Extendable Threshold Ring Signatures with Enhanced Anonymity

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

[1] Università degli Studi di Salerno, Fisciano, Italy {gavitabile,vbotta}@unisa.it
[2] IMDEA Software Institute, Madrid, Spain. dario.fiore@imdea.org

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 while visiting IMDEA Software Institute.

1 Introduction

Anonymity is a central requirement in several privacy-preserving technologies. Notable examples are e-voting protocols [33], anonymous authentication [29], and privacy-protecting cryptocurrencies [34]. A central cryptographic primitive that can be used to provide anonymity in applications is ring signatures [32]. Ring signatures [32] 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 [16], pairing groups [9,15,36], non-interactive zero-knowledge proofs [10,3,21], and lattices [26,27,18,8]. 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 [11] 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 [30,28]. 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 [28], 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 Suppl. A.1 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 Suppl. A.1, 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]	$\mathcal{O}(tn)$	$\mathcal{O}(tn)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(tn)$	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]. We use n to indicate the size of the ring and t to indicate 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], while we say that a scheme achieves strong anonymity if our stronger anonymity definition is satisfied. FSA stands for fellow-signer anonymity.

ENIWI *PoKs.* In [13], 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. A transformation is allowable w.r.t. R if on

[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.

input $(x, w) \in R$ it gives as output $(T_x(x) = x', T_w(w) = w') \in R$. Then, a proof 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 [23] 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_t is defined w.r.t. a relation R as $R_t = \{(x = (k, x_1, \dots, x_n), w = ((w_1, \alpha_1), \dots, (w_k, \alpha_k)))| 1 \leq \alpha_1 < \dots < \alpha_k \leq n \wedge \ \forall\ j \in [k] : (x_{\alpha_j}, w_j) \in R\}$. Let $\mathcal{L}_t$ 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 $(k, x_1, \dots, x_n) \in \mathcal{L}_t$ into a proof for $(k, x_1, \dots, x_n, x_{n+1}) \in \mathcal{L}_t$.
- **Add**: transform a proof for $(k, x_1, \dots, x_n) \in \mathcal{L}_t$ into a proof for $(k+1, x_1, \dots, x_n) \in \mathcal{L}_t$.

While the extend operation can be realized without using any private input of the "previous" prover, as modelled in [13], the same does not hold for the add operation. Indeed, thanks to extractability, an accepting proof for $(k + 1, x_1, \dots, x_n) \in \mathcal{L}_t$ 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 $(k, x_1, \dots, x_n) \in \mathcal{L}_t$, 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 $(k + 1, x_1, \dots, x_n) \in \mathcal{L}_t$. 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 $(k, x_1, \dots, x_n) \in \mathcal{L}_t$, and a statement x_{n+1}, outputs a proof Π' for $(k, x_1, \dots, x_{n+1}) \in \mathcal{L}_t$ 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. $(x, w^i) \in R_t$ for $i \in \{0, 1\}$. 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 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 [22,12] 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 allow to replace an $\hat{M}_i$ allowing for simulation with a new one preventing simulation. 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 [19], 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 [19] 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 pk_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 sk_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 $(1, x_1, \ldots, x_n) \in R_t$. 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. 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 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 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[4] SoK without 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

[4] Chase and Lysyanskaya [14] 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.

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. [11]. 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 [1,31,37,25], while some others proposed constructions with signature size that can be sub-linear in n [35,4,5][5], or even $\mathcal{O}(t)$ [28,24]. Some works have also focused on providing post-quantum security [1,7,25].

In [30], 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 [4,5], 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. [28] 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 Suppl. A.2 for more widely known definitions and assumptions. 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 $e : \hat{\mathbb{G}} \times \check{\mathbb{H}} \rightarrow \mathbb{T}$

[5] In particular, [35] has size $\mathcal{O}(t\sqrt{n})$, [5] is $\mathcal{O}(t \log n)$, and [4] is $\mathcal{O}(\log n)$.

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*

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

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

See Suppl. A.3 for the proof.

3.1 Groth-Sahai Proofs

The Groth-Sahai proof system [23] 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 [17]. 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 (Suppl. A.2). 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 $\mathsf{PrVerify}$ and RandPr algorithms of the NIWI PoK interface have just to appropriately parse their inputs and call the $\mathsf{PrVerify}$ 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[17] (i.e., a variation of adaptive extractable soundness, see Def. 14 in Suppl. A.2) 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 [17], 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.

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

In Fig. 3 and in Fig. 4, we report the prover and verifier algorithm respectively. Finally, for space reasons, we defer to Suppl. A.4 for a list of the possible

```
$(\mathsf{crs}, xk) \leftarrow \mathsf{Ext}_1(\mathsf{gk})$
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.

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

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

commitment labels for each equation type, as well as for a description of the proof re-randomization algorithm.

```
$\mathsf{Prove}(L, \mathit{\Gamma}, \{(l_{x_i}, x_i, (r_{x_i}, s_{x_i}))\}_{i=1}^m, \{(l_{y_j}, y_j, (r_{y_j}, s_{y_j}))\}_{j=1}^n)$
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 $L = \mathrm{QE}$ $\alpha \leftarrow\!\$\, \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} = \boldsymbol{r}_x\mathit{\Gamma}\check{D} + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$    $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} = (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} = \boldsymbol{s}_x\mathit{\Gamma}\check{D} + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$    $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} = (\hat{C} - \hat{\boldsymbol{v}}\boldsymbol{r}_x - \hat{\boldsymbol{w}}\boldsymbol{s}_x)\mathit{\Gamma}\boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$
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.

```
$\mathsf{PrVerify}(L, \Gamma, \{(l_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^{m}, \{(l_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^{n}), \boldsymbol{\pi})$
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 $\hat{C}\Gamma\check{D} = \hat{\boldsymbol{v}}\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + \hat{\boldsymbol{w}}\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}\check{\boldsymbol{v}} + \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}\check{\boldsymbol{w}}$
return 1 if and only if all checks pass and 0 otherwise.
```

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

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.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$: 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. In figure 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.

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 $\mathcal{R}$, for any chosen value $1 \le t < |\mathcal{R}|$ it holds that:*

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

Definition 2 (Unforgeability for ETRS). *An extendable threshold ring signature scheme* ETRS *is said to be unforgeable if for all thresholds t, 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 anonimity if for all thresholds t, 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 ladder submitted by $\mathcal{A}$ are said to be well-formed if all the actions in the ladder are pairwise of the same type, and they have the same ring as input.*

Remarks on anonymity and unforgeability for ETRS. We modify our definition of anonymity for ETRS making it stronger w.r.t. the one in [2]. 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 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 Combine algorithm can be run without knowing any secret key. In [2], the authors showed that the Join operation can be obtained 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

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

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

```
$\mathsf{OKey}(i, \mathsf{pk})$
if $\mathsf{pk} = \perp$
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{ETRS.KeyGen}()$
  $\mathsf{L_{keys}} \leftarrow \mathsf{L_{keys}} \cup \{(i, \mathsf{pk}_i, \mathsf{sk}_i)\}$
else
  $\mathsf{L_{corr}} \leftarrow \mathsf{L_{corr}} \cup \{i\}$
  $\mathsf{pk}_i \leftarrow \mathsf{pk}$
  $\mathsf{L_{keys}} \leftarrow \mathsf{L_{keys}} \cup \{(i, \mathsf{pk}_i, \perp)\}$
return $\mathsf{pk}_i$
```

```
$\mathsf{OCorr}(i)$
if $(i, \mathsf{pk}_i, \mathsf{sk}_i) \in \mathsf{L_{keys}} \wedge \mathsf{sk}_i \neq \perp$
  $\mathsf{L_{corr}} \cup \{i\}$
  return $(\mathsf{pk}_i, \mathsf{sk}_i)$
return $\perp$
```

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

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

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 thresholds t, for all PPT adversaries $\mathcal{A}$, the success probability in the experiment of Fig. 8 is* $\Pr\Big[\mathsf{Exp}^{ANFS}_{\mathcal{A},\mathsf{ETRS}}(\lambda) = win\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

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_t as follows. We name the corresponding language $\mathcal{L}_t$.

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

An extendable non-interactive proof system for a threshold relation R_t 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:

```
$\mathsf{Exp}^{\mathrm{ANEXT}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
$b \leftarrow\!\$\ \{0,1\}, \mathsf{L_{keys}}, \mathsf{L_{corr}}, \mathsf{L_{sign}}, \mathsf{L_{join}} \leftarrow \emptyset$
$pp \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
$\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$
$(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
$\Sigma \leftarrow \mathsf{Chal}_b(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*)$
$b^* \leftarrow \mathcal{A}^{\mathsf{O}}(\Sigma)$
if $\exists\, i \in \mathsf{lad}_0^*.\mathcal{S}$ $s.t.$ $i \in \mathsf{L_{corr}}$
  return $lose$
if $\exists\, i \in \mathsf{lad}_1^*.\mathcal{S}$ $s.t.$ $i \in \mathsf{L_{corr}}$
  return $lose$
if $\exists\, (m^*, \cdot, i) \in \mathsf{L_{sign}}$ for $i \in \mathsf{lad}_0^*.\mathcal{S}$
  return $lose$
if $\exists\, (m^*, \cdot, i) \in \mathsf{L_{sign}}$ for $i \in \mathsf{lad}_1^*.\mathcal{S}$
  return $lose$
if $\exists\, (m^*, i, \cdot) \in \mathsf{L_{join}}$ for $i \in \mathsf{lad}_0^*.\mathcal{S}$
  return $lose$
if $\exists\, (m^*, i, \cdot) \in \mathsf{L_{join}}$ for $i \in \mathsf{lad}_1^*.\mathcal{S}$
  return $lose$
if $b^* \neq b$ return $lose$
return $win$
```

```
$\mathsf{Chal}_b(m^*, \mathsf{lad}_0^*, \mathsf{lad}_1^*)$
if $\mathsf{lad}_0^*$, or $\mathsf{lad}_1^*$ is not well-formed
  return $\perp$
if $\exists\, i \in \mathsf{lad}_0^*.\mathcal{S}$ $s.t.$ $i \in \mathsf{L_{corr}}$
  return $\perp$
if $\exists\, i \in \mathsf{lad}_1^*.\mathcal{S}$ $s.t.$ $i \in \mathsf{L_{corr}}$
  return $\perp$
if $\exists\, i \in \mathsf{lad}_0^*\mathcal{S}$ $s.t.$ $(\mathsf{pk}_i, \cdot) \notin \mathsf{L_{keys}}$
  return $\perp$
if $\exists\, i \in \mathsf{lad}_1^*.\mathcal{S}$ $s.t.$ $(\mathsf{pk}_i, \cdot) \notin \mathsf{L_{keys}}$
  return $\perp$
$(\Sigma_0, t_0, \mathcal{R}_0) \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}_0^*)$
$(\Sigma_1, t_1, \mathcal{R}_1) \leftarrow \mathsf{Proc}(m^*, \mathsf{L_{keys}}, \mathsf{lad}_1^*)$
$\Sigma \leftarrow \Sigma_b$
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.

