)$
 6 :   for $j \in \mathcal{R}^1$ if $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$ return $\perp$
 7 :   if $sk_{i^1} = \perp \vee i^1 \notin \mathcal{R}^1$ return $\perp$
 8 :   $\mathcal{R} \leftarrow \mathcal{R}^1, \mathcal{S} \leftarrow \{i^1\}$
 9 :   $\sigma \leftarrow \mathsf{Sign}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_{i^1}), \Sigma \leftarrow \Sigma \cup \{\sigma\}$
10 : for $l' \in [2, \ldots, l]$
11 :   if $\mathsf{action}^{l'} = \mathsf{Sign}$ return $\perp$
12 :   else
13 :     if $\mathsf{action}^{l'} = \mathsf{Join}$ parse $\mathsf{input}^{l'}$ as $(i^{l'})$
14 :       if $i^{l'} \notin \mathcal{R} \vee i^{l'} \in \mathcal{S}$ return $\perp$
15 :       $\sigma \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_i^{l'}, \sigma)$
16 :       $\Sigma \leftarrow \Sigma \cup \{\sigma\}, \mathcal{S} \leftarrow \mathcal{S} \cup \{i^{l'}\}, t = t + 1$
17 :     if $\mathsf{action}^{l'} = \mathsf{Extend}$ parse $\mathsf{input}^{l'}$ as $(\mathcal{R}^{l'})$
18 :       for $j \in \mathcal{R}^{l'}$ if $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$ return $\perp$
19 :       $\sigma \leftarrow \mathsf{Extend}(m, \sigma, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \{\mathsf{pk}_j\}_{j \in \mathcal{R}^{l'}})$
20 :       $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}^{l'}, \Sigma \leftarrow \Sigma \cup \{\sigma\}$
21 :     else return $\perp$
22 : return $(\Sigma, t, \mathcal{R})$
```

Fig. 7. Process algorithm for ETRS.

$$\begin{aligned} R_{\mathcal{L}^{tr}} = \{ & (x = (k, x_1, \ldots, x_n), w = ((w_1, \alpha_1), \ldots, (w_k, \alpha_k)))| \\ & 1 \leq \alpha_1 < \ldots < \alpha_k \leq n \wedge \ \forall\, j \in [k] : (x_{\alpha_j}, w_j) \in R_{\mathcal{L}} \}. \end{aligned}$$

An extendable non-interactive proof system for a threshold relation $R_{\mathcal{L}^{tr}}$ consists of the following PPT algorithms. The group key $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$ is considered as an implicit input to all algorithms:

– $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$: on input the group key gk, output a uniformly random[7] common reference string $\mathsf{crs} \in \{0,1\}^\lambda$.

[7] Here we are also assuming that the crs is uniformly random since it is needed by our ETRS construction.

```
$\mathsf{Exp}^{\mathrm{ANFS}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
1 :  $b \leftarrow\!\!\$ \{0,1\}$
2 :  $\mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}}, \mathsf{L_{join}} \leftarrow \emptyset$
3 :  $pp \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
4 :  $\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$
5 :  $(m^*, \mathsf{lad}^*, i^*, j^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
6 :  $\Sigma \leftarrow \mathsf{Chal}_b(m^*, \mathsf{lad}^*, i^*, j^*)$
7 :  $b^* \leftarrow \mathcal{A}^{\mathsf{O}}(\Sigma)$
8 :  if $i^* \in \mathsf{L_{corr}} \lor j^* \in \mathsf{L_{corr}}$
9 :    return $lose$
10 : if $\exists (m^*, \cdot, i^*) \in \mathsf{L_{sign}} \lor (m^*, \cdot, j^*) \in \mathsf{L_{sign}}$
11 :   return $lose$
12 : if $\exists (m^*, i^*, \cdot) \in \mathsf{L_{join}} \lor (m^*, j^*, \cdot) \in \mathsf{L_{join}}$
13 :   return $lose$
14 : if $b^* \neq b$
15 :   return $lose$
16 : return $win$
```

```
$\mathsf{Chal}_b(m^*, \mathsf{lad}^*, i^*, j^*)$
1 :  if $i^* \in \mathsf{L_{corr}} \lor j^* \in \mathsf{L_{corr}}$
2 :    return $\perp$
3 :  $\mathsf{lad}^*.add((\mathsf{Extend}, \{i^*\}))$
4 :  $\mathsf{lad}^*.add((\mathsf{Extend}, \{j^*\}))$
5 :  if $b = 0$
6 :    $\mathsf{lad}^*.add((\mathsf{Join}, i^*))$
7 :  if $b = 1$
8 :    $\mathsf{lad}^*.add((\mathsf{Join}, j^*))$
9 :  $val \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}^*)$
10 : if $val = \perp$
11 :   return $\perp$
12 : else
13 :   Parse $val$ as $(\Sigma, t, \mathcal{R})$
14 : return $\Sigma$
```

Fig. 8. Fellow signer anonymity game. We use $\mathsf{lad}.\mathcal{S}$ to indicate the set of signers of a ladder lad and $\mathsf{lad}.add$ to indicate that we are adding the pair $(\mathsf{action}, \mathsf{input})$ as the last element of the ladder.

- $(\Pi, (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)))$: on input $((k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k))) \in R_{\mathcal{L}^{tr}}$, output a proof Π and auxiliary values $(\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$. The auxiliary value aux_i is used later on to perform an add operation using a witness for a not previously used statement x_i.
- $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$: on input statement $(k, x_1, \ldots, x_n)$, and a proof Π, output 1 to accept and 0 to reject.
- $(\Pi', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, (k, x_1, \ldots, x_n), x_{n+1}, \Pi)$: on input statements $(k, x_1, \ldots, x_n)$, x_{n+1}, and a proof Π for $(k, x_1, \ldots, x_n) \in \mathcal{L}^{tr}$, output an updated proof Π' for $(k, x_1, \ldots, x_n, x_{n+1}) \in \mathcal{L}^{tr}$, and additional auxiliary value aux_{n+1}. The auxiliary value aux_{n+1} is used later on to perform an add operation using a witness for x_{n+1}.
- $(\Pi', \mathsf{aux}'_\alpha) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, (k, x_1, \ldots, x_n), (w, \alpha), \mathsf{aux}, \Pi)$: on input statement $(k, x_1, \ldots, x_n)$, witness (w, α), auxiliary value aux, and proof Π for $(k, x_1, \ldots, x_n) \in \mathcal{L}^{tr}$, output an updated proof Π' for $(k+1, x_1, \ldots, x_n) \in \mathcal{L}^{tr}$, and updated auxiliary value aux'_α.
- $(\Pi', r = (r_1, \ldots, r_n)) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$: on input statement x and proof Π for $x \in \mathcal{L}^{tr}$, output a re-randomized proof Π' and update randomness r_i (related to auxiliary value aux_i) with $i \in [n]$.

- $\mathsf{aux}'_i \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{aux}_i, r_i)$: on input auxiliary value aux_i, and update randomness r_i, output updated auxiliary value aux'_i. $\mathsf{AuxUpdate}$ is used to update the auxiliary values after a proof has been re-randomized. The used input randomness is the one given in output by RandPr. To simplify the notation, we write $\mathsf{AUX}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}, r)$ to indicate that a list of auxiliary values is updated by appropriately parsing AUX and r and running the update operation on each element of the list.