```
$\mathsf{Proc}(m, \mathsf{L_{keys}}, \mathsf{lad})$
$\Sigma \leftarrow \emptyset, t = 0$
Parse $\mathsf{lad}$ as $((\mathsf{action}^1, \mathsf{input}^1), \ldots, (\mathsf{action}^l, \mathsf{input}^l))$
if $\mathsf{action}^1 \neq \mathsf{Sign}$ return $\perp$
else
    Parse $\mathsf{input}^1$as $(\mathcal{R}^1, i^1)$
    for $j \in \mathcal{R}^1$ if $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L_{keys}}$ return $\perp$
    if $sk_i = \perp \vee i \notin \mathcal{R}^1$ return $\perp$
    $\mathcal{R} \leftarrow \mathcal{R}^1, \mathcal{S} \leftarrow \{i^1\}$
    $\sigma \leftarrow \mathsf{Sign}(m, , \mathsf{sk}_{i^1}), \Sigma \leftarrow \Sigma \cup \{\sigma\}$
for $l' \in [2, \ldots, l]$
    if $\mathsf{action}^{l'} = \mathsf{Sign}$ return $\perp$
    else
        if $\mathsf{action}^{l'} = \mathsf{Join}$ parse $\mathsf{input}^{l'}$ as $(i^{l'})$
            if $i^{l'} \notin \mathcal{R} \vee i^{l'} \in \mathcal{S}$ return $\perp$
            $\sigma \leftarrow \mathsf{Join}(m, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \mathsf{sk}_i^{l'}, \sigma)$
            $\Sigma \leftarrow \Sigma \cup \{\sigma\}, \mathcal{S} \leftarrow \mathcal{S} \cup \{i^{l'}\}, t = t + 1$
        if $\mathsf{action}^{l'} = \mathsf{Extend}$ parse $\mathsf{input}^{l'}$ as $(\mathcal{R}^{l'})$
            for $j \in \mathcal{R}^{l'}$if $(j, \mathsf{pk}_j, \cdot) \notin \mathsf{L_{keys}}$ return $\perp$
            $\sigma \leftarrow \mathsf{Extend}(m, \sigma, \{\mathsf{pk}_j\}_{j \in \mathcal{R}}, \{\mathsf{pk}_j\}_{j \in \mathcal{R}^{l'}})$
            $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}^{l'}, \Sigma \leftarrow \Sigma \cup \{\sigma\}$
return $(\Sigma, t, \mathcal{R})$
```

Fig. 7. Process algorithm for ETRS.

- $\mathsf{crs} \leftarrow\!\$\, \mathsf{CRSSetup}(\mathsf{gk})$: on input the group key gk, output a uniformly random common reference string $\mathsf{crs} \in \{0,1\}^{\lambda}$[6].
- $(\Pi, (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)))$: on input $((k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k))) \in R_t$, 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.

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

```
$\mathsf{Exp}^{\mathrm{ANFS}}_{\mathcal{A},\mathsf{ETRS}}(\lambda)$
$b \leftarrow\!\$\, \{0,1\}$
$\mathsf{L}_{\mathsf{keys}}, \mathsf{L}_{\mathsf{corr}}, \mathsf{L}_{\mathsf{sign}}, \mathsf{L}_{\mathsf{join}} \leftarrow \emptyset$
$pp \leftarrow \mathsf{ETRS.Setup}(1^\lambda)$
$\mathsf{O} \leftarrow \{\mathsf{OSign}, \mathsf{OKey}, \mathsf{OCorr}, \mathsf{OJoin}\}$
$(m^*, \mathsf{lad}^*) \leftarrow \mathcal{A}^{\mathsf{O}}(pp)$
$\Sigma \leftarrow \mathsf{Chal}_b(m^*, \mathsf{lad}^*, i^*, j^*)$
$b^* \leftarrow \mathcal{A}^{\mathsf{O}}(\Sigma)$
$\textbf{if } i^* \in \mathsf{L}_{\mathsf{corr}} \vee j^* \in \mathsf{L}_{\mathsf{corr}}$
    $\textbf{return } \mathit{lose}$
$\textbf{if } \exists\,(m^*, \cdot, i^*) \in \mathsf{L}_{\mathsf{sign}} \vee (m^*, \cdot, j^*) \in \mathsf{L}_{\mathsf{sign}}$
    $\textbf{return } \mathit{lose}$
$\textbf{if } \exists\,(m^*, i^*, \cdot) \in \mathsf{L}_{\mathsf{join}} \vee (m^*, j^*, \cdot) \in \mathsf{L}_{\mathsf{join}}$
    $\textbf{return } \mathit{lose}$
$\textbf{if } b^* \neq b$
    $\textbf{return } \mathit{lose}$
$\textbf{return } \mathit{win}$
```

```
$\mathsf{Chal}_b(m^*, \mathsf{lad}^*, i^*, j^*)$
$\textbf{if } \mathsf{lad}^* \text{ is not well-formed}$
    $\textbf{return } \perp$
$\textbf{if } i^* \in \mathsf{L}_{\mathsf{corr}} \vee j^* \in \mathsf{L}_{\mathsf{corr}}$
    $\textbf{return } \perp$
$\textbf{if } \exists\, i \in \mathsf{lad}^*.\mathcal{S}\ s.t.\ (\mathsf{pk}_i, \cdot) \notin \mathsf{L}_{\mathsf{keys}}$
    $\textbf{return } \perp$
$\textbf{if } \ \nexists\, i^* \in \mathsf{L}_{\mathsf{keys}} \vee j^* \in \mathsf{L}_{\mathsf{keys}}$
    $\textbf{return } \perp$
$\mathsf{lad}^*.\mathit{add}((\mathsf{Extend}, \{i^*\}))$
$\mathsf{lad}^*.\mathit{add}((\mathsf{Extend}, \{j^*\}))$
$\textbf{if } b = 0$
    $\mathsf{lad}^*.\mathit{add}((\mathsf{Join}, i^*))$
$\textbf{if } b = 1$
    $\mathsf{lad}^*.\mathit{add}((\mathsf{Join}, j^*))$
$(\Sigma, t, \mathcal{R}) \leftarrow \mathsf{Proc}(m^*, \mathsf{L}_{\mathsf{keys}}, \mathsf{lad}^*)$
$\textbf{return } \Sigma$
```

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

- $(\Pi', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, (k, x_1, \ldots, x_n), x_{n+1}, \Pi)$: on input statements $(k, x_1, \ldots, x_n)$, x_{n+1}, and a proof Π for $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$, output an updated proof Π' for $(k, x_1, \ldots, x_n, x_{n+1}) \in \mathcal{L}_t$, 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 $(k, x_1, \ldots, x_n) \in \mathcal{L}_t$, output an updated proof Π' for $(k+1, x_1, \ldots, x_n) \in \mathcal{L}_t$, 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 $x \in \mathcal{L}_t$, 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[7].

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) and the following properties.

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

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

Definition 6 (Transformation Completeness). *An extendable non-interactive proof system for* R_t *is transformation complete if* $\forall \lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $\mathsf{crs} \leftarrow\!\!\$\, \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in R_t$, *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'. *Then, assuming that* $\alpha' \notin \{\alpha_1, \ldots \alpha_k\}$, $(x_{\alpha'}, w') \in R_{\mathcal{L}}$, $\mathsf{PrVerify}(\mathsf{crs}, (k+1, x_1, \ldots, x_n), \mathit{\Pi}') = 1$, *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:*

- $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$
- $\mathsf{crs} \leftarrow\!\!\$\, \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w, \mathit{\Pi}^*, \mathsf{AUX}^*) \leftarrow \mathcal{A}(\mathsf{crs})$
- *Parse* x *as* $(k, x_1, \ldots, x_n)$ *and* w *as* $((w_1, \alpha_1) \ldots, (w_k, \alpha_k))$
- *If* $(x, w) \notin R_t$ *or* $\mathsf{PrVerify}(\mathsf{crs}, (k{-}1, x_1, \ldots, x_n), \mathit{\Pi}^*) = 0$ *or* $\mathsf{AuxVerify}(\mathsf{crs}, (k{-}1, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_{k-1}, \alpha_{k-1})), \mathsf{AUX}^*, \mathit{\Pi}^*) = 0$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\!\!\$\, \{0, 1\}$ *and do the following:*
  - *If* $b = 0$, $(\mathit{\Pi}_0, \mathsf{AUX}_0) \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$; $(\mathit{\Pi}, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \mathit{\Pi}_0)$, $\mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}_0, r)$

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

  - *If* $b = 1$, $(\Pi_1, \mathsf{aux}^*) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, x, (w_k, \alpha_k), \mathsf{AUX}^*, \Pi^*)$. *Replace in* AUX^* *the value* aux_{α_k} *with* aux^*. $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \Pi_1), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}^*, r)$
- $b' \leftarrow \mathcal{A}(\Pi, \mathsf{AUX})$

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* $\Pr[b = b'] \leq 1/2 + \nu(\lambda)$.

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

- $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$
- $\mathsf{crs} \leftarrow\$\, \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w, x_n, \Pi^*, \mathsf{AUX}^*) \leftarrow \mathcal{A}(\mathsf{crs})$
- *Parse* x *as* $(k, x_1, \ldots, x_{n-1})$
- *If* $(x, w) \notin R_t$ *or* $\mathsf{PrVerify}(\mathsf{crs}, x, \Pi^*) = 0$ *or* $\mathsf{AuxVerify}(\mathsf{crs}, x, w, \mathsf{AUX}^*, \Pi^*) = 0$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\$\, \{0, 1\}$ *and do the following:*
  - *If* $b = 0$ $(\Pi_0, \mathsf{AUX}_0) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), w)$*;* $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi_0), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}_0, r)$
  - *If* $b = 1$ $(\Pi_1, \mathsf{aux}^*) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, x, x_n, \Pi^*)$. *Append the value* aux^* *to* AUX^*. $(\Pi, r) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \Pi_1), \mathsf{AUX} \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{AUX}^*, r)$
- $b' \leftarrow \mathcal{A}(\Pi, \mathsf{AUX})$

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* $\Pr[b = b'] \leq 1/2 + \nu(\lambda)$.

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

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

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[8].

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

[8] Our ENIWI PoK is based on GS, so we need a cryptographic hash function that allows to hash directly to both the source groups of the pairing group. See [20] for more details.

constrained to have disjoint rings. Indeed, our instantiation of Sec. 7.3 could be 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 an 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 will prove Thm. 1 using Lem. 2 to prove that the signature scheme described in Fig. 9 is unforgeable and Lem. 3 to prove that the signature scheme described in Fig. 9 is anonymous.

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 rerandomizable 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

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