- $0/1 \leftarrow \mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)), (\mathsf{aux}_1, \ldots, \mathsf{aux}_n), \mathit{\Pi})$: on input statement $(k, x_1, \ldots, x_n)$, witness $((w_1, \alpha_1) \ldots, (w_k, \alpha_k))$, auxiliary values $(\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$, and proof $\mathit{\Pi}$, output 1 if the auxiliary values are consistent with the statement, the proof, and the witness. Return 0 otherwise. If $\mathsf{AuxUpdate}$ returns 1, we are guaranteed that the subsequent extend/add operations can be correctly executed[8].

An extendable non-interactive proof system is said to be an extendable non-interactive witness indistinguishable (ENIWI) proof of knowledge if it satisfies adaptive extractable soundness (Def. 14 of [5]) and the following properties.

Definition 5 (Completeness). *An extendable non-interactive proof system for $R_{\mathcal{L}^{tr}}$ is complete if $\forall \lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $\mathsf{crs} \leftarrow\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in R_{\mathcal{L}^{tr}}$, and $(\mathit{\Pi}, \mathsf{AUX}) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ it holds that*

$$\Pr[\mathsf{PrVerify}(\mathsf{crs}, x, \mathit{\Pi}) = 1 \land \mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}, \mathit{\Pi}) = 1] = 1$$

Definition 6 (Transformation Completeness). *An extendable non-interactive proof system for $R_{\mathcal{L}^{tr}}$ is transformation complete if $\forall \lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $\mathsf{crs} \leftarrow\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in R_{\mathcal{L}^{tr}}$, and $(\mathit{\Pi}, \mathsf{AUX})$ such that $\mathsf{PrVerify}(\mathsf{crs}, x, \mathit{\Pi}) = 1$ and $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}, \mathit{\Pi}) = 1$ the following holds with probability 1:*

- $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}', \mathit{\Pi}') = 1$, *where* $(\mathit{\Pi}', r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \mathit{\Pi})$ *and* $\mathsf{AUX}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}, r)$.
- *Parse x as $(k, x_1, \ldots, x_n)$ and w as $((w_1, \alpha_1) \ldots, (w_k, \alpha_k))$.* $(\mathit{\Pi}', \mathsf{aux}') \leftarrow \mathsf{PrAdd}(\mathsf{crs}, x, (w', \alpha'), \mathsf{aux}, \mathit{\Pi})$, *modify* AUX *replacing* $\mathsf{aux}_{\alpha'}$ *with* aux'.
  If $\alpha' \notin \{\alpha_1, \ldots \alpha_k\}$ and $(x_{\alpha'}, w') \in R_{\mathcal{L}}$, then $\mathsf{PrVerify}(\mathsf{crs}, (k{+}1, x_1, \ldots, x_n), \mathit{\Pi}') = 1$, *and* $\mathsf{AuxVerify}(\mathsf{crs}, (k{+}1, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k), (w', \alpha')), \mathsf{AUX}, \mathit{\Pi}') = 1$.
- $(\mathit{\Pi}', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, x, x_{n+1}, \mathit{\Pi})$, *modify* AUX *adding auxiliary value* aux_{n+1}. *Then,* $\mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \ldots, x_{n+1}), \mathit{\Pi}') = 1$, *and* $\mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \ldots, x_{n+1}), w, \mathsf{AUX}, \mathit{\Pi}') = 1$.

Definition 7 (Re-Randomizable Addition). *Consider the following experiment:*

- $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$
- $\mathsf{crs} \leftarrow\!\$\ \mathsf{CRSSetup}(\mathsf{gk})$

[8] We introduce $\mathsf{AuxVerify}$ merely as an internal utility to simplify the description of our definitions.

– $(x, w, \varPi^*, \mathsf{AUX}^*) \leftarrow \mathcal{A}(\mathsf{crs})$
– *Parse* x *as* $(k, x_1, \dots, x_n)$ *and* w *as* $((w_1, \alpha_1) \dots, (w_k, \alpha_k))$
– *If* $(x, w) \notin R_{\mathcal{L}^{tr}}$ *or* $\mathsf{PrVerify}(\mathsf{crs}, (k-1, x_1, \dots, x_n), \varPi^*) = 0$ *or* $\mathsf{AuxVerify}(\mathsf{crs}, (k-1, x_1, \dots, x_n), ((w_1, \alpha_1) \dots, (w_{k-1}, \alpha_{k-1})), \mathsf{AUX}^*, \varPi^*) = 0$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\$ \{0, 1\}$ *and do the following:*
  - *If* $b = 0$, $(\varPi_0, \mathsf{AUX}_0) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$; $(\varPi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \varPi_0), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}_0, r)$
  - *If* $b = 1$, $(\varPi_1, \mathsf{aux}^*) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, x, (w_k, \alpha_k), \mathsf{AUX}^*, \varPi^*)$. *Replace in* AUX^* *the value* aux_{α_k} *with* aux^*. $(\varPi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \varPi_1), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}^*, r)$
– $b' \leftarrow \mathcal{A}(\varPi, \mathsf{AUX})$

We say that the proof system has re-randomizable addition if for every PPT $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$*, such that* $\Pr[b = b'] \le 1/2 + \nu(\lambda)$.

Definition 8 (Re-Randomizable Extension). *Consider the following experiment:*

– $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$
– $\mathsf{crs} \leftarrow\$ \mathsf{CRSSetup}(\mathsf{gk})$
– $(x, w, x_n, \varPi^*, \mathsf{AUX}^*) \leftarrow \mathcal{A}(\mathsf{crs})$
– *Parse* x *as* $(k, x_1, \dots, x_{n-1})$
– *If* $(x, w) \notin R_{\mathcal{L}^{tr}}$ *or* $\mathsf{PrVerify}(\mathsf{crs}, x, \varPi^*) = 0$ *or* $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}^*, \varPi^*) = 0$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\$ \{0, 1\}$ *and do the following:*
  - *If* $b = 0$ $(\varPi_0, \mathsf{AUX}_0) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \dots, x_n), w)$; $(\varPi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \dots, x_n), \varPi_0), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}_0, r)$
  - *If* $b = 1$ $(\varPi_1, \mathsf{aux}^*) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, x, x_n, \varPi^*)$. *Append the value* aux^* *to* AUX^*. $(\varPi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \dots, x_n), \varPi_1), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}^*, r)$
– $b' \leftarrow \mathcal{A}(\varPi, \mathsf{AUX})$

We say that the proof system has re-randomizable extension if, for every PPT $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$*, such that* $\Pr[b = b'] \le 1/2 + \nu(\lambda)$.

Definition 9 (Extended Witness Indistinguishability). *Consider the following experiment.*

– $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$
– $\mathsf{crs} \leftarrow\$ \mathsf{CRSSetup}(\mathsf{gk})$
– $(x, w^0, w^1) \leftarrow \mathcal{A}(\mathsf{crs})$
– *Parse* x *as* $(k, x_1, \dots, x_n)$, w^i *as* $((w^i_1, \alpha^i_1) \dots, (w^i_k, \alpha^i_k))$, *for* $i \in \{0, 1\}$
– *If* $(x, w^0) \notin R_{\mathcal{L}^{tr}}$ *or* $(x, w^1) \notin R_{\mathcal{L}^{tr}}$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\$ \{0, 1\}$ *and do the following:*
  - $(\varPi, (\mathsf{aux}_1, \dots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w_b)$.
  - *Set* $I_0 = \{\alpha^0_1, \dots, \alpha^0_k\}$, $I_1 = \{\alpha^1_1, \dots, \alpha^1_k\}$, $I = I_0 \cap I_1$, $S = ([n] \setminus (I_0 \cup I_1)) \cup I$, *and* $\mathsf{AUX} = \{\mathsf{aux}_i\}_{i \in S}$.
– $b' \leftarrow \mathcal{A}(\varPi, \mathsf{AUX})$

The proof system has extended witness indistinguishability (EWI) *if for every PPT* $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$*, such that* $\Pr[b = b'] \le 1/2 + \nu(\lambda)$.

6 Our Extendable Threshold Ring Signature

In Fig. 9, we show our ETRS from an ENIWI PoK ENIWI for a *hard* relation $R_{\mathcal{L}}$, and an IND-CPA public key encryption scheme PKE which is homomorphic w.r.t. $\mathsf{ENIWI.AuxUpdate}$. By *hard* relation we mean that a PPT $\mathcal{A}$ who is given $x \in \mathcal{L}$, has negligible probability of providing a witness w such $(x, w) \in R_{\mathcal{L}}$. We also require that $R_{\mathcal{L}}$ is public coin samplable, meaning that it is possible to efficiently sample random $x \in \mathcal{L}$. We omit the Setup algorithm from the description since it simply runs the setup algorithm of PKE and samples a hash function mapping arbitrary strings into elements in the correct space[9].

Instantiating our ETRS. We work over a bilinear group $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h})$ for which the SXDH assumption is believed to hold. In Sec. 7.3, we show an ENIWI PoK having as base relation pairing product equations in which all the variables are elements of $\check{\mathbb{H}}$, public constants are either paired with secret values or with $\check{h}$, and the target element is $0_{\mathbb{T}}$. In particular, we can use as base relation the following: $R_{\mathcal{L}} = \{(x = (\hat{a}, \hat{b}, \check{h}), w = \check{b}' | \hat{a} \cdot \check{h} + \hat{b} \cdot \check{b}' = 0_{\mathbb{T}}\}$. In Lem. 1, we prove that this is a hard relation under the DDH assumption in $\hat{\mathbb{G}}$. Additionally, since in our ENIWI $\mathsf{AuxUpdate}$ simply consists of applying the group operation between two elements of $\check{\mathbb{H}}$, we can use ElGamal instantiated in $\check{\mathbb{H}}$ as public key encryption scheme.

Remark on malicious extenders. As in [2], we do not consider security definitions accounting for malicious signers that try to prevent future signers from joining the signature. For example, in our construction a malicious extender could just encrypt a wrong auxiliary value. An approach that could be investigated to tackle this issue is adding a NIZK proving that the content of the encrypted auxiliary values is s.t. $\mathsf{AuxVerify} = 1$. Such NIZK would need to be malleable so that it could be updated after every re-randomization step, as well as whenever the signature is extended.

On combining signatures. One might wonder if concrete instantiations of our ETRS could also support the $\mathsf{Combine}$ operation as described in [2]. Whenever there is a shared public key (i.e., statement) in two ETRS, such signatures cannot be combined. Indeed, consider the case of two proofs over the same ring where there is a common base statement for which a corresponding witness was used in both proofs. Then, the combined proof should not have a resulting threshold that counts it twice. This means that the output of $\mathsf{Combine}$ would be different depending on whether two NIWI proofs on the same statement used the same witness or not. This is in clear contradiction with the witness indistinguishability property. On the other hand, the above observation does not exclude the possibility of having a weaker form of $\mathsf{Combine}$ where the starting signatures are constrained to have disjoint rings. Indeed, our instantiation of Sec. 7.3 could be

[9] We need a cryptographic hash function that allows to hash directly to both the source groups of the pairing group. See [21] for more details.

easily modified to support the corresponding Combine operation. Such operation exploits basically the same technique of the extend operation, and thus we omit its description.

6.1 Security of Our Extendable Threshold Ring Signature

Theorem 1. *Let* ENIWI *be an extendable non-interactive witness indistinguishable proof of knowledge for a hard relation* $R_{\mathcal{L}}$*, and* PKE *be an* IND-CPA *public key encryption scheme which is homomorphic w.r.t.* ENIWI.AuxUpdate*, then the scheme of Fig. 9 is an extendable threshold ring signature scheme.*

We prove Thm. 1 using Lem. 2 and Lem. 3.

Lemma 2. *The signature scheme described in Fig. 9 is unforgeable according to Def. 2.*

Proof sketch. The basic idea of the proof is to turn an adversary $\mathcal{A}$ breaking the unforgeability with non-negligible probability into another adversary $\mathcal{B}$ that extracts a witness for an instance $x \in \mathcal{L}$ of the hard relation, which is sampled by a challenger $\mathcal{C}$. In order to build this reduction, we need to show how to simulate all the oracle queries of $\mathcal{A}$ during the game. We do this by showing a series of hybrid games, starting from the game described in Fig. 5.

The first change consists into replying to Join queries by computing every time a new proof from scratch using ENIWI.Prove, instead of updating the current proof using PrAdd. This change is not detected by $\mathcal{A}$ thanks to the re-randomizable addition of the ENIWI.

The second change is that $\mathcal{B}$ can guess j^*, that is the index of the random oracle query in which $\mathcal{A}$ will query the message used in the forgery, and i^*, that is the index of a "new" signer used to create the forgery for m_{j^*}. We notice that, by the rules of the unforgeability game (see checks of lines $5-8$ of the unforgeability experiment in Fig. 5), this index i^* must exist, $\mathcal{A}$ never makes a corruption query for i^*, and it does not ask for any Sign/Join query involving i^* on message m_{j^*}. Whenever $\mathcal{B}$ discovers that it did not guess such indices correctly, $\mathcal{B}$ aborts. Nevertheless, since these indices can be kept perfectly hidden in $\mathcal{A}$'s view, $\mathcal{B}$ guesses these two indices with noticeable probability.