```
$\mathsf{Extend}(m, \sigma, \{\mathsf{pk}_i\}_{i\in\mathcal{R}}, \mathsf{pk}^*)$
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 $\sigma = (k, \mathit{\Pi}, \mathsf{A})$
Let $x = (k, x_1, \ldots, x_{n+1})$
$(\mathit{\Pi}, \mathsf{aux}) \leftarrow \mathsf{ENIWI.PrExtend}(x, x_{n+2}, \mathit{\Pi})$
$\mathsf{a} \leftarrow \mathsf{PKE.Enc}(\mathsf{aux}, \mathsf{pk}_\mathsf{e}^{n+2})$
$\mathsf{A} \leftarrow \mathsf{A} \cup \mathsf{a}$
Let $\bar{x} = (k, x_1, \ldots, x_{n+2})$
$(\mathit{\Pi}, r_1, \ldots, r_{n+2}) \leftarrow \mathsf{ENIWI.RandPr}(\bar{x}, \mathit{\Pi})$
for $\mathsf{a}_i \in \mathsf{A}$
  $\mathsf{a}_i \leftarrow \mathsf{PKE.Eval}(\mathsf{a}_i, r_i, \mathsf{pk}_\mathsf{e}^i)$
return $\sigma = (k, \mathit{\Pi}, \mathsf{A})$
```