The next change consists into programming the random oracle to switch to an extraction-mode crs for the query on message m_{j^*}. Additionally, for each $j \neq j^*$, we can program the random oracle to output a $\mathsf{pk}_{\mathsf{O}_j}$ for which $\mathcal{B}$ knows the witness w_{1_j} s.t. $(x_{1_j}, w_{1_j}) \in R_{\mathcal{L}}$. Every Join/Sign query involving the signer i^* and a message m_j, with $j \neq j^*$, is answered using w_{1_j} instead of w_{i^*}. This change is not detectable by $\mathcal{A}$ thanks to the extended WI and the adaptive extractable soundness of ENIWI. Indeed, extended WI guarantees that $\mathcal{A}$ cannot notice the change of the used witness, and the adaptive extractable soundness guarantees that the probability of extracting a witness for statement x_{i^*} from the forgery does not change, except up to a negligible factor. Importantly, in order to reduce the indistinguishability of these changes to these two properties

```
$\mathsf{Sign}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk})$
  $\mathsf{A} \leftarrow \emptyset$
  $(\mathsf{crs}, \mathsf{pk}_\mathsf{O} = (x_1, \mathsf{pk}_\mathsf{e}^1)) \leftarrow \mathsf{H}(m)$
  Parse $\{\mathsf{pk}_i\}_{i\in\mathcal{R}} = (\mathsf{pk}_2, \ldots, \mathsf{pk}_{n+1})$
  Parse $\mathsf{pk}_i = (x_i, \mathsf{pk}_\mathsf{e}^i)$ for $i \in [n+1]$
  Parse $\mathsf{sk} = (w, \mathsf{sk}_\mathsf{e})$
  if $\nexists x_j, j \in [n+1]$ s.t. $(x_j, w) \in R_\mathcal{L}$
    return $\perp$
  Let $x = (1, x_1, \ldots, x_n, x_{n+1})$
  $(\Pi, \mathsf{AUX}) \leftarrow \mathsf{Prove}(x, (w, j))$
  for $i \in [n+1]$
    if $i = j \vee i = 1$
      $\mathsf{a} \leftarrow \mathsf{Enc}(\perp, \mathsf{pk}_\mathsf{e}^j)$
    else
      $\mathsf{a} \leftarrow \mathsf{Enc}(\mathsf{AUX}[i], \mathsf{pk}_\mathsf{e}^i)$
    $\mathsf{A} \leftarrow \mathsf{A} \cup \mathsf{a}$
  return $\sigma = (1, \Pi, \mathsf{A})$
```

```
$\mathsf{Extend}(m, \sigma, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{pk}^*)$
  if $\mathsf{pk}^* \in \{\mathsf{pk}_i\}_{i\in\mathcal{R}}$ return $\perp$
  $(\mathsf{pk}_\mathsf{O} = (x_1, \mathsf{pk}_\mathsf{e}^1)) \leftarrow \mathsf{H}(m)$
  Parse $\{\mathsf{pk}_i\}_{i\in\mathcal{R}} = (\mathsf{pk}_2, \ldots, \mathsf{pk}_{n+1})$
  Parse $\mathsf{pk}_i = (x_i, \mathsf{pk}_\mathsf{e}^i)$ for $i \in [n+1]$
  Parse $\mathsf{pk}^* = (x_{n+2}, \mathsf{pk}_\mathsf{e}^{n+2})$
  Parse $\sigma = (k, \Pi, \mathsf{A})$
  Let $x = (k, x_1, \ldots, x_{n+1})$
  $(\Pi, \mathsf{aux}) \leftarrow \mathsf{PrExtend}(x, x_{n+2}, \Pi)$
  $\mathsf{a} \leftarrow \mathsf{Enc}(\mathsf{aux}, \mathsf{pk}_\mathsf{e}^{n+2})$
  $\mathsf{A} \leftarrow \mathsf{A} \cup \mathsf{a}$
  Let $\bar{x} = (k, x_1, \ldots, x_{n+2})$
  $(\Pi, r_1, \ldots, r_{n+2}) \leftarrow \mathsf{RandPr}(\bar{x}, \Pi)$
  for $\mathsf{a}_i \in \mathsf{A}$
    $\mathsf{a}_i \leftarrow \mathsf{Eval}(\mathsf{a}_i, r_i, \mathsf{pk}_\mathsf{e}^i)$
  return $\sigma = (k, \Pi, \mathsf{A})$
```

```
$\mathsf{KeyGen}()$
  $(\mathsf{pk}_\mathsf{e}, \mathsf{sk}_\mathsf{e}) \leftarrow \mathsf{KeyGen}()$
  Sample $(x, w) \in R_\mathcal{L}$
  $(\mathsf{pk} = (x, \mathsf{pk}_\mathsf{e}), \mathsf{sk} = (w, \mathsf{sk}_\mathsf{e}))$
  return $(\mathsf{pk}, \mathsf{sk})$
```

```
$\mathsf{Join}(m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{sk}, \sigma)$
  $(\mathsf{pk}_\mathsf{O} = (x_1, \mathsf{pk}_\mathsf{e}^1)) \leftarrow \mathsf{H}(m)$
  Parse $\{\mathsf{pk}_i\}_{i\in\mathcal{R}} = (\mathsf{pk}_2, \ldots, \mathsf{pk}_{n+1})$
  Parse $\mathsf{pk}_i = (x_i, \mathsf{pk}_\mathsf{e}^i)$ for $i \in [n+1]$
  Parse $\mathsf{sk} = (w, \mathsf{sk}_\mathsf{e})$
  if $\nexists x_j, j \in [n+1]$ s.t $(x_j, w) \in R_\mathcal{L}$
    return $\perp$
  Parse $\sigma = (k, \Pi, \mathsf{A}), \mathsf{A} = (\mathsf{a}_1, \ldots, \mathsf{a}_{n+1})$
  Parse $\mathsf{sk} = (w, \mathsf{sk}_\mathsf{e})$
  $\mathsf{aux} \leftarrow \mathsf{Dec}(\mathsf{a}_j, \mathsf{sk}_\mathsf{e})$
  Let $x = (k, x_1, \ldots, x_{n+1})$
  $(\Pi, \mathsf{aux}'_j) \leftarrow \mathsf{PrAdd}(x, (w, j), \mathsf{aux}, \Pi)$
  Set $\mathsf{a}_j \in \mathsf{A}$ as $\mathsf{a}_j \leftarrow \mathsf{Enc}(\perp, \mathsf{pk}_\mathsf{e}^j)$
  $k \leftarrow k + 1$
  $(\Pi, r_1, \ldots, r_{n+1}) \leftarrow \mathsf{RandPr}(x, \Pi)$
  for $\mathsf{a}_i \in \mathsf{A}$
    $\mathsf{a}_i \leftarrow \mathsf{Eval}(\mathsf{a}_i, r_i, \mathsf{pk}_\mathsf{e}^i)$
  return $\sigma = (k, \Pi, \mathsf{A})$
```

```
$\mathsf{Verify}(t, m, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \sigma)$
  $(\mathsf{crs}, \mathsf{pk}_\mathsf{O} = (x_1, \mathsf{pk}_\mathsf{e}^1)) \leftarrow \mathsf{H}(m)$
  Parse $\{\mathsf{pk}_i\}_{i\in\mathcal{R}} = (\mathsf{pk}_2, \ldots, \mathsf{pk}_{n+1})$
  Parse $\mathsf{pk}_i = (x_i, \mathsf{pk}_\mathsf{e}^i)$ for $i \in [n+1]$
  Parse $\sigma = (k, \Pi, \mathsf{A})$
  Let $x = (k, x_1, \ldots, x_{n+1})$
  if $k < t$
    return $0$
  else
    return $\mathsf{PrVerify}(x, \Pi)$
```

Fig. 9. ETRS from ENIWI PoK and IND-CPA homomorphic PKE. For space reasons, we directly write the internal algorithms of the schemes (e.g., Prove instead of ENIWI.Prove), and we omit crs from ENIWI algorithms input considering it as implicit. We use AUX[i] to indicate the i-th element of list AUX.

of the ENIWI we take advantage of the fact that we have a *different* CRS for every message. Finally, after applying all these changes, $\mathcal{B}$ can set x_{i^*} as the x received from $\mathcal{C}$. Given the forgery generated by $\mathcal{A}$, $\mathcal{B}$ can extract a witness for statement x, breaking the hardness of $R_{\mathcal{L}}$.

Let $\Pr\Big[\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\Big]$ be the probability that the adversary wins the unforgeability game, we have that: $\Pr\Big[\mathsf{Exp}^{\mathrm{cmEUF}}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\Big] \leq \epsilon_{rr} + q_m(\epsilon_{crs} + (q_{KG} + 1)(\epsilon_{HR} + \epsilon_{EWI}))$, where q_{KG} and q_m are polynomial bounds on the number of key generation queries and random oracle queries that $\mathcal{A}$ can do. While ϵ_{rr}, ϵ_{crs}, ϵ_{HR}, ϵ_{EWI} are the advantages in the re-randomizable addition game of ENIWI, in distinguishing a regular CRS from an extraction-mode CRS, in the hard relation game, and in the extended witness indistinguishability game respectively. We defer to [5] for the complete proof.

Lemma 3. *The signature scheme described in Fig. 9 satisfies the anonymity property of Def. 3.*

Proof sketch. Through a sequence of indistinguishable hybrids, we switch from a challenger $\mathcal{B}$ using lad_0 to a $\mathcal{B}$ using lad_1. We show that at every hybrid, $\mathcal{B}$ can exploit $\mathcal{A}$ distinguishing between the two hybrids to break some properties of the underlying primitives. First, $\mathcal{B}$ changes the way it processes the ladders and replies to Join queries. In particular, $\mathcal{B}$ computes every time a new proof from scratch using $\mathsf{ENIWI.Prove}$, instead of running the $\mathsf{Join}/\mathsf{Extend}$ algorithms, analogously to the proof of unforgeability. After that, when processing the ladders, $\mathcal{B}$ will encrypt $\perp$ in all signers' ciphertexts. This change is not detected by $\mathcal{A}$ thanks to the IND-CPA property of the encryption scheme. At the end, $\mathcal{B}$ fixes the ladder used in the anonymity game to be lad_1. This change is unnoticeable thanks to the extended WI of ENIWI. We defer to [5] for the complete proof.

Lemma 4. *The signature scheme described in Fig. 9 enjoys fellow-signer anonymity (cfr., Def. 4).*

The proof follows essentially the same path of the one of Lem. 3.

7 Our Extendable Non-Interactive Witness Indistinguishable Proof of Knowledge

In this section, we first show how to extend the GS proof system to define a proof system for a threshold relation. After that, we show how to further modify such scheme to get our ENIWI PoK.

7.1 GS Proofs of Partial Satisfiability

In [23,13], it is shown how to transform n sets of certain types of equations $S_1, \ldots, S_n$ to a set of equations S' s.t. S' is satisfied whenever one of $S_1, \ldots, S_n$

is satisfied. A witness for S_i, with $i \in [n]$, is easily mapped to a witness for S'. Indeed, this transformation realizes a disjunction. The transformation works by assuming that $S_1, \ldots, S_n$ have independent variables, adding variables $b_1, \ldots b_{n-1} \in \{0,1\}$, and defining $b_n = 1 - b_1 - \ldots - b_{n-1}$. Then, for $i \in [n]$, b_i is used to modify all the equations in S_i so that they remain the same if $b_i = 1$, but they admit the trivial solution for $b_i = 0$. Slightly increasing the overhead of these compilers, it is also possible to implement partial satisfiability proofs for an arbitrary threshold k, meaning that S' is satisfied iff k of $S_1, \ldots, S_n$ are satisfied. To do so, the main idea is to define $b_n \in \{0,1\}$, and to prove that $b_1 + \ldots + b_n = k$.

A case which is relevant to this paper is when $S_1, \ldots, S_n$ contain only PPEs with $t_{\mathbb{T}} = 0_{\mathbb{T}}$, all the variables of the PPEs are elements of $\check{\mathbb{H}}$, and public constants are either paired with secret values or with $\check{h}$. In this case, the prover would:

1. Add variables $b_1, \ldots, b_n$ and prove that $b_i \in \{0,1\}$ $\forall i \in [n]$. This can be done with quadratic equations, by adding the equations $b_i(1-b_i) = 0$. Let us define such equations to be of type $\mathcal{B}$, we will refer to a specific equation using $\mathcal{B}_i$.
2. Add variables $\hat{M}_1, \ldots, \hat{M}_n$ and prove $b_i\hat{g} - \hat{M}_i = 0$, with $i \in [n]$. This can be done via multi-scalar multiplication equations in $\hat{\mathbb{G}}$. Since $b_i \in \{0,1\}$, it follows that $\hat{M}_i \in \{\hat{0}, \hat{g}\}$. Let us define such equations to be of type $\mathcal{M}$.
3. Add equation $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$. Since $\hat{M}_i \in \{\hat{0}, \hat{g}\}$, this equation implies that exactly k of the $\hat{M}_i$, with $i \in [n]$, are equal to $\hat{g}$. Let us call such equation as $\mathcal{K}$.
4. For each S_i, with $i \in [n]$, let Q_i be the number of equations in S_i, let $J_{i,q}$ be the number of variables in the equation $q \in [Q_i]$ of S_i. For each variable $\check{y}_{i,q,j}$ with $q \in [Q_i], j \in [J_{i,q}]$, define variable $\check{x}_{i,q,j}$ and add equation $\hat{M}_i \cdot \check{y}_{i,q,j} - \hat{M}_i \cdot \check{x}_{i,q,j} = 0_{\mathbb{T}}$. Since k of the $\hat{M}_i$ are equal to $\hat{g}$, this implies that for k equations sets it must hold that all $\check{y}_{i,q,j} = \check{x}_{i,q,j}$. Let us define such equations to be of type $\mathcal{Y}$.
5. For each equation in each S_i, replace all the original $\check{y}_{i,q,j}$ with the corresponding $\check{x}_{i,q,j}$. This allows to set all $\check{x}_{i,q,j} = \check{y}_{i,q,j} = \check{0}$ for each set S_i for which the prover does not have a satisfying assignment. For the k sets for which the prover does have a satisfying assignment, the prover sets $\check{y}_{i,q,j} = \check{x}_{i,q,j}$. Let us define such equations to be of type $\mathcal{X}$.

7.2 High-level Overview of our ENIWI.

We construct our ENIWI by observing that GS proofs of partial satisfiability can be updated in two ways:

– **Extend**: consider a proof Π for a set of equations S which is satisfied if k out of n of the original equations sets $S_1, \ldots, S_n$ are satisfied. On input a new equations set S_{n+1} and the proof Π, compute a new equations set S' which is satisfied if k out of the $n+1$ equations sets $S_1, \ldots, S_n, S_{n+1}$ are satisfied. Output S' and the corresponding updated proof Π'.

– **Add**: consider a proof Π for a set of equations S which is satisfied if k out n of the original equations sets $S_1, \ldots, S_n$ are satisfied. On input the proof Π for S, a witness for an equations set S_i with $i \in [n]$ which was not previously used to create Π, and some corresponding auxiliary information aux_i, compute a new equations set S' which is satisfied if $k+1$ out of the n equations sets $S_1, \ldots, S_n$ are satisfied. Output S' and the corresponding updated proof Π'.

In particular, one can notice that each step of the partial satisfiability proof described in Sec. 7.1 only adds equations featuring independent variables, except for step 3. In step 3, one equation is added combining all variables $\hat{M}_i$ with $i \in [n]$. The equation is $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$. Let us compute the GS proof for such equation. Let crs be $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$.

– Variables $\hat{M}_i$ are committed as group elements (i.e., with label $\mathsf{com}_{\hat{\mathbb{G}}}$), thus $\hat{c}_{\hat{M}_i} = \boldsymbol{e}^\top \hat{M}_i + \hat{\boldsymbol{v}} r_i + \hat{\boldsymbol{w}} s_i$, with $r_i, s_i \leftarrow\!\!\$\ \mathbb{Z}_p$.
– $\hat{g}$ is the base element of $\hat{\mathbb{G}}$, thus it is publicly committed with label $\mathsf{base}_{\hat{\mathbb{G}}}$ as $\hat{c}_{\hat{g}} = (0, \hat{g})^\top$.
– $\check{h}$ is the base element of $\check{\mathbb{H}}$, and thus it is publicly committed with label $\mathsf{base}_{\check{\mathbb{H}}}$ as $(0, \check{h})$.