```
$\mathsf{KeyGen}()$
$(\mathsf{pk}_\mathsf{e}, \mathsf{sk}_\mathsf{e}) \leftarrow \mathsf{PKE.KeyGen}()$
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 $\sigma = (k, \mathit{\Pi}, \mathsf{A}), \mathsf{A} = (\mathsf{a}_1, \ldots, \mathsf{a}_{n+1})$
Parse $\mathsf{sk} = (w, \mathsf{sk}_\mathsf{e})$
$\mathsf{aux} \leftarrow \mathsf{PKE.Dec}(\mathsf{a}_j, \mathsf{sk}_\mathsf{e})$
Let $x = (k, x_1, \ldots, x_{n+1})$
$(\mathit{\Pi}, \mathsf{aux}'_j) \leftarrow \mathsf{ENIWI.PrAdd}(x, (w, j), \mathsf{aux}, \mathit{\Pi})$
Set $\mathsf{a}_j \in \mathsf{A}$ as $\mathsf{a}_j \leftarrow \mathsf{PKE.Enc}(\perp, \mathsf{pk}_\mathsf{e}^j)$
$k \leftarrow k + 1$
$(\mathit{\Pi}, r_1, \ldots, r_{n+1}) \leftarrow \mathsf{ENIWI.RandPr}(x, \mathit{\Pi})$
for $\mathsf{a}_i \in \mathsf{A}$
  $\mathsf{a}_i \leftarrow \mathsf{PKE.Eval}(\mathsf{a}_i, r_i, \mathsf{pk}_\mathsf{e}^i)$
return $\sigma = (k, \mathit{\Pi}, \mathsf{A})$
```

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

Fig. 9. ETRS from ENIWI PoK and IND-CPA homomorphic PKE. For space reasons, we omit crs from the input of the ENIWI algorithms and consider it as an implicit input. We use $\mathsf{AUX}[i]$ to indicate the i-th element of list AUX.

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 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 $R_{\mathcal{L}}$. For the complete proof we defer to Suppl. A.5.

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. For the complete proof we defer to Suppl. A.6.

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 [22,12], 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 $\mathit{\Pi}'$.

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

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

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 $\hat{M}_i = \hat{g}$.

It is pretty straightforward to notice that, 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

We now describe our ENIWI in detail. In particular, it 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 $\mathsf{crs} = (\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \check{\boldsymbol{u}}, \check{\boldsymbol{v}}, \check{\boldsymbol{w}})$.

– $(\Pi, (\mathsf{aux}_1, \ldots, \mathsf{aux}_n)) \leftarrow \mathsf{Prove}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)))$: on input $((k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k))) \in R_t$, define $A = \{\alpha_1, \ldots, \alpha_k\}$[9] 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[10]. 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[11]:
     - 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 $\hat{c}_{\hat{M}_i} = \boldsymbol{e}^{\top}\hat{M}_i + \hat{\boldsymbol{v}} r_i + \hat{\boldsymbol{w}} s_i$.

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

[10] 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.

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

  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 $(\mathit{\Pi} = (\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}}), \mathsf{AUX} = (\mathsf{aux}_1, \dots, \mathsf{aux}_n))$.

- $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, (k, x_1, \dots, x_n), \mathit{\Pi})$ reconstruct equations of type $\mathcal{B}, \mathcal{M}, \mathcal{Y}, \mathcal{X}, \mathcal{K}$, appropriately parse $\mathit{\Pi}$, 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.
- $(\mathit{\Pi}', \mathsf{aux}_{n+1}) \leftarrow \mathsf{PrExtend}(\mathsf{crs}, (k, x_1, \dots, x_n), x_{n+1}, \mathit{\Pi})$ do the following:
  1. Parse $\mathit{\Pi}$ as $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}}, C_{\mathcal{K}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}}, \pi_{\mathcal{K}}), \mathsf{AUX} = (\mathsf{aux}_1, \dots, \mathsf{aux}_n)$.
  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 $(\mathit{\Pi}, \mathsf{aux}_{n+1})$.
- $(\mathit{\Pi}', \mathsf{aux}'_\alpha) \leftarrow \mathsf{PrAdd}(\mathsf{crs}, (k, x_1, \dots, x_n), (w, \alpha), \mathsf{aux}, \mathit{\Pi})$ do the following:
  1. Parse $\mathit{\Pi}$ 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 $(\mathit{\Pi}, \mathsf{aux}'_\alpha)$.
- $(\mathit{\Pi}', r_1, \ldots, r_n) \leftarrow \mathsf{RandPr}(\mathsf{crs}, (k, x_1, \ldots, x_n), \mathit{\Pi})$:
    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 $\mathit{\Pi}'$.
    2. Output $(\mathit{\Pi}', r_1, \ldots, r_n)$
- $\mathsf{aux}' \leftarrow \mathsf{AuxUpdate}(\mathsf{crs}, \mathsf{aux}, r)$:
    1. Parse r as (r', s'), and aux as $(\mathsf{aux}^1, \mathsf{aux}^2)$.
    2. Output $\mathsf{aux}' = (\mathsf{aux}^1 + r'\check{h}, \mathsf{aux}^2 + s'\check{h})$.
- $0/1 \leftarrow \mathsf{AuxVerify}(\mathsf{crs}, (k, x_1, \ldots, x_n), ((w_1, \alpha_1) \ldots, (w_k, \alpha_k)), (\mathsf{aux}_1, \ldots, \mathsf{aux}_n), \mathit{\Pi})$:
    1. Parse $\mathit{\Pi}$ 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 Suppl. A.7 for the proof.

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, Virtual Event.

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., Friolo, D., Visconti, I.: Efficient proofs of knowledge for threshold relations. In: ESORICS 2022 , Part III. LNCS, vol. 13556, pp. 42–62. Springer.
6. 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.
7. Bettaieb, S., Schrek, J.: Improved lattice-based threshold ring signature scheme. In: PQCrypto 2013. pp. 34–51. Springer.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: EUROCRYPT 2012. LNCS, vol. 7237, pp. 281–300. Springer.
14. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer.
15. 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.
16. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer.
17. Escala, A., Groth, J.: Fine-tuning Groth-Sahai proofs. In: PKC 2014. LNCS, vol. 8383, pp. 630–649. Springer.
18. 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.
19. Faonio, A., Fiore, D., Nizzardo, L., Soriente, C.: Subversion-Resilient Enhanced Privacy ID. In: CT-RSA 2022. LNCS, vol. 13161, pp. 562–588. Springer.
20. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics **156**(16), 3113–3121. Applications of Algebra to Cryptography
21. 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.
22. 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.
23. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer.

24. 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.
25. 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.
26. 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.
27. 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.
28. 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.
29. Naor, M.: Deniable ring authentication. In: CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer.
30. Okamoto, T., Tso, R., Yamaguchi, M., Okamoto, E.: A k-out-of-n ring signature with flexible participation for signers. Cryptology ePrint Archive, Report 2018/728.
31. Petzoldt, A., Bulygin, S., Buchmann, J.: A multivariate based threshold ring signature scheme. Appl. Algebra Eng. Commun. Comput. **24**(3-4), 255–275.
32. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer.
33. 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.
34. 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.
35. 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.
36. Zhang, F., Kim, K.: ID-based blind signature and ring signature from pairings. In: ASIACRYPT 2002. LNCS, vol. 2501, pp. 533–547. Springer.
37. Zhou, G., Zeng, P., Yuan, X., Chen, S., Choo, K.R.: An Efficient Code-Based Threshold Ring Signature Scheme with a Leader-Participant Model. Secur. Commun. Networks **2017**, 1915239:1–1915239:7.

A Auxiliary Supporting Material

A.1 A Closer Look to the Results of [2]

In this section, we give a high-level overview of the two ETRS presented in [2].

SMLERS-based ETRS. The first ETRS given in [2] is based on same-message linkable ring signatures (SMLERS). In a nutshell, a SMLERS is a ring signature which is provided with the Extend algorithm. In addition, it allows to link two signatures produced by the same signer on the same message, even on different rings. This is done via a linkability tag that is uniquely determined by the signer and the message. To construct an ETRS, it simply suffices to concatenate several SMLERS on the same message, provided they do not carry the same linkability tag, and to extend all the SMLERS to have the same ring. Their SMLERS has size $\mathcal{O}(n)$, where n is the size of the ring. Therefore, the compiled ETRS has size $\mathcal{O}(tn)$. The same reasoning applies to the time complexity of most of the operations. Although the anonymity property was proven in [2] according to their weaker definition, it seems reasonable to assume that this construction could be proven secure under our stronger definition without much effort.

DL+SoK+PKE ETRS. The second construction of ETRS proposed in [2] works with a prime order group, with two public group elements (g, H), and a signature of knowledge for relation R for knowledge of the discrete logarithm either of a certain value h or of a pk. When the first signature is generated, the signer generates N points of the form $(x_i, td_i) \in \mathbb{Z}_p^2$. All these points define a unique polynomial p of degree N such that $p(0) = dlog(H)$ and $p(x_i) = td_i$ for all $i \in [N]$. The discrete log of H is not known, but this polynomial can be interpolated in the exponent to obtain a polynomial f s.t. $f(x) = g^{p(x)}$. To either sign or join a signature, the signer has to produce a signature of knowledge for R using a random point $(x, y = g^{p(x)})$ with $x \notin \{x_i\}_{i \in [N]}$. If the DL problem is hard, the signer cannot know the discrete log of y and thus the signature of knowledge must satisfy the second clause of R which requires proving knowledge of the secret key. To extend a signature instead, an $x \in \{x_i\}_{i \in [N]}$ will be used, so that the corresponding trapdoor can be used to satisfy the first clause of the relation. The used pair (x_i, td_i) is removed from the list of trapdoors. However, to make the owner of the public key pk able to join at a later time, along with the signature of knowledge, the point x_i is posted together with an encryption of td_i under pk. As a result, the owner of pk can decrypt td_i and put it back to the list of trapdoors, before producing a fresh signature of knowledge using her secret key as a witness. Although neither space nor time complexities depend on the threshold t, they all take time and space proportional to N even though the actual ring size $n << N$. In particular, every operation requires to interpolate the polynomial in the exponent, which takes quadratic time in the number of interpolated points N. Regarding anonymity, it is pretty straightforward to observe that Def. 3 is not satisfied. Indeed, after a join operation, a “trapdoored” point x_i which was related to a signature of knowledge for a certain key pk_i goes back in the trapdoor list, and it is replaced by a new random "non-trapdoored" point. This movement of x_i from one list to another clearly indicates that pk_i has joined the signature.

A.2 Assumptions and Cryptographic Tools

Assumption 2 (DDH) *The Decisional Diffie-Hellman (DDH) holds in $\hat{\mathbb{G}}$ if for all PPT adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ distinguishes the two distributions $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \xi\rho\hat{g})$ and $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \kappa\hat{g})$, where $\xi, \rho, \kappa \leftarrow \mathbb{Z}_p$ is negligible. Tuples of the form $(\hat{g}, \xi\hat{g}, \rho\hat{g}, \xi\rho\hat{g})$ are called Diffie-Hellman (DH) tuples. The DDH problem in $\check{\mathbb{H}}$ is defined in a similar way.*

Assumption 3 (SXDH) *The Symmetric eXternal Diffie-Hellman (SXDH) assumption holds relative to $\mathcal{G}$ if there is no PPT adversary $\mathcal{A}$ that breaks the DDH problem in both $\hat{\mathbb{G}}$ and $\check{\mathbb{H}}$ for $\mathsf{gk} = (p, \hat{\mathbb{G}}, \check{\mathbb{H}}, \mathbb{T}, e, \hat{g}, \check{h}) \leftarrow \mathcal{G}(1^\lambda)$.*

Public Key Encryption. A public key encryption scheme is a set of PPT algorithms $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: on input the security parameter, outputs public parameters pp.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$: generates a new public and secret key pair.
- $\mathsf{a} \leftarrow \mathsf{Enc}(m, \mathsf{pk})$: on input a message m, and a public key pk, output a ciphertext a.
- $m \leftarrow \mathsf{Dec}(\mathsf{a}, \mathsf{sk})$ on input a ciphertext a, and a secret key sk, output a message m.

Additionally, a public key encryption scheme is homomorphic w.r.t a function f, if there exists a PPT algorithm that works as follows.

- $\mathsf{a}' \leftarrow \mathsf{Eval}(\mathsf{a}, x, \mathsf{pk})$: on input a ciphertext a, a message x, and the public key pk. Let $y \leftarrow \mathsf{Dec}(\mathsf{a}, \mathsf{sk})$, it returns a ciphertext a' s.t. $f(y, x) = \mathsf{Dec}(\mathsf{a}', \mathsf{sk})$.

A public key encryption scheme is is IND-CPA secure if the probability that a PPT adversary $\mathcal{A}$ wins the following game is negligibly close to $\frac{1}{2}$. The game involves the following steps: (i) $\mathcal{A}$ has access to the public key and outputs two messages m_0 and m_1; (ii) the challenger encrypts one of the two messages; (iii) $\mathcal{A}$ has to guess which message was encrypted.

ElGamal Encryption. The ElGamal encryption scheme is a public key encryption scheme with the following algorithms. The public parameters pp produced by Setup are implicitly available to all other algorithms:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$: on input the security parameter, sample a cyclic group $\hat{\mathbb{G}}$ of prime order p, a generator $\hat{g}$. Output $\mathsf{pp} = (\hat{\mathbb{G}}, \hat{g})$.
- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$: sample an element $\zeta \leftarrow\!\$\, \mathbb{Z}_p^*$. Define public key as $\mathsf{pk} = \hat{\boldsymbol{v}} = (\zeta\hat{g}, \hat{g})^\top \in \hat{\mathbb{G}}^{2\times 1}$ and $\mathsf{sk} = \boldsymbol{\zeta} = (-\zeta^{-1}, 1)$. Output $(\mathsf{pk}, \mathsf{sk})$.

- $\hat{\mathbf{a}} \leftarrow \mathsf{Enc}(\hat{m}, \mathsf{pk})$: with input the public key and a message $\hat{m} \in \hat{\mathbb{G}}$, sample $r \leftarrow\$\, \mathbb{Z}_p$ and output ciphertext $\hat{\mathbf{a}} = \boldsymbol{e}^\top \hat{m} + \hat{\boldsymbol{v}} r \in \hat{\mathbb{G}}^{2\times 1}$, where $\boldsymbol{e} = (0, 1)$.
- $\hat{m} \leftarrow \mathsf{Dec}(\hat{\mathbf{a}}, \mathsf{sk})$: with input the secret key and a ciphertext $\mathbf{a} \in \hat{\mathbb{G}}^{2\times 1}$, output $\hat{m} = \boldsymbol{\zeta}\hat{\mathbf{a}}$.

The ElGamal encryption scheme is also homomorphic, with the function f being the group operation. In more detail:

- $\mathbf{a}' \leftarrow \mathsf{Eval}(\mathbf{a}_1, \hat{m}_2, \mathsf{pk})$: compute $\mathbf{a_2} = \mathsf{Enc}(\hat{m}_2, \mathsf{pk})$, output $\mathbf{a}' = \mathbf{a}_1 + \mathbf{a}_2$. If the ciphertexts contained messages $\hat{m}_1$ and $\hat{m}_2$, the output ciphertext will contain message $\hat{m}_1 + \hat{m}_2$.

The ElGamal encryption is IND-CPA secure if the DDH assumption holds in $\hat{\mathbb{G}}$. Additionally, ciphertexts updated with Eval are identically distributed to freshly generated ciphertexts.

Non-interactive Witness Indistinguishable Proof of Knowledge. Let us consider an NP language $\mathcal{L}$ with associated poly-time relation $R_{\mathcal{L}}$. A non-interactive proof system for $R_{\mathcal{L}}$ consists of the following 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, output a uniformly random common reference string $\mathsf{crs} \in \{0,1\}^\lambda$.
- $\mathit{\Pi} \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$: on input statement x and witness w s.t. $(x, w) \in R_{\mathcal{L}}$, output a proof $\mathit{\Pi}$.
- $0/1 \leftarrow \mathsf{PrVerify}(\mathsf{crs}, x, \mathit{\Pi})$: on input statement x and proof $\mathit{\Pi}$, output either 1 to accept or 0 to reject.
- $\mathit{\Pi}' \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \mathit{\Pi})$: on input statement x and proof $\mathit{\Pi}$ for $x \in \mathcal{L}$, output a randomized proof $\mathit{\Pi}'$.

A non-interactive proof system is said to be witness indistinguishable (NIWI) if all the properties below are satisfied.

Definition 10 (Completeness). *A proof system for $R_{\mathcal{L}}$ is complete if $\forall \lambda \in \mathbb{N}$, $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, $\mathsf{crs} \leftarrow\$\, \mathsf{CRSSetup}(\mathsf{gk})$, $(x, w) \in R_{\mathcal{L}}$, and $\mathit{\Pi} \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ it holds that* $\Pr[\mathsf{PrVerify}(\mathsf{crs}, x, \mathit{\Pi}) = 1] = 1$.

Definition 11 (Re-Randomizable Proof System). *Consider the following experiment:*

- $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$
- $\mathsf{crs} \leftarrow\$\, \mathsf{CRSSetup}(\mathsf{gk})$
- $(x, w, \mathit{\Pi}) \leftarrow \mathcal{A}(\mathsf{crs})$
- *If either* $\mathsf{PrVerify}(\mathsf{crs}, x, \mathit{\Pi}) = 0$ *or* $(x, w) \notin R_{\mathcal{L}}$ *output* $\perp$ *and abort. Otherwise, sample* $b \leftarrow\$\, \{0, 1\}$.
  - *If* $b = 0$ $\mathit{\Pi}' \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$.
  - *If* $b = 1$ $\mathit{\Pi}' \leftarrow \mathsf{RandPr}(\mathsf{crs}, x, \mathit{\Pi})$.
- $b' \leftarrow \mathcal{A}(\mathit{\Pi}')$.

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

Definition 12 (Witness Indistinguishability). *We say that the proof system is witness indistinguishable (WI), if the following holds. For all (x, w_1, w_2) such that $(x, w_1), (x, w_2) \in R_{\mathcal{L}}$, the tuples* $(\mathsf{crs}, \mathit{\Pi}_1)$ *and* $(\mathsf{crs}, \mathit{\Pi}_2)$, *where* $\mathsf{crs} \leftarrow\$ \mathsf{CRSSetup}(\mathsf{gk})$, $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$ *and for $i \in [2]$, $\mathit{\Pi}_i \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w_i)$, are computationally indistinguishable. If the two tuples are identically distributed, we say that the proof system is perfect WI.*

Definition 13 (Soundness). *For all PPT $\mathcal{A}$, and for* $\mathsf{crs} \leftarrow\$ \mathsf{CRSSetup}(\mathsf{gk})$, $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, *the probability that* $\mathcal{A}(\mathsf{crs})$ *outputs $(x, \mathit{\Pi})$ such that $x \notin \mathcal{L}$ but* $\mathsf{PrVerify}(\mathsf{crs}, x, \mathit{\Pi}) = 1$, *is negligible.*

Additionally, a NIWI is said to be a NIWI proof of knowledge (PoK) if the property below is also satisfied.

Definition 14 (Adaptive Extractable Soundness). *There exists a polynomial-time extractor* $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$, *for all* $\mathsf{gk} \leftarrow \mathcal{G}(1^\lambda)$, *with the following properties:*

- $\mathsf{Ext}_1(\mathsf{gk})$ *outputs* $(\mathsf{crs}_{\mathsf{Ext}}, xk)$ *such that* $\mathsf{crs}_{\mathsf{Ext}}$ *is indistinguishable from* crs *obtained running* $\mathsf{crs} \leftarrow\$ \mathsf{CRSSetup}(\mathsf{gk})$.
- *For all PPT $\mathcal{A}$, the probability that* $\mathcal{A}(\mathsf{crs}_{\mathsf{Ext}}, xk)$ *outputs $(x, \mathit{\Pi})$ such that* $\mathsf{PrVerify}(\mathsf{crs}_{\mathsf{Ext}}, x, \mathit{\Pi}) = 1$ *and $(x, w) \notin R_{\mathcal{L}}$ where* $w \leftarrow \mathsf{Ext}_2(\mathsf{crs}_{\mathsf{Ext}}, xk, x, \mathit{\Pi})$ *is negligible.*

A.3 Proof of Lem. 1

Proof. We build an adversary $\mathcal{B}$ for the DDH assumption which makes black-box use of $\mathcal{A}$. $\mathcal{B}$ gets the DDH challenge $(\hat{a}, \hat{b}, \hat{c})$. If $\hat{a} = \hat{0}$ or $\hat{b} = \hat{0}$ or $\hat{c} = \hat{0}$, $\mathcal{B}$ aborts. This event occurs with negligible probability. $\mathcal{B}$ sets the challenge for $\mathcal{A}$ to be $(\hat{c}, \hat{b})$. With probability ϵ, $\mathcal{B}$ gets back $\check{b}' \in \check{\mathbb{H}}$ s.t. $\hat{c} \cdot \check{h} + \hat{b} \cdot \check{b}' = 0_{\mathbb{T}}$ from $\mathcal{A}$. $\mathcal{B}$ outputs 1 iff $\hat{a} \cdot \check{h} + \hat{g} \cdot \check{b}' = 0_{\mathbb{T}}$, and 0 otherwise. If $(\hat{a}, \hat{b}, \hat{c})$ is a DH tuple, $\mathcal{B}$ outputs 1 with probability exactly ϵ. In the other case, since $\hat{a} \neq \hat{0}$, $\hat{b} \neq \hat{0}$, and $\hat{c} \neq \hat{0}$, there is no trivial $\check{b}'$ s.t. $\hat{a} \cdot \check{h} + \hat{g} \cdot \check{b}' = 0_{\mathbb{T}}$.

A.4 More on GS Proofs Internals

In Fig. 10, we report the possible commitment labels for each equation type of GS proofs, while in Fig. 11, we report the proof re-randomization algorithm.

A.5 Proof of Lem. 2

Proof. We build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to extract a witness for an instance of $R_{\mathcal{L}}$ which is sampled by a challenger $\mathcal{C}$. In particular, $\mathcal{B}$ replies to all oracle queries of $\mathcal{A}$, including queries to the random oracle.

L	Labels for x_i, $i \in [m]$	Labels for y_j, $j \in [n]$
PPE	$\mathsf{base}_{\hat{\mathbb{G}}}, \mathsf{pub}_{\hat{\mathbb{G}}}, \mathsf{com}_{\hat{\mathbb{G}}}$	$\mathsf{base}_{\check{\mathbb{H}}}, \mathsf{pub}_{\check{\mathbb{H}}}, \mathsf{com}_{\check{\mathbb{H}}}$
$\mathrm{ME}_{\hat{\mathbb{G}}}$	$\mathsf{base}_{\hat{\mathbb{G}}}, \mathsf{pub}_{\hat{\mathbb{G}}}, \mathsf{com}_{\hat{\mathbb{G}}}$	$\mathsf{sca}_{\check{\mathbb{H}}}$
$\mathrm{ME}_{\check{\mathbb{H}}}$	$\mathsf{sca}_{\hat{\mathbb{G}}}$	$\mathsf{base}_{\check{\mathbb{H}}}, \mathsf{pub}_{\check{\mathbb{H}}}, \mathsf{com}_{\check{\mathbb{H}}}$
QE	$\mathsf{sca}_{\hat{\mathbb{G}}}$	$\mathsf{sca}_{\check{\mathbb{H}}}$

Fig. 10. Possible GS commit labels for each equation type.