This results in $\hat{C} = (\hat{c}_{\hat{M}_1}, \ldots, \hat{c}_{\hat{M}_n}, \hat{c}_{\hat{g}}), \check{D} = (0, \check{h}), \boldsymbol{r}_x = (r_1, \ldots, r_n, 0)^\top, \boldsymbol{s}_x = (s_1, \ldots, s_n, 0)^\top, \boldsymbol{r}_y = 0, \boldsymbol{s}_y = 0$.

This means that $\check{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} = -\hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$ and $\check{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} = -\hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$, with $\alpha, \gamma, \beta, \delta$ being random elements in $\mathbb{Z}_p$.

Let us compute $\boldsymbol{r}_x \Gamma \check{D} = (r_1, \ldots, r_n, 0)^\top (1, \ldots, 1, -k)(0, \check{h}) = (0, \sum_{i=1}^{n} r_i \check{h})$. Similarly, we have that $\boldsymbol{s}_x \Gamma \check{D} = (0, \sum_{i=1}^{n} s_i \check{h})$. Let us define $\mathsf{aux}_i = (\mathsf{aux}_i^1, \mathsf{aux}_i^2) = (r_i \check{h}, s_i \check{h})$. This means that $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x \Gamma \check{D} + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}} = (0, \sum_{i=1}^{n} \mathsf{aux}_i^1) + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x \Gamma \check{D} + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}} = (0, \sum_{i=1}^{n} \mathsf{aux}_i^2) + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}}$.

We notice that the proof elements for equation $\mathcal{K}$ are essentially a sum of n independent contributions (i.e., the aux_i values) for each of the involved n variables (i.e., $\hat{M}_i$ with $i \in [n]$). We can exploit this fact to perform the extend and add operations in the following way. Let us consider the steps of Sec. 7.1.

– **Extend**: Add new equations of types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$ by defining the corresponding new independent variables, and compute the related GS proofs. Modify equation $\mathcal{K}$ to be $\sum_{i=1}^{n+1} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$ and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + (0, r_{n+1}\check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + (0, s_{n+1}\check{h})$, where r_{n+1} and s_{n+1} are the randomnesses used to commit to the new variable $\hat{M}_{n+1} = \hat{0}$.
– **Add**: Replace the committed variables for the equations $\mathcal{B}_i, \mathcal{M}_i, \mathcal{Y}_i, \mathcal{X}_i$ with new committed variables $b_i = 1$, $\hat{M}_i = \hat{g}$, and $\check{y}_{i,q,j} = \check{x}_{i,q,j}$. Replace the old corresponding GS proofs with freshly computed ones. Modify equation $\mathcal{K}$ to be $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - (k+1)\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$, and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} - (0, \mathsf{aux}_i^1) + (0, r_i'\check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} - (0, \mathsf{aux}_i^2) + (0, s_i'\check{h})$, where r_i' and s_i' are the randomnesses used for the fresh commitment to $\hat{M}_i = \hat{g}$.

After any of the two above modifications, the resulting proof is an accepting proof for the updated threshold relation. Indeed, both the extend and add operation symbolically compute the proofs in the same way a prover for the updated threshold relation would do from scratch.

7.3 Our ENIWI

Our ENIWI is an ENIWI PoK over the language of sets of pairing product equations where all the variables are elements of $\check{\mathbb{H}}$, public constants are either paired with secret values or with $\check{h}$, and the target element is $0_{\mathbb{T}}$. For simplicity, we consider each statement x_i as containing only one equation.

- $\mathsf{crs} \leftarrow \mathsf{CRSSetup}(\mathsf{gk})$: run $\mathsf{GS.Setup}(\mathsf{gk})$. This results in $\mathsf{crs} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$.
- $(\Pi, (\mathsf{aux}_1, \dots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \dots, x_n), ((w_1, \alpha_1) \dots, (w_k, \alpha_k)))$: on input $((k, x_1, \dots, x_n), ((w_1, \alpha_1) \dots, (w_k, \alpha_k))) \in \mathsf{rl}$, define $A = \{\alpha_1, \dots, \alpha_k\}$[10] and do the following.
  1. For each equation x_i, $i \in [n]$, define new variables and equations:
     - Define variable $b_i = 1$ if $i \in A$, and $b_i = 0$ otherwise.
     - Define quadratic equation $\mathcal{B}_i$ as $b_i(1 - b_i) = 0$.
     - Define variables $\hat{M}_i = \hat{g}$ if $i \in A$, and $\hat{M}_i = \hat{0}$ otherwise.
     - Define multi-scalar multiplication equation $\mathcal{M}_i$ as $b_i\hat{g} - \hat{M}_i = 0$.
     - Let J_i be the number of variables in equation x_i. For each variable $\check{y}_{i,j}$, with $j \in [J_i]$, define a variable $\check{x}_{i,j}$. Set $\check{x}_{i,j} = \check{y}_{i,j}$, if $i \in A$, and $\check{x}_{i,j} = \check{0}$ otherwise.
     - For each variable $\check{y}_{i,j}$, with $j \in [J_i]$, define pairing product equation $\mathcal{Y}_{i,j}$ as $\hat{M}_i \cdot \check{y}_{i,j} - \hat{M}_i \cdot \check{x}_{i,j} = 0_{\mathbb{T}}$.
     - Modify pairing product equation x_i by replacing each variable $\check{y}_{i,j}$, with $j \in [J_i]$, with variable $\check{x}_{i,j}$. Let us call such modified equation $\mathcal{X}_i$.

     Moreover, define pairing product equation $\mathcal{K}$ as $\sum_{i=1}^{n} \hat{M}_i \cdot \check{h} - k\hat{g} \cdot \check{h} = 0_{\mathbb{T}}$. At the end of this step, there will be n equations of types $\mathcal{B}, \mathcal{M}, \mathcal{X}$, $n \sum_{i=1}^{n} J_i$ equations of type $\mathcal{Y}$, and one equation of type $\mathcal{K}$.
  2. For each equation of types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$ generate appropriate commitments (using $\mathsf{GS.Com}$) to all variables, resulting in lists of commitments $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}$ respectively[11]. Then, for each equation of types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$, run $\mathsf{GS.Prove}$ with the obvious inputs obtaining proof elements lists $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}$. For example, $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}$ contains proof elements $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i}$, with $i \in [n]$, each of them obtained running $\mathsf{GS.Prove}$ for equation $\mathcal{B}_i$ using commitments $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}$ (and related randomnesses) from $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}$.

     Moreover, for equation $\mathcal{K}$ do the following[12]:

[10] A indicates what are the k equations the prover has a satisfying assignment for.

[11] Whenever different equations share the same variables, we can think of the commitments lists as containing copies of the exact same commitments. Clearly, in practice data does not need to be replicated.

[12] We report the whitebox computation of the GS prover to show how to compute the auxiliary values. Furthermore, for sake of clarity, we report again commitments to variables $\hat{M}_i$ with $i \in [n]$, which were already created to prove other equations.

- Commit to $\hat{M}_i$, with $i \in [n]$, with label $\mathsf{com}_{\hat{\mathbb{G}}}$ and randomness (r_i, s_i), i.e., $(\mathsf{com}_{\hat{\mathbb{G}}}, \hat{c}_{\hat{M}_i}) \leftarrow \mathsf{GS.Com}(\mathsf{com}_{\hat{\mathbb{G}}}, \hat{M}_i; (r_i, s_i))$, resulting in $\hat{c}_{\hat{M}_i} = \boldsymbol{e}^\top \hat{M}_i + \hat{\boldsymbol{v}} r_i + \hat{\boldsymbol{w}} s_i$.