```
$(\hat{C}', \check{D}', \pi') \leftarrow \mathsf{RandPr}(L, \Gamma, \{(l_{x_i}, \hat{\boldsymbol{c}}_i)\}_{i=1}^m, \{(l_{y_j}, \check{\boldsymbol{d}}_j)\}_{j=1}^n, \boldsymbol{\pi}; \boldsymbol{r})$
Parse $\boldsymbol{r} = (\boldsymbol{r}_x, \boldsymbol{s}_x, \boldsymbol{r}_y, \boldsymbol{s}_y)$
Define $\hat{C} = (\hat{\boldsymbol{c}}_1 \ldots \hat{\boldsymbol{c}}_m) \in \hat{\mathbb{G}}^{2\times m}, \check{D} = (\check{\boldsymbol{d}}_1 \ldots \check{\boldsymbol{d}}_n)^\top \in \check{\mathbb{H}}^{n\times 2}$and parse $\boldsymbol{\pi} = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}, \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}, \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}})$
**if** $\boldsymbol{x} \in \hat{\mathbb{G}}^m$ define $\hat{C}' = \hat{C} + \hat{\boldsymbol{v}}\boldsymbol{r}_x + \hat{\boldsymbol{w}}\boldsymbol{s}_x$ **else if** $\boldsymbol{x} \in \mathbb{Z}_p^m$ define $\hat{C}' = \hat{C} + \hat{\boldsymbol{v}}\boldsymbol{r}_x$
**if** $\boldsymbol{y} \in \check{\mathbb{H}}^n$ define $\check{D}' = \check{D} + \boldsymbol{r}_y\check{\boldsymbol{v}} + \boldsymbol{s}_y\check{\boldsymbol{w}}$ **else if** $\boldsymbol{y} \in \mathbb{Z}_p^n$ define $\check{D}' = \check{D} + \boldsymbol{r}_y\check{\boldsymbol{v}}$
**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** $L = \mathrm{QE}\ \alpha \leftarrow\$\, \mathbb{Z}_p$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}{}' = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}} + \boldsymbol{r}_x\Gamma\check{D}' + \alpha\check{\boldsymbol{v}} + \beta\check{\boldsymbol{w}}$        $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}{}' = \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}} + \hat{C}\Gamma\boldsymbol{r}_y - \hat{\boldsymbol{v}}\alpha - \hat{\boldsymbol{w}}\gamma$
    $\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}{}' = \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}} + \boldsymbol{s}_x\Gamma\check{D}' + \gamma\check{\boldsymbol{v}} + \delta\check{\boldsymbol{w}}$        $\hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}{}' = \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}} + \hat{C}\Gamma\boldsymbol{s}_y - \hat{\boldsymbol{v}}\beta - \hat{\boldsymbol{w}}\delta$
**return** $(\hat{C}', \check{D}', \pi' = (\check{\boldsymbol{\pi}}_{\hat{\boldsymbol{v}}}{}', \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{v}}}{}', \check{\boldsymbol{\pi}}_{\hat{\boldsymbol{w}}}{}', \hat{\boldsymbol{\pi}}_{\check{\boldsymbol{w}}}{}'))$
```

Fig. 11. Proof re-randomization algorithm of the GS proof system.

We prove unforgeability via a sequence of hybrid arguments. In each hybrid, $\mathcal{B}$ will change how it replies to certain queries, but in a way s.t. $\mathcal{A}$ cannot detect the change. In the final hybrid, $\mathcal{B}$ will be able to use a forgery from $\mathcal{A}$ to extract a valid witness for the instance sampled by $\mathcal{C}$.

$\mathcal{H}_0$**:** This is exactly the unforgeability game of figure Fig. 5.

$\mathcal{H}_1$**:** This is equivalent to $\mathcal{H}_0$ except that when replying to Join queries, $\mathcal{B}$ uses the Prove algorithm instead of the PrAdd algorithm. Additionally, before performing $\mathsf{AuxUpdate}$, each element of A is replaced with a fresh encryption of the auxiliary values in output of Prove algorithm. Let (k, Π_0, A_0) and (k, Π_1, A_1) the output of a Join query in $\mathcal{H}_0$ and $\mathcal{H}_1$ respectively. If $\mathcal{A}$ can distinguish between the two hybrids, $\mathcal{B}$ can use $\mathcal{A}$ to break the re-randomizable addition (Def. 7) of ENIWI. Let (Π', AUX') be the proof and auxiliary values (i.e., not encrypted) held by $\mathcal{B}$ before the Join query. Let $\mathcal{C}_{add}$ be the challenger of Def. 7. $\mathcal{B}$ sends $(x, w, \Pi', \mathsf{AUX}')$ ($\mathcal{B}$ can recover x, and w from $\mathsf{L}_{\mathsf{keys}}$) to $\mathcal{C}_{add}$ and receives back (Π, AUX). Once $\mathcal{A}$ asks for a join operation, $\mathcal{B}$ sends (k, Π, A), where A is the encryption of the elements in AUX (again the encryption keys are available in $\mathsf{L}_{\mathsf{keys}}$). This perfectly simulates the input of the adversary of the re-randomizable addition game Def. 7.

$\mathcal{H}_2$**:** This is equivalent to $\mathcal{H}_1$ except that $\mathcal{B}$ tries to guess an index $j^* \in [q_m]$, where q_m is a bound on the number of random oracle queries. In particular, $\mathcal{B}$'s guess is that the j^*-th *different* queried message will be the one w.r.t. $\mathcal{A}$ outputs its forgery. If the message used by $\mathcal{A}$ in the forgery differs from m_{j^*}, then $\mathcal{B}$ aborts. The probability that $\mathcal{B}$ does not abort in this hybrid is at least $\frac{1}{q_m}$.

$\mathcal{H}_3$**:** This is equivalent to $\mathcal{H}_2$ except that $\mathcal{B}$ programs the random oracle on message m_{j^*} to give as output a CRS $\mathsf{crs}_{\mathsf{Ext}}$ s.t. $(\mathsf{crs}_{\mathsf{Ext}}, xk) \leftarrow \mathsf{Ext}_1(\mathsf{gk})$. $\mathcal{H}_3$ is indistinguishable from $\mathcal{H}_2$ due to the adaptive extractable soundness of ENIWI. (Def. 14). Indeed, $\mathsf{crs}_{\mathsf{Ext}}$ is indistinguishable from a crs generated with $\mathsf{CRSSetup}$, which is in turn a random string.

$\mathcal{H}_4$ This is equivalent to $\mathcal{H}_3$ except that for every message m_j, with $j \neq j^*$, $\mathcal{B}$ programs the random oracle to give as output a random $\mathsf{pk}_{\mathsf{O}_j} = (x_{1_j}, \mathsf{pk}^1_{\mathsf{e}_\mathsf{j}})$ for which $\mathcal{B}$ knows w_{1_j} s.t. $(x_{1_j}, w_{1_j}) \in R_{\mathcal{L}}$. Since $R_{\mathcal{L}}$ is public-coin samplable, $\mathsf{pk}_{\mathsf{O}_j}$ is equally distributed in $\mathcal{H}_3$ and $\mathcal{H}_4$.

$\mathcal{H}_5$**:** Let us consider the forgery given in output by $\mathcal{A}$ as $(k^*, m_{j^*}, \mathcal{R}^*, (k^*, \Pi^*, \mathsf{A}^*))$. Let $x_{\tilde{j^*}}$ be the trapdoor statement corresponding to message m_{j^*}[12] and $\{x_i\}_{i \in \mathcal{R}^*}$ be the statements corresponding to all the users in the ring of the forgery. For $\mathcal{A}$ to be admissible, there must be at least a statement $x_{i^*} \in \{x_i\}_{i \in \mathcal{R}^*} \cup \{x_{\tilde{j^*}}\}$ that was not involved in any *corrupt* query, or any $\mathsf{Join}/\mathsf{Sign}$ w.r.t. the forgery message m_{j^*}. If this does not hold, the checks of the unforgeability experiment of line 8 of Fig. 5 cannot be successful.

Let $\mathcal{H}_5$ be equivalent to $\mathcal{H}_4$ except that $\mathcal{B}$ tries to guess i^* sampling it uniformly at random from $\{i\}_{i \in [q_{KG}]} \cup \{\tilde{j^*}\}$, where q_{KG} is a polynomial bound on the number of key generation queries that $\mathcal{A}$ can do. When $\mathcal{A}$ outputs its

[12] We use $\tilde{j^*}$ as a special index for the trapdoor statement related to m_{j^*}.

forgery, $\mathcal{B}$ uses the extractor $(w_1, \ldots, w_{k^*}) \leftarrow \mathsf{Ext}_2(\mathsf{crs}_{\mathsf{Ext}}, xk, (k^*, \{x_i\}_{i \in \mathcal{R}^*} \cup \{x_{\tilde{j^*}}\}), \mathit{\Pi}^*)$ to extract the witnesses from proof $\mathit{\Pi}^*$. If the extraction fails, or none of the extracted witnesses w_z, with $z \in [k^*]$, is s.t. $(x_{i^*}, w_z) \in R_{\mathcal{L}}$, $\mathcal{B}$ aborts. Due to the adaptive extractable soundness of ENIWI, the extraction fails only with negligible probability. Thus, the probability that $\mathcal{B}$ does not abort in this hybrid is at least $\frac{1}{q_{KG}+1}$.

$\mathcal{H}_6$: This is equivalent to $\mathcal{H}_5$ except that every time $\mathcal{A}$ makes Sign or Join queries involving i^* for message $m_j \neq m_{j^*}$, $\mathcal{B}$ answers using the witness w_1^j for the trapdoor statement x_{1_j} to compute the proof and the auxiliary values while still encrypting $\perp$ in a_{i^*}. The queries for message $m_j = m_{j^*}$ are answered in the same way, since no query for message m_{j^*} ever involves i^*.

If i^* is equal to $\tilde{j^*}$, then $\mathcal{H}_5$ and $\mathcal{H}_6$ are equally distributed since no Sign or Join query can involve $x_{\tilde{j^*}}$ by construction. Let us consider the case for which i^* is related to a registered key. We now argue that $\mathcal{H}_5$ is indistinguishable from $\mathcal{H}_6$ thanks to extended WI property (Def. 9). Let us call $\mathcal{D}$ the distinguisher that distinguishes $\mathcal{H}_5$ from $\mathcal{H}_6$ with probability greater than negligible. Let $\mathcal{A}_{\mathsf{EWI}}$ be the adversary that exploits $\mathcal{D}$ to break the extended WI of ENIWI against a challenger $\mathcal{C}_{\mathsf{EWI}}$. W.l.o.g. we consider a query $\mathsf{OSign}(m_j, \mathcal{R}, i^*)$ for message m_j, with $j \neq j^*$, and signer index i^*. Trivially, it can be extended to any query involving i^*.

1. $\mathcal{A}_{\mathsf{EWI}}$ chooses as statement x the public keys of the ring $\mathcal{R}$, $\mathcal{A}_{\mathsf{EWI}}$ chooses $w_0 = w_{1_{\tilde{j}}}$ and $w_1 = w_{i^*}$.
2. $\mathcal{A}_{\mathsf{EWI}}$ sends (x, w_0, w_1) to the challenger of extended WI $\mathcal{C}_{\mathsf{EWI}}$, and receives back $(\mathit{\Pi}, \mathsf{AUX})$[13].
3. $\mathcal{A}_{\mathsf{EWI}}$ generates the signature starting from $(\mathit{\Pi}, \mathsf{AUX})$ as in the Sign algorithm (except for the fact that a_{i^*} is obtained encrypting $\perp$) and sends this signature as answer to the query $\mathsf{OSign}(m_j, \mathcal{R}, i^*)$ performed by $\mathcal{A}$.
4. At the end of the experiment, $\mathcal{A}$ outputs the forgery $(k^*, m_{j^*}, \mathcal{R}^*, (k^*, \mathit{\Pi}^*, \mathsf{A}^*))$ to $\mathcal{A}_{\mathsf{EWI}}$.
5. $\mathcal{A}_{\mathsf{EWI}}$ runs $\mathcal{D}$ on input its view.$\mathcal{A}_{\mathsf{EWI}}$ returns 1 if $\mathcal{D}$ says that the hybrid is $\mathcal{H}_5$, and 0 otherwise.

$\mathcal{A}_{\mathsf{EWI}}$ breaks the extended WI property with the same advantage that $\mathcal{D}$ has in distinguishing the two hybrids. Therefore, $\mathcal{H}_5$ and $\mathcal{H}_6$ are indistinguishable[14]. Additionally, we now argue that the probability of extracting a witness for i^* does not change between the two hybrids, except for a negligible quantity. To see why this holds, let us consider an $\mathcal{A}_{\mathsf{EWI}}$ that at step 5, runs the extractor $\mathsf{Ext}_2(\mathsf{crs}_{\mathsf{Ext}}, xk, (k^*, \{x_i\}_{i \in \mathcal{R}^*} \cup \{x_{\tilde{j^*}}\}), \mathit{\Pi}^*)$ to extract the witnesses $(w_1, \ldots, w_{k^*})$ used in the forgery instead of calling $\mathcal{D}$.

[13] Notice that here AUX does not include aux_{i^*} and $\mathsf{aux}_{\tilde{j}}$.

[14] Notice that this reduction in principle does not capture the fact that $\mathcal{A}$ can perform multiple sequential queries in a single execution. Consider a modified EWI definition with a game in which the challenger accepts multiple queries and always replies according to the bit sampled at the beginning of the game. It is straightforward to observe that a proof system which fulfils the regular EWI definition also fulfils the modified EWI definition.

If the forgery contains w_{i^*} s.t. $(x_{i^*}, w_{i^*}) \in R_{\mathcal{L}}$, $\mathcal{A}_{\mathsf{EWI}}$ returns 1, and 0 otherwise. If $|\Pr[\mathcal{A}_{\mathsf{EWI}}$ outputs $1|\mathcal{H}_5] - \Pr[\mathcal{A}_{\mathsf{EWI}}$ outputs $1|\mathcal{H}_6]| > \mathsf{negl}(\lambda)$, then $\mathcal{A}_{\mathsf{EWI}}$ would itself be a distinguisher between the two hybrids. Therefore, since this would break the extended WI property, we reach a contradiction. Importantly, we exploit the fact that we have a *different* CRS for every message. Indeed, we only switch one CRS to the extraction mode; namely the one corresponding to the forgery message m_{j^*}. This allows us to simultaneously reduce to EWI for all the queries on messages $m_j \neq m_{j^*}$, and to extract the witnesses from the forgery on message m_{j^*} in order to finalize the game as shown in the next hybrid.

$\mathcal{H}_7$: This is equivalent to $\mathcal{H}_6$ except that when prompted to generate a key pair for i^*, $\mathcal{B}$ would return as public key the pair $(x, \mathsf{pk}_\mathsf{e})$, where x, instead of being freshly sampled, is the one that $\mathcal{C}$ sent to $\mathcal{B}$ in the hard relation game. $\mathcal{H}_6$ and $\mathcal{H}_7$ are equally distributed. Now, $\mathcal{B}$ extracts from the forgery a witness for the hard relation sampled by the challenger of the hard relation game, thus reaching a contradiction.

A.6 Proof of Lem. 3

Proof. **$\mathcal{H}_0$:** This is exactly the anonymous extendability game of Fig. 6.

$\mathcal{H}_1$: This is equivalent to $\mathcal{H}_0$ except that when running the Join algorithm, $\mathcal{B}$ uses the Prove algorithm instead of the PrAdd algorithm. Additionally, instead of performing $\mathsf{AuxUpdate}$, each element of A is replaced with a fresh encryption of the auxiliary values in output of Prove algorithm. Showing that $\mathcal{H}_1$ is indistinguishable from $\mathcal{H}_0$ basically mirrors the discussion of $\mathcal{H}_1$ in Lem. 2.

$\mathcal{H}_2$: This is equivalent to $\mathcal{H}_1$ except that when running the Extend algorithm in the ladder, $\mathcal{B}$ uses the Prove algorithm instead of the $\mathsf{PrExtend}$ algorithm. Additionally, instead of performing $\mathsf{AuxUpdate}$, each element of A is replaced with a fresh encryption of the auxiliary values in output of Prove algorithm. Showing that $\mathcal{H}_2$ is indistinguishable from $\mathcal{H}_1$ basically mirrors the discussion for the previous hybrid, except that now we would use $\mathcal{A}$ to break the rerandomizable extension property (cfr., Def. 8) of ENIWI.

$\mathcal{H}_3$: This is equivalent to $\mathcal{H}_2$ except that, when processing the ladders, $\mathcal{B}$ encrypts $\perp$ in all the signers' ciphertexts. Note that auxiliary values are already not used at all by $\mathcal{B}$ at this point. Recall that in order to win, $\mathcal{A}$ cannot corrupt any of the signers. This is indistinguishable from $\mathcal{H}_2$ thanks to the IND-CPA property of the encryption scheme [15].

[15] Let us consider an extended version of the IND-CPA game that is trivially implied by the standard IND-CPA game. In this extended game, the adversary $\mathcal{A}$ has access to a key generation and a corruption oracle. Let n be the number of key generation queries. At the end of this step, $\mathcal{A}$ outputs a pair of messages (m_0, m_1) and an index i. $\mathcal{C}$ checks that i is not corrupted, and samples a random bit $b \leftarrow\$ \{0, 1\}$. $\mathcal{C}$ encrypts m_b using public key pk_e^i, $\mathcal{A}$ wins if it guesses the value of b. Indistinguishability of $\mathcal{H}_3$ and $\mathcal{H}_2$ can be proven via a reduction to this extended IND-CPA game through a sequence of hybrids in which one ciphertext at a time is modified to encrypt $\perp$.

$\mathcal{H}_4$**:** This is equivalent to $\mathcal{H}_3$ except that $\mathcal{B}$ does not sample b at random but fixes $b = 0$. If $b = 0$ also in $\mathcal{H}_3$, the hybrids are identical. Let us assume that $b = 1$ in $\mathcal{H}_3$. Let $\Sigma^{\mathcal{H}_j}$ be the challenge list given to $\mathcal{A}$ at the end of $\mathcal{H}_j$, with $j \in \{3, 4\}$. Let us denote the i-th element of $\Sigma^{\mathcal{H}_j}$ as $\sigma_i^{\mathcal{H}_j} = (k, \Pi_i^{\mathcal{H}_j}, \mathsf{A}_i^{\mathcal{H}_j})$, where $i \in [l]$ and l is the length of the ladders. We now argue that $\mathcal{A}$ cannot distinguish $\mathcal{H}_4$ from $\mathcal{H}_3$. First, since the ladders are well-formed all the $\sigma_i^{\mathcal{H}_3}$ and $\sigma_i^{\mathcal{H}_4}$ contain an ENIWI w.r.t. the same statement. Let us assume $\mathcal{A}$ corrupted all the keys that are not in the ladders[16]. Therefore, $\mathcal{A}$ gets to see the corresponding auxiliary values. As a result, an $\mathcal{A}$ distinguishing between $\mathcal{H}_4$ and $\mathcal{H}_3$ would get access to two ENIWI with auxiliary values associated to keys whose corresponding witnesses were not used in any of the two ENIWI. In addition to that, $\mathcal{A}$ just gets encryptions of values that are uncorrelated with the two ENIWI and the auxiliary values. It follows that we can directly reduce the indistinguishability between $\mathcal{H}_4$ and $\mathcal{H}_3$ to the security game of extended witness indistinguishability (cfr., Def. 9).

A.7 Proof of Thm. 2

Lemma 5. *The proof system described in Sec. 7.3 enjoys completeness and transformation completeness.*

Proof. Due to the completeness of GS, for honestly generated proofs the output of $\mathsf{PrVerify}$ is always 1. Let us prove that the same holds for $\mathsf{AuxVerify}$. Let us consider a commitment to a group element $\hat{x}$ as $\hat{c} = \boldsymbol{e}\hat{x} + \hat{\boldsymbol{v}}r + \hat{\boldsymbol{w}}s$ with $r, s \leftarrow\$ \mathbb{Z}_p$, with corresponding auxiliary value $\mathsf{aux} = (\mathsf{aux}^1 = r\check{h}, \mathsf{aux}^2 = s\check{h})$. The check performed by $\mathsf{AuxVerify}$ is the following, where $\hat{x} \in \{\hat{g}, \hat{0}\}$ depends on the commitment and the auxiliary value to be checked.

$$\hat{c} \cdot (\check{h}, \check{h}) + \hat{\boldsymbol{v}} \cdot (-\mathsf{aux}^1, -\mathsf{aux}^1) + \hat{\boldsymbol{w}} \cdot (-\mathsf{aux}^2, -\mathsf{aux}^2) = (\hat{0}, \hat{x})^\top \cdot (\check{h}, \check{h})$$

Let us consider the left side of the above equation, we have:

$$\begin{aligned}
&[(\hat{0}, \hat{x}) + (r\xi\hat{g}, r\hat{g})^\top + (s\rho\xi\hat{g}, s\rho\hat{g})^\top] \cdot (\check{h}, \check{h}) + \\
&(\xi\hat{g}, \hat{g})^\top \cdot (-r\check{h}, -r\check{h}) + (\rho\xi\hat{g}, \rho\hat{g})^\top \cdot (-s\check{h}, -s\check{h}) = \\
&(\hat{0}, \hat{x})^\top \cdot (\check{h}, \check{h})
\end{aligned}$$

By observing that RandPr and $\mathsf{AuxUpdate}$ shift the commitments and the auxiliary values of the same randomness, it is straightforward to see that the first requirement of transformation completeness is also satisfied. Similarly, regarding the second requirement, PrAdd just replaces a commitment and corresponding auxiliary value with new ones that still pass the check performed by $\mathsf{AuxVerify}$. The same holds for the third requirement, where $\mathsf{PrExtend}$ just adds a new commitment with corresponding auxiliary value.

[16] These are the only keys $\mathcal{A}$ can corrupt while still being admissible.

Lemma 6. *The proof system described in Sec. 7.3 enjoys re-randomizable addition. (Def. 7).*

Proof. Let us point out the differences between the experiment executions with $b = 0$ and $b = 1$. In both cases, RandPr and $\mathsf{AuxUpdate}$ are executed to update both the proof and the auxiliary values before handling them to $\mathcal{A}$. Let us now consider what happens to the elements of Π and AUX before the above steps. When $b = 0$ both Π and AUX are freshly computed using the witness provided by $\mathcal{A}$. When $b = 1$ all proofs and commitments are just taken from the proof Π^* (provided by $\mathcal{A}$), excluding the ones related to variables b_{α_k}, $\hat{M}_{\alpha_k}$, x_{α_k} $y_{\alpha_k,j}$,for $j \in [J_{\alpha_k}]$, which are freshly generated.

Let us now focus on commitments $\hat{c}_{\hat{M}_i}$ to variables $\hat{M}_i$, with $i \in [n]$, contained in proof Π^*. It is straightforward to observe that after running $\mathsf{GS.RandPr}$, we get new commitments that are equally distributed to randomly chosen commitments to $\hat{M}_i$. The list AUX^* contains group elements satisfying the verification equations checked by $\mathsf{AuxVerify}$[17]. Auxiliary values are updated as random group elements of $\check{\mathbb{H}}$, using the same randomness previously used to re-randomize the commitments (which are elements of $\hat{\mathbb{G}}$). Due to transformation completeness, re-randomized commitments together with updated auxiliary values still satisfy such equation.

Therefore, the joint distribution of commitments and auxiliary values after re-randomization and update is equally distributed to randomly chosen commitments to $\hat{M}_i$ along with corresponding auxiliary values satisfying the verification equations checked by $\mathsf{AuxVerify}$.

Finally, as already shown in previous works [23,6,13], re-randomized proof elements (i.e, π values) are distributed as randomly chosen proof elements from the space of all valid proof elements, given that the commitments to the involved variables are fixed. Since the commitments are fully re-randomized, the result is a randomly chosen proof given a fixed solution, along with openings of the randomly chosen commitments contained in the proof. This means that, for all equations involving any of the variables $\hat{M}_i$ with $i \in [n]$, the joint distribution of commitments, proofs elements, and AUX is identically distributed both when $b = 0$ and $b = 1$[18]. Finally, for all the equations not involving any of the variables $\hat{M}_i$ with $i \in [n]$, it suffices to reduce to the re-randomizability property of GS.

Lemma 7. *The proof system described in Sec. 7.3 enjoys re-randomizable extension (Def. 8).*

Proof. The proof basically mirrors the one of re-randomizable addition.

Lemma 8. *The proof system described in Sec. 7.3 enjoys extended witness indistinguishability (Def. 9).*

[17] We are guaranteed of that since otherwise $\mathcal{A}$ would not be admissible.

[18] In particular, note that after the invocation of PrAdd, all the proofs are accepting, including the one of equation $\mathcal{K}$.

Proof. Notice that the proof $\mathit{\Pi}$ contains a tuple of commitments and proof elements for each statement x_i with $i \in [n]$ (i.e., $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}, \boldsymbol{C}_{\boldsymbol{\mathcal{M}}i}, \boldsymbol{C}_{\boldsymbol{\mathcal{Y}}i}, \boldsymbol{C}_{\boldsymbol{\mathcal{X}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}i})$). Let us now consider the EWI experiment, where $\mathcal{A}$ chooses a statement $x = (k, x_1, \ldots, x_n)$ and two witnesses w_0 and w_1 such that $(x, w_0) \in R_t$ and $(x, w_1) \in R_t$. Let $(\mathit{\Pi}_b, \mathsf{AUX}_b)$ be the output of the challenger conditioned on the value of b. We argue that $(\mathit{\Pi}_0, \mathsf{AUX}_0)$ is indistinguishable from $(\mathit{\Pi}_1, \mathsf{AUX}_1)$.

For each of the tuples of the proof $\mathit{\Pi}_b$, we have that either $i \notin S$ or $i \in S$.

If $i \notin S$, AUX does not contain any information regarding the i-th tuple[19]. In this case, the i-th tuples when $b = 0$ and $b = 1$ are indistinguishable thanks to the WI of the underlying GS proof system.

If $i \in S$, we can distinguish two additional cases: (i) i is the index of a witness used both in w_0 and in w_1, and (ii) i is an index of a statement not used in both w_0 and w_1.

Let $i \in S$ be an index used both in w_0 and w_1. In this case, the value in $\hat{M}_i$ is $\hat{g}$. $\mathcal{A}$ obtained the auxiliary value associated with $\hat{M}_i$. We now consider all the elements of the i-th tuple and we argue that they are indistinguishable in both hybrids:

1. $(\boldsymbol{C}_{\boldsymbol{\mathcal{B}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{B}}i})$ are indistinguishable since $b_i = 1$ in both hybrids.
2. $(\boldsymbol{C}_{\boldsymbol{\mathcal{M}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{M}}i})$ are indistinguishable since $\hat{M}_i = \hat{g}$, $b_i = 1$ in both hybrids;
3. Variables $\check{y}_{i,q,j}, \check{x}_{i,q,j}$ may differ in the two hybrids. However, $\mathcal{A}$ cannot distinguish $(\boldsymbol{C}_{\boldsymbol{\mathcal{Y}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{Y}}i})$ in the two hybrids. Indeed, the distribution of the proof elements is uniform over all possible terms satisfying the equation conditioned on the commitments, and all the commitments and auxiliary values are equally distributed in both hybrids.
4. Finally, $(\boldsymbol{C}_{\boldsymbol{\mathcal{X}}i}, \boldsymbol{\pi}_{\boldsymbol{\mathcal{X}}i})$ do not involve any $\hat{M}_i$, and thus they are not related to AUX. They are indistinguishable thanks to the WI of the underlying GS proof system.

Whenever $i \in S$ is not used in both w_0 and w_1, we have that $\hat{M}_i = \hat{0}$ and AUX will contain openings to random commitments to $\hat{0}$. We notice that in the two hybrids, the entire i-th tuple is computed running the GS prover on the same inputs. Therefore, they are identically distributed.

There are two last element to take into account, namely $C_{\mathcal{K}}$ and $\pi_{\mathcal{K}}$. These are indistinguishable in both hybrids for the reasons previously argued in 3.

Lemma 9. *The proof system described in Sec. 7.3 enjoys adaptive extractable soundness (Def. 14).*

Proof. The algorithm Ext_1 just runs $\mathsf{GS.Ext}_1$. The algorithm Ext_2 runs $\mathsf{GS.Ext}_2$ for the equation $\mathcal{K}$. Due to the soundness of GS proofs, and thanks to the proofs of equations of type $\mathcal{M}$ and $\mathcal{B}$, we are guaranteed to extract, except with negligible probability, k values $\hat{M}_i = \hat{g}$. Let $A = \{\alpha_1, \ldots, \alpha_k : \hat{M}_{\alpha_1} = \ldots = \hat{M}_{\alpha_k} = \hat{g}\}$. Ext_2 runs $\mathsf{GS.Ext}_2$ for the k equations $\mathcal{X}_{\alpha_i}$ with $\alpha_i \in A$. Let w_i be the witness

[19] Notice that the randomness used in different commitments is always uniformly sampled.

extracted from the proofs for equation $\mathcal{X}_{\alpha_i}$, with $\alpha_i \in A$. Due to the soundness of GS proofs, and thanks to the proofs of equations of type $\mathcal{Y}$ it holds that equations $\mathcal{X}_{\alpha_i}$ are the same as statement x_{α_i}, with $\alpha_i \in A$. Therefore, $w = \{(w_1, \alpha_1), \ldots, (w_k, \alpha_k)\}$ is such that $((k, x_1, \ldots, x_n), w) \in R_t$.