- Commit to $\hat{g}$ with label $\mathsf{base}_{\hat{\mathbb{G}}}$ and randomness $(0, 0)$, i.e., $(\mathsf{base}_{\hat{\mathbb{G}}}, \hat{c}_{\hat{g}}) \leftarrow \mathsf{GS.Com}(\mathsf{base}_{\hat{\mathbb{G}}}, \hat{g}; (0, 0))$, resulting in $\hat{c}_{\hat{g}} = (0, \hat{g})^\top$.
- Commit to $\check{h}$ with label $\mathsf{base}_{\check{\mathbb{H}}}$ and randomness $(0, 0)$, i.e., $(\mathsf{base}_{\check{\mathbb{H}}}, \check{d}_{\check{h}}) \leftarrow \mathsf{GS.Com}(\mathsf{base}_{\check{\mathbb{H}}}, \check{h}; (0, 0))$, resulting in $\check{d}_{\check{h}} = (0, \check{h})$.

Do the following steps:

- Define $\hat{C} = (\hat{c}_{\hat{M}_1}, \ldots, \hat{c}_{\hat{M}_n}, \hat{c}_{\hat{g}})$, $\check{D} = (0, \check{h})$, $\boldsymbol{r}_x = (r_1, \ldots, r_n, 0)^\top$, $\boldsymbol{s}_x = (s_1, \ldots, s_n, 0)^\top$, $\boldsymbol{r}_y = 0$, $\boldsymbol{s}_y = 0$. This means that $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} = -\hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$ and $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} = -\hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$.
- Compute $\boldsymbol{r}_x \Gamma \check{D} = (r_1, \ldots, r_n, 0)^\top (1, \ldots, 1, -k)(0, \check{h}) = (0, \sum_{i=1}^n r_i \check{h})$. Similarly, we have that $\boldsymbol{s}_x \Gamma \check{D} = (0, \sum_{i=1}^n s_i \check{h})$. Define $\mathsf{aux}_i = (\mathsf{aux}_i^1, \mathsf{aux}_i^2) = (r_i \check{h}, s_i \check{h})$, with $i \in [n]$.
- Compute $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x \Gamma \check{D} + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}} = (0, \sum_{i=1}^n \mathsf{aux}_i^1) + \alpha \check{\boldsymbol{v}} + \beta \check{\boldsymbol{w}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x \Gamma \check{D} + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}} = (0, \sum_{i=1}^n \mathsf{aux}_i^2) + \delta \check{\boldsymbol{v}} + \gamma \check{\boldsymbol{w}}$.

Let $\pi_{\mathcal{K}} = (\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}})$ and $C_{\mathcal{K}} = (\hat{C}, \check{D})$. Output $(\Pi = (\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}}), \mathsf{AUX} = (\mathsf{aux}_1, \ldots, \mathsf{aux}_n))$.

– $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$ reconstruct equations of type $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}, \mathcal{K}$, appropriately parse Π, and for every equation run $\mathsf{GS.PrVerify}$ with the obvious inputs. For example, the proof for equation $\mathcal{B}_i$ is verified giving, after appropriate parsing, commitments $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}$ and proof element $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i}$ in input to $\mathsf{GS.PrVerify}$. Return 1 iff all the calls to $\mathsf{GS.PrVerify}$ return 1.

– $(\Pi', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, (k, x_1, \ldots, x_n), x_{n+1}, \Pi)$ do the following:

1. Parse Π as $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}}), \mathsf{AUX} = (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)$.
2. For each of the 4 equation types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$, add a new equation related to x_{n+1} by defining the corresponding new independent variables, $b_{n+1} = 0, \hat{M}_{n+1} = \hat{0}$ and all the $\check{y}_{n+1,j} = \check{0}$, with $j \in [J_{n+1}]$.
3. Compute commitments to new variables and appropriately add them to $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}$.
4. Compute the related new GS proofs and add them to $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}$ accordingly.
5. Parse $\pi_{\mathcal{K}}$ as $(\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}})$ and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + (0, r_{n+1}\check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + (0, s_{n+1}\check{h})$, where r_{n+1} and s_{n+1} are the randomnesses used to commit to the new variable $\hat{M}_{n+1} = \hat{0}$.
6. Set $\mathsf{aux}_{n+1} = (\mathsf{aux}_{n+1}^1, \mathsf{aux}_{n+1}^2) = (r_{n+1}\check{h}, s_{n+1}\check{h})$.
7. Output $(\Pi, \mathsf{aux}_{n+1})$.

– $(\Pi', \mathsf{aux}'_\alpha) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, (k, x_1, \ldots, x_n), (w, \alpha), \mathsf{aux}, \Pi)$ do the following:

1. Parse Π as $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}})$.
2. For each of the 4 equation types $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}$, replace the variables in equations related to x_α (i.e, $\mathcal{B}_\alpha, \mathcal{M}_\alpha, \mathcal{X}_\alpha$, and all $\mathcal{Y}_{\alpha,j}$ with $j \in J_\alpha$) as follows: $b_\alpha = 1, \hat{M}_\alpha = \hat{g}$ and all the $\check{y}_{\alpha,j} = \check{x}_{\alpha,j}$, with $j \in [J_\alpha]$.
3. Replace the commitments related to equations $\mathcal{B}_\alpha, \mathcal{M}_\alpha, \mathcal{X}_\alpha$, and all $\mathcal{Y}_{\alpha,j}$, with $j \in J_\alpha$ with freshly generated ones updating $\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}$ accordingly.

4. Replace the GS proofs related to equations $\mathcal{B}_\alpha, \mathcal{M}_\alpha, \mathcal{X}_\alpha$, and all $\mathcal{Y}_{\alpha,j}$ with $j \in J_\alpha$, with freshly generated ones replacing proof elements of $\boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}$ accordingly.
5. Parse $\pi_{\mathcal{K}}$ as $(\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}})$ and update $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}$ and $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}$ as $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} - (0, \mathsf{aux}^1_\alpha) + (0, r'_\alpha \check{h}), \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} - (0, \mathsf{aux}^2_\alpha) + (0, s'_\alpha \check{h})$, where r'_α and s'_α are the randomnesses used for the fresh commitment to $\hat{M}_\alpha = \hat{g}$.
6. Set $\mathsf{aux}'_\alpha = (\mathsf{aux}^1_\alpha, \mathsf{aux}^2_\alpha) = (r'_\alpha \check{h}, s'_\alpha \check{h})$.
7. Output $(\Pi, \mathsf{aux}'_\alpha)$.

– $(\Pi', r_1, \ldots, r_n) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi)$:
  1. Run $\mathsf{GS.RandPr}$ on each of the proofs, appropriately fixing the random coins when randomizing proofs related to equations involving shared variables (i.e., s.t. we end up again with shared variables having the exact same commitments). Let r'_i, s'_i, with $i \in [n]$ be the randomnesses used to update commitments to all $\hat{M}_i$, with $i \in [n]$. Define $r_i = (r'_i, s'_i)$. Let randomized proof elements and commitments be contained in Π'.
  2. Output $(\Pi', r_1, \ldots, r_n)$

– $\mathsf{aux}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{aux}, r)$:
  1. Parse r as (r', s'), and aux as $(\mathsf{aux}^1, \mathsf{aux}^2)$.
  2. Output $\mathsf{aux}' = (\mathsf{aux}^1 + r'\check{h}, \mathsf{aux}^2 + s'\check{h})$.

– $0/1 \leftarrow \mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)), (\mathsf{aux}_1, \ldots, \mathsf{aux}_n), \Pi)$:
  1. Parse Π as $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}})$. Parse $C_{\mathcal{K}}$ as $\hat{C} = (\hat{c}_{\hat{M}_1}, \ldots, \hat{c}_{\hat{M}_n}, \hat{c}_{\hat{g}})$ and $\check{D} = (0, \check{h})$.
  2. Check that $(\mathsf{aux}_{\alpha_1}, \ldots, \mathsf{aux}_{\alpha_k})$ all open $(\hat{c}_{\hat{M}_{\alpha_1}}, \ldots, \hat{c}_{\hat{M}_{\alpha_k}})$ to $\hat{g}$. Namely, check that $\hat{c}_{\hat{M}_i} \cdot (\check{h}, \check{h}) + \hat{\boldsymbol{v}} \cdot (-\mathsf{aux}^1_i, -\mathsf{aux}^1_i) + \hat{\boldsymbol{w}} \cdot (-\mathsf{aux}^2_i, -\mathsf{aux}^2_i) = (\hat{0}, \hat{g})^\top \cdot (\check{h}, \check{h})$, for all $i \in A$.
  3. Check that remaining auxiliary values open commitments $\hat{c}_{\hat{M}_i}$ with $i \in [n] \setminus A$ to $\hat{0}$. Namely, check that $\hat{c}_{\hat{M}_i} \cdot (\check{h}, \check{h}) + \hat{\boldsymbol{v}} \cdot (-\mathsf{aux}^1_i, -\mathsf{aux}^1_i) + \hat{\boldsymbol{w}} \cdot (-\mathsf{aux}^2_i, -\mathsf{aux}^2_i) = (\hat{0}, \hat{0})^\top \cdot (\check{h}, \check{h})$, for all $i \in [n] \setminus A$.

Theorem 2. *If* GS *(cfr., Sec. 3.1) is a NIWI for all equation types and a NIWI PoK for pairing product equations, then the construction above is an* ENIWI *PoK. The base relation $R_{\mathcal{L}}$ consists of pairing product equations in which all the variables are elements of $\check{\mathbb{H}}$, public constants are either paired with secret values or with $\check{h}$, and the target element is $0_{\mathbb{T}}$.*

See [5] for the proof.

Acknowledgements

This result is part of projects that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under projects PICOCRYPT (grant agreement No. 101001283) and PROCONTRA (grant agreement No. 885666), from the Spanish Government under project PRODIGY (TED2021-132464B-I00), and from the Madrid Regional Government under project BLOQUES (S2018/TCS-4339). The last two projects are co-funded by European Union EIE, and NextGenerationEU/PRTR funds.

References

1. Aguilar Melchor, C., Cayrel, P.L., Gaborit, P.: A new efficient threshold ring signature scheme based on coding theory. In: PQCrypto 2008. pp. 1–16. Springer.
2. Aranha, D.F., Hall-Andersen, M., Nitulescu, A., Pagnin, E., Yakoubov, S.: Count Me In! Extendability for Threshold Ring Signatures. In: PKC 2022, Part II. LNCS, vol. 13178, pp. 379–406. Springer.
3. Attema, T., Cramer, R., Fehr, S.: Compressing proofs of k-out-of-n partial knowledge. In: CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 65–91. Springer.
4. Attema, T., Cramer, R., Rambaud, M.: Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In: ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 526–556. Springer.
5. Avitabile, G., Botta, V., Fiore, D.: Extendable Threshold Ring Signatures with Enhanced Anonymity. ePrint, Report 2022/1568.
6. Avitabile, G., Botta, V., Friolo, D., Visconti, I.: Efficient proofs of knowledge for threshold relations. In: ESORICS 2022 , Part III. LNCS, vol. 13556, pp. 42–62. Springer.
7. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer.
8. Bettaieb, S., Schrek, J.: Improved lattice-based threshold ring signature scheme. In: PQCrypto 2013. pp. 34–51. Springer.
9. Beullens, W., Katsumata, S., Pintore, F.: Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 464–492. Springer.
10. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer.
11. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: ESORICS 2015, Part I. LNCS, vol. 9326, pp. 243–265. Springer.
12. Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: CRYPTO 2002. LNCS, vol. 2442, pp. 465–480. Springer.
13. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer.
14. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 281–300. Springer.
15. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer.
16. Chow, S.S.M., Wei, V.K.W., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: ASIACCS 06. pp. 297–302. ACM Press.
17. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer.

18. Escala, A., Groth, J.: Fine-tuning Groth-Sahai proofs. In: PKC 2014. LNCS, vol. 8383, pp. 630–649. Springer.
19. 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: ACNS 19. LNCS, vol. 11464, pp. 67–88. Springer.
20. Faonio, A., Fiore, D., Nizzardo, L., Soriente, C.: Subversion-Resilient Enhanced Privacy ID. In: CT-RSA 2022. LNCS, vol. 13161, pp. 562–588. Springer.
21. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics **156**(16), 3113–3121.
22. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking sigmas: A framework to compose Σ-protocols for disjunctions. In: EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 458–487. Springer.
23. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer.
24. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer.
25. Haque, A., Krenn, S., Slamanig, D., Striecks, C.: Logarithmic-size (linkable) threshold ring signatures in the plain model. In: PKC 2022, Part II. pp. 437–467. Springer.
26. Haque, A., Scafuro, A.: Threshold ring signatures: New definitions and post-quantum security. In: PKC 2020, Part II. LNCS, vol. 12111, pp. 423–452. Springer.
27. Liu, Z., Nguyen, K., Yang, G., Wang, H., Wong, D.S.: A lattice-based linkable ring signature supporting stealth addresses. In: ESORICS 2019, Part I. LNCS, vol. 11735, pp. 726–746. Springer.
28. Lu, X., Au, M.H., Zhang, Z.: Raptor: A practical lattice-based (linkable) ring signature. In: ACNS 19. LNCS, vol. 11464, pp. 110–130. Springer.
29. Munch-Hansen, A., Orlandi, C., Yakoubov, S.: Stronger notions and a more efficient construction of threshold ring signatures. In: LATINCRYPT 2021. LNCS, vol. 12912, pp. 363–381. Springer.
30. Naor, M.: Deniable ring authentication. In: CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer.
31. Okamoto, T., Tso, R., Yamaguchi, M., Okamoto, E.: A k-out-of-n ring signature with flexible participation for signers. ePrint, Report 2018/728.
32. Petzoldt, A., Bulygin, S., Buchmann, J.: A multivariate based threshold ring signature scheme. Appl. Algebra Eng. Commun. Comput. **24**(3-4), 255–275.
33. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer.
34. Russo, A., Anta, A.F., Vasco, M.I.G., Romano, S.P.: Chirotonia: A Scalable and Secure e-Voting Framework based on Blockchains and Linkable Ring Signatures. In: 2021 IEEE International Conference on Blockchain (Blockchain). pp. 417–424.
35. Thyagarajan, S.A.K., Malavolta, G., Schmid, F., Schröder, D.: Verifiable timed linkable ring signatures for scalable payments for monero. In: ESORICS 2022 , Part II. LNCS, vol. 13555, pp. 467–486. Springer.
36. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient Linkable and/or Threshold Ring Signature Without Random Oracles. Comput. J. **56**(4), 407–421.
37. Zhang, F., Kim, K.: ID-based blind signature and ring signature from pairings. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer.