Efficient Proofs of Knowledge for Threshold Relations

Gennaro Avitabile*[1], Vincenzo Botta†[1], Daniele Friolo‡[2], and Ivan Visconti§[1]

[1]DIEM, University of Salerno
[2]Sapienza University of Rome

Abstract

Recently, there has been great interest towards constructing efficient zero-knowledge proofs for practical languages. In this work, we focus on proofs for threshold relations, in which the prover is required to prove knowledge of witnesses for k out of ℓ statements.

The main contribution of our work is an efficient and modular transformation that starting from a large class of Σ-protocols and a corresponding threshold relation $\mathcal{R}_{\mathsf{k},\ell}$, provides an efficient Σ-protocol for $\mathcal{R}_{\mathsf{k},\ell}$ with improved communication complexity w.r.t. prior results. Moreover, our transformation preserves statistical/perfect honest-verifier zero knowledge.

*gavitabile@unisa.it
†vbotta@unisa.it
‡friolo@di.uniroma1.it
§visconti@unisa.it

Contents

1 Introduction

With the advent of blockchain technology and cryptocurrencies there is much more interest towards designing practical systems for decentralized computations. In particular there is an effort towards systems producing succinct messages that can therefore be uploaded on blockchains guaranteeing some public verifiability. Notable examples of such tools are threshold signatures and succinct non-interactive arguments of knowledge (SNARKs).

Proofs over threshold relations. In this work we are interested in efficient proofs over threshold relations (PTRs) where a statement consists of n instances and the prover would like to prove knowledge of witnesses for at least k of them. There has been an effort in the past to obtain such proofs for practical languages.

In [CDS94] Cramer et al. showed how to efficiently combine Σ-protocols in order to prove knowledge of witnesses corresponding to at least k out of ℓ instances. For simplicity we will refer to such a proof as a (k, ℓ)-PTR. Their construction mainly consists of running Σ-protocols for all instances, combining them efficiently and thus the costs (i.e., computations and communication) of their (k, ℓ)-PTRs essentially consist of the sum of the costs of all underlying Σ-protocols. The resulting protocol is still a Σ-protocol. Interestingly, their composition enjoys two more properties: it allows also the use of non-threshold access structures; the ℓ instances can belong to different languages since the starting point is the direct use of Σ-protocols for the involved languages.

More recently, a different technique has been proposed in [CPS$^+$16] where Ciampi et al. showed how to obtain a similar result with the additional feature of postponing the need to know the instances to the last round (i.e., delayed input). The delayed-input (k, ℓ)-PTR of Ciampi et al. relies on the DDH assumption and can be used for instances of multiple languages since they allow the use of a large class of Σ-protocols. The resulting protocol is a 3-round public-coin proof of knowledge. Unfortunately, since this composition technique relies on a computationally-hiding commitment scheme it produces a protocol which only achieves computational zero knowledge regardless of the underlying Σ-protocols being statistical/perfect zero knowledge. However, statistical/perfect zero knowledge is very important since it protects the privacy of the prover forever (e.g., even if quantum computers become a concrete threat).

Very recently, Attema et al. in [ACF21], improving a prior work of Groth and Kohlweiss [GK15], have shown how to obtain a very compact (k, ℓ)-PTR that however works only for discrete logarithms (and variations) thus remaining far from the general results of [CDS94]. The resulting construction requires a logarithmic number of rounds[1] and is secure against polynomial-time adversarial provers only (while preserving statistical/perfect zero knowledge). Moreover, they require as trusted parameters a shared random string (SRS).

Even more recently, Goel et al. in [GGHK21] have broken the barrier of linear (in ℓ) communication complexity when composing generic Σ-protocols, showing an efficient composition for a large class of Σ-protocols (which they call stackable Σ-protocols) obtaining logarithmic communication complexity. Their construction is secure against polynomial-time adversarial provers only[2] obtaining computational special soundness. They give an instantiation of their construction based on a commitment scheme that relies on the discrete logarithm assumption and requires trusted parameters including the description of a collision-resistant hash function (CRHF) and parameters for Pedersen commitments. The perfect hiding of the commitment scheme allows to preserve statistical/perfect zero knowledge. In Sec. 8 of their work they also show techniques to combine instances of different languages. While their construction applies to a large class of Σ-protocols, unfortunately, the techniques of [GGHK21] are communication efficient only when $k = 1$[3]. Goel et al. discuss (see Sec. 9.1 and App. F of [GGHK21]) an approach for the case of $k > 1$ but unfortunately, as they admit, their proposal strongly affects communication, without providing substantial

[1]The result of [GK15] instead works only for $k = 1$ but in 3 rounds.

[2]For ease of presentation, in this work we will use the term PPK even when the soundness property holds only against a computationally bounded adversarial prover. We will do the same for computational Σ-protocols which only satisfy a weaker version of special soundness called computational special soundness (cfr., Sec. 3.2).

[3]Note that the protocol for the case $k = 1$ appeared in Eurocrypt 2022 [GGHK22]. The protocol for $k > 1$ was only described in the pre-print version [GGHK21].

improvements over [CDS94]. Goel et al. left explicitly open the problem of efficiently combining Σ-protocols in order to break, for generic values of k, the linear (in ℓ) barrier achieved by [CDS94] (see [GGHK21], page 32, Sec. 9.1).

Open problem. In light of the above state of affairs, we have the following natural and interesting (both theoretically and practically) open question:

Is it possible to obtain practical (i.e., round efficient, communication efficient and computationally efficient) (k, ℓ)-proofs of knowledge for threshold relations for a large class of Σ-protocols (and thus for several useful languages) with communication complexity $o(\ell)$ and which also preserve statistical/perfect zero knowledge?

1.1 Our Contribution

In this work we solve the above open problem when $k = o(\frac{\ell}{\log \ell})$ by showing how to efficiently combine the same large class of Σ-protocols considered in [GGHK21] obtaining a (k, ℓ)-PTR with communication complexity that is roughly[4] $k \log \ell$. In scenarios where k is way smaller than ℓ (e.g., k is constant or even $\sqrt{\ell}$) this is a big improvement. Moreover, our construction, similarly to [GGHK21], can also be used for (k, ℓ)-PTR involving Σ-protocols for different languages. The protocol obtained through our techniques is still a Σ-protocol. As a result, it can be combined again with our techniques or other techniques (e.g., [CDS94]) for composing Σ-protocols, thus allowing non-threshold access structures. Finally our construction preserves the flavour of the zero knowledge property of the composed protocols. Indeed, our (k, ℓ)-PTR is still perfect honest-verifier zero knowledge if the base Σ-protocols are perfect honest-verifier zero knowledge.

We use the $(1, \ell)$-PTR of [GGHK21] as a building block and start with their observation that repeating k times their construction is insecure since an adversarial prover might succeed using a witness for the same instance in all the k executions. This is precisely the problem left unsolved for (k, ℓ)-PTRs with sublinear communication in [GGHK21] that instead we solve in this work.

Compact proof of consistency of commitment parameters. In [GGHK21], the use of a witness is associated to $\log \ell$ pairs of parameters of a commitment scheme such that for every pair one parameter allows for equivocation and the other parameter prevents equivocation. For simplicity, we will say that one parameter is equivocal and the other one is binding and only the prover knows which element is equivocal for every pair. Informally, we say that[5] a commitment scheme is 1-out-of-2 equivocal when a commitment phase requires to commit to two messages, one with binding parameters and one with equivocal parameters, without allowing the receiver to distinguish them, even after the commitment is opened.

In the following, we assume w.l.o.g. that ℓ is a power of 2 and give an abstract/simplified description of our approach, while a more detailed and rigorous explanation will be given in Sec. 6.2. Let $x_0, \ldots, x_{\ell-1}$ be the instances and consider x_i for $i \in \{0, \ldots, \ell-1\}$ be the instance corresponding to witness w_i known to the prover. The $\log \ell$ pairs of parameters are chosen so that the j-th pair has the first parameter binding if the j-th bit of i is zero (for $j = 0, \ldots, \log \ell - 1$) and equivocal otherwise. Notice that the connection between a pair of parameters that can be either (equivocal,binding) or (binding,equivocal) and a bit of the index of an instance makes the $\log \ell$ pairs of parameters associated to x_i logically different from the $\log \ell$ pairs of parameters associated to x_j, as long as $i \neq j$.

We observe that in order to show that in k executions of the $(1, \ell)$-PTR of [GGHK21] the k witnesses correspond to k different instances, one can focus on showing that the k sequences of $\log \ell$ pairs of parameters are all disjoint in the sense that for every pair $(\bar{a} = a_0, a_1, \ldots, a_{\log \ell-1}), (\bar{b} = b_0, b_1, \ldots, b_{\log \ell-1})$ of elements in those k sequences, there is always a position $j \in \{0, \ldots, \log \ell - 1\}$ such that only one out of a_j and b_j has the first parameter that is binding. We focus on efficiently proving the above property of all pairs in those k sequences as follows: first, we require the prover to sort the k sequences according to the order relation

[4]We will be more precise later making the impact of the security parameter explicit.

[5]We use the same term originally used in [CPS+16], while the term 1-out-of-2 Partially Binding Vector Commitment is instead used in [GGHK21].

derived by assigning to every sequence of $\log \ell$ pairs of parameters a string of $\log \ell$ bits where the j-th bit is 0 if and only if the j-th pair of parameters in the sequence has the first element that is binding. The prover has all the information to sort those k sequences since the prover decided those parameters and thus it knows which one is binding and which one is equivocal. Once the k sequences are sorted, in order to prove that they are all disjoint (in the sense explained above) for the prover it is enough to show that for every two consecutive elements $(\bar{a}, \bar{b})$ in such ordered sequence of k elements, the bit representation of $\bar{b}$ is greater than the one of $\bar{a}$. With such trick, the prover must provide $k-1$ proofs in total to show that all sequences are different. Each of such proofs is about proving a property of the involved $4 \log \ell$ parameters[6]. We show a concrete and efficient instantiation of such proofs with communication $\mathcal{O}(\log \ell)$ for the parameters of the 1-out-of-2 equivocal commitment scheme from [GGHK21]. Our construction can also be instantiated, with small modifications, using the commitment scheme of [CPS$^+$16][7]. In this case, our (k, ℓ)-PTR would provide only computational honest-verifier zero knowledge and it would only require a collision-resistant hash function as setup.

In Table 1, we compare our results to the previously discussed approaches to obtain (k, ℓ)-PTR. For the sake of completeness, in this comparison we make the security parameter λ explicit. Note that our approach while being less communication efficient than [GK15, ACF21] is more flexible since it applies to a much wider family of languages.

An additional discussion on related work in comparison with our results can be found in Sec. 2.

Non-threshold access structures. In [CDS94], Cramer et al. provided a generalization of their technique for arbitrary monotone access structures other than the threshold one. Attema et al. in [ACF21] also appropriately modify their protocol to enable monotone access structures for their argument system. Interestingly, our (k, ℓ)-PTR is a general compiler that can take other (k', ℓ')-PTRs as input to create a PTR for threshold-of-threshold relations.

Threshold ring signatures. As pointed out in [GGHK21], by making their $(1, \ell)$-PTR non-interactive with the aid of a random oracle one gets a ring signature whose size is logarithmic in the size of the ring ℓ (See [GGHK21] Page 4 and Sec. 9.3). Following a similar approach, starting from our (k, ℓ)-PTR we can get a threshold ring signature scheme according to Def. 3 of [HS20] but considering PPT adversaries instead of quantum polynomial-time adversaries. In a threshold ring signature scheme, k signers cooperate to sign a message hiding their identities within a larger group of size ℓ. In our threshold ring signature scheme the size of a signature corresponds roughly to $\mathcal{O}(k \log \ell)$ group elements (e.g., consider the instantiation using the protocol of [Sch89] as underlying Σ-protocol). Interestingly, while featuring a relatively simple design, our construction surpasses in terms of signature tag size many literature works [BSS02, YLA$^+$13, OTYO18]. Other schemes have signature size which is linear in ℓ (while being independent of k) and thus are also outperformed when $k << \ell$ [ZZY$^+$17, CHGa19, HS20]. When comparing our construction to others achieving more compact signature sizes [HS20, MOY21, ACR21], our construction still has interesting advantages in terms of resilience to adversarially chosen keys (i.e., given two sets of honestly generated keys S_0 and S_1 of size k on the same ring of size ℓ the adversary cannot tell if a signature is generated using the signing keys in S_0 or in S_1, even if any subset of all the keys of the ring but the ones in S_0 and S_1 is maliciously generated, see [HS20]) or used assumptions. A major feature of our threshold ring signature is that it can be instantiated from a variety of assumptions (i.e., the assumptions depend on the chosen languages and Σ-protocols). A more detailed comparison is reported in Sec. 2.

Finally, our techniques allow threshold ring signatures with more advanced hiding properties than a simple subset of signers. Indeed, we can consider also non-threshold access structures obtaining more expressive power (i.e., better anonymity) than a regular (k, ℓ) threshold.

[6]For each proof there are two sequences of $\log \ell$ pairs of parameters.

[7]This modification would mainly consists in adapting the definitions of 1-out-of-2 equivocal commitment scheme to 3-round public coin protocols as the one of [CPS$^+$16]. Additionally, our proof system for proving an ordering relation among commitment parameters would follow the same structure but would take into account the different nature of the commitment parameters. Indeed, in this case equivocal parameters are Diffie-Hellman tuples and binding parameters are a particular kind of non-Diffie-Hellman tuples (i.e., 1-non-Diffie-Hellman tuples).

Protocol	**# Rounds**	**Communication**	**Values of k**	**Language**
[CDS94]	3	$\mathcal{O}(\ell\mathsf{CC}(\Sigma))$	$k > 0$	All Σ
[GK15]	3	$\mathcal{O}(\lambda \log \ell)$	$k = 1$	DL-like
[ACF21]	$\mathcal{O}(\log \ell)$	$\mathcal{O}(\lambda \log(2\ell - k))$	$k > 0$	DL-like
[GGHK21]	3	$\mathcal{O}(\mathsf{CC}(\Sigma) + \lambda \log \ell)$	$k = 1$	Stackable Σ
[GGHK21]	3	$\mathcal{O}(k(\mathsf{CC}(\Sigma) + \lambda\ell))$	$k > 1$	Stackable Σ
Ours	3	$\mathcal{O}(k(\mathsf{CC}(\Sigma) + \lambda \log \ell))$	$k > 0$	Stackable Σ

Table 1: Comparison of several techniques for (k, ℓ)-PTR. When comparing more language-generic techniques like ours and [CDS94, GGHK21] we express the communication complexity both in terms of the the communication complexity of the underlying Σ-protocol $\mathsf{CC}(\Sigma)$ and of the security parameter λ. Note that the communication complexity of [GK15] does not depend on k since their technique only works for $k = 1$. The Language column reports the languages supported by the corresponding composition technique. Despite being the least communication efficient, [CDS94] supports a wider class of languages (i.e., all languages admitting a Σ-protocol).

Remark on [GGHK21]. We have also noticed some subtleties in the security analysis of the main claim of [GGHK21] that we discuss in Remark 2 and Remark 3. For completeness, we have added in the appendix an instantiation (along with a proof) of [GGHK21] with the 1-out-of-2 commitment scheme defined as in Sec. 4 taking care of those subtleties.

1.2 Technical Overview of [GGHK21]

We first describe 1-out-of-2 equivocal commitments (see Sec. 4 for more details) that are a major tool used in [GGHK21]. Then, we show how [GGHK21] exploits 1-out-of-2 equivocal commitments to get a $(1, \ell)$-PTR.

1-out-of-2 equivocal commitments in a nutshell. A 1-out-of-2 equivocal commitment allows a sender to commit to two values one of which is guaranteed to be binding, either unconditionally or under computational assumptions. The other element instead can be equivocated using a trapdoor that is known to the sender. Once the commitment is opened, the commitment scheme itself would guarantee the equivocal position is not leaked. Before sending the commitment to the receiver, the commitment scheme parameters are generated by the sender who has to decide which position is equivocal. From now on, we call non-trapdoor (NT) a parameter that is associated with a binding position, while a trapdoor parameter (T) is associated with an equivocal position. We will use the word "trapdoorness" when referring to the property of a commitment parameter of allowing to equivocate or not.

$(1, \ell)$**-PTR through Σ-protocols.** For simplicity, we will focus on instances belonging to the same language. Nevertheless, both in [GGHK21] and in our results it is possible to go beyond this restriction (see Sec. 8 of [GGHK21]).

The main idea in [GGHK21] is that every involved Σ-protocol has a deterministic Honest Verifier Zero-Knowledge (HVZK) simulator, called Extended HVZK (EHVZK) simulator which, given a challenge c, a third-round message z, and a statement x, successfully outputs a simulated a such that (a, c, z) is an accepting transcript for the instance x. With EHVZK in their hands, the authors introduce the notion of stackable Σ-protocols. A Σ-protocol is stackable if (i) it has an EHVZK simulator and (ii) the third-round message is recyclable, meaning that the distribution of such messages is independent of the instance for every instance in the language.

Let us first consider just two of the ℓ instances, say x_1 and x_2. Given two executions Σ_1 and Σ_2 of a stackable Σ-protocol Π for instances x_1 and x_2 respectively, an execution $\Sigma_{1,2}$ of the combined protocol Π' defined by Goel et al. [GGHK21] for $x_1 \vee x_2$ can be constructed as follows. Let us assume that the prover $\mathsf{P}_{1,2}$ knows the witness corresponding to x_1. We name a_1 (respectively a_2) the first-round message of the underlying execution Σ_1 (respectively Σ_2), a the first-round message of the execution $\Sigma_{1,2}$ of Π', c

the challenge sampled by the verifier $\mathsf{V}_{1,2}$ for Π', and z the last message of $\Sigma_{1,2}$. Since Σ_1 and Σ_2 are executions of the stackable Σ-protocol Π, their third-round messages have the same distribution. Therefore, the accepting third-round message from the execution Σ_1 can be re-used as a third-round message for the execution Σ_2 as described in the composed Σ-protocol Π' below:

- $\mathsf{P}_{1,2}$ computes the first-round message a_1 of protocol Π on input the statement x_1 and witness w_1. $\mathsf{P}_{1,2}$ commits to a_1 using a 1-out-of-2 equivocal commitment scheme. The value a_1 is put in the binding position, while the equivocal position commits to 0. We denote the resulting commitment as $\mathtt{com}$. The first-round message a in the execution $\Sigma_{1,2}$ of the composed protocol Π' includes $\mathtt{com}$ as well as the parameters of the commitment scheme.

- Upon receiving the challenge c from $\mathsf{V}_{1,2}$, $\mathsf{P}_{1,2}$ computes z' using witness w_1, and equivocates the equivocal position of the commitment with a simulated a_2. The value a_2 is obtained by running the EHVZK simulator of Π with input the instance x_2, c, and the value z' computed above. Then $\mathsf{P}_{1,2}$ sends z' and the opening values of $\mathtt{com}$ to $\mathsf{V}_{1,2}$ as a third-round message z of $\Sigma_{1,2}$. The value z also includes the commitment parameters[8].

- $\mathsf{V}_{1,2}$ reconstructs a_1 and a_2 by running the EHVZK simulator of Π. Then $\mathsf{V}_{1,2}$ checks that both (a_1, c, z') and (a_2, c, z') are accepting transcripts for V_1 and V_2, and that $\mathtt{com}$ actually opens to a_1 and a_2.

Since Π' is still a stackable Σ-protocol, it can be recursively used to prove the statement $x_1 \vee x_2 \vee x_3 \vee x_4$. Indeed, this can be seen again as an OR of two statements, therefore the Σ-protocol for the statement $(x_1 \vee x_2) \vee (x_3 \vee x_4)$ can be composed using the same technique. Then, one can iterate the same process to obtain a $(1, 8)$-PTR by applying the same technique to two $(1, 4)$-PTR, and so on.

Such composition of ℓ disjunctive statements can be represented by the following binary tree[9]: the leaves of the tree represent the ℓ base executions $(\Sigma_1, \ldots, \Sigma_\ell)$ of the Σ-protocol Π. Given two siblings nodes i and j, with associated protocol execution Σ_i for the instance x_i and Σ_j for the instance x_j respectively, the parent node t of i and j describes the execution of the protocol Σ_t obtained by applying the compiler for $(1, 2)$-PTR of [GGHK21]. Moreover, the edges (t, i) and (t, j) are labeled as follows: if P_t knows a witness for the statement x_i, then the edge (t, i) is labeled with NT to indicate that, in the commitment computed by P_t in the first round the position where x_i is used is binding. The edge (t, j) is labeled with T to indicate that the position where x_j is used is equivocal. If instead P_t knows a witness for x_j, then the opposite holds. An example of a tree induced by recursively applying the composition of [GGHK21] for a $(1, 2)$-PTR to get a $(1, 8)$-PTR is shown in Fig. 1. As explained in Sec. 1.3, this recursive application of the (1,2)-PTR gives a communication complexity for the $(1, \ell)$-PTR which is roughly logarithmic in ℓ.

(k, ℓ)-PTR extension. In [GGHK21] an extension of their compiler to achieve a (k, ℓ)-PTR[10] is also proposed. An immediate solution might consist in just repeating the $(1, \ell)$-PTR k times. Unfortunately, such simple approach does not work because the prover could use the same witness in each of the k executions of the $(1, \ell)$-PTR. In [GGHK21] it is proposed to address the above issue modifying their 1-out-of-ℓ equivocal commitment scheme. They propose as replacement a k-out-of-ℓ binding vector-of-vectors commitment scheme. This modification allows to equivocate at most $\ell - k$ positions. Roughly speaking, they instantiate such a primitive by making the commit algorithm output a matrix of $k \times \ell$ commitment values (i.e., each row is a 1-out-of-ℓ equivocal commitment), together with a non-interactive (NI) zero-knowledge (ZK) proof that the binding position is different in each row. As pointed out in [GGHK21], with this technique they lose the ability to recursively apply the $(1, 2)$-PTR compiler. As a result, their (k, ℓ)-PTR has a communication complexity of roughly $\mathcal{O}(k\ell)$.

[8]Later on it will be more clear why this allows to exploit extended simulation to achieve logarithmic communication complexity.

[9]Without loss of generality it can be assumed that the number of composed instances is always a power of two.

[10]Sec. 9.1 and App. F of [GGHK21].

1.3 Our Techniques

Our approach for a communication-efficient (k, ℓ)-PTR. In [GGHK21], the location of the instance for which the prover knows the witness uniquely determines the way parameters are laid out over the composition tree. For example, in Fig. 1 the instance for which $\mathsf{P}_{1,2}$ knows the witness is x_1. This means that, starting from $\Sigma_{1,2}$, the position containing the first message of Σ_1 has to be binding. Indeed, since $\mathsf{P}_{1,2}$ only holds a witness for x_1, $\mathsf{P}_{1,2}$ is able to produce an accepting transcript exclusively for Σ_1. Therefore, the third-round message to be recycled has to come from Σ_1, while the committed first-round message of the execution Σ_2 needs to be equivocated with the output of the EHVZK simulator. It follows that, climbing up the tree, the commitment position containing the first message of $\Sigma_{1,2}$ has to be binding. Indeed, $\mathsf{V}_{1,2,3,4}$ will in turn execute $\mathsf{V}_{1,2}$ and $\mathsf{V}_{3,4}$, which internally use the verifiers of the base Σ-protocols. This means that in $\Sigma_{3,4}$ the prover recycles the third-round message of $\Sigma_{1,2}$ and that in $\Sigma_{1,2,3,4}$ the committed first-round message of $\Sigma_{3,4}$ has to be equivocated accordingly in order to get an accepting transcript. Applying the same reasoning again, it is easy to conclude that in $\Sigma_{1,\ldots,8}$ the binding position of the 1-out-of-2 equivocal commitment is again the same.

A crucial idea of [GGHK21] to achieve logarithmic communication complexity is reusing commitment parameters and openings across the same levels of the composition tree. The composition is designed so that commitment parameters and openings are part of the third-round message of the composed protocol. Indeed, since the composed Σ-protocol of [GGHK21] is itself stackable, it follows that its EHVZK simulator takes as input commitment parameters and openings to generate a suitable first-round message, namely a 1-out-of-2 equivocal commitment reusing the same openings and parameters[11]. This means that since all the Σ-protocols executions that belong to the same level of the tree share the same third-round message they also have to use the exact same commitment parameters. This is emphasized in Fig. 1 by coloring all the edges of each level with the same color. Therefore, in the $(1, \ell)$-PTR of [GGHK21] depending on the instance x_i corresponding to the witness used by the prover there is a different way commitment parameters are laid out over the composition tree. Thus, to build a (k, ℓ)-PTR it suffices to repeat the construction of [GGHK21] k times and to prove that the composition trees of such k executions are all different. In this section we describe how we design a communication-efficient computational sound Σ-protocol for the above statement.

Problem statement. We use the following notation: for any vector $\mathbf{v}$, $\mathbf{v}[z]$ indicates the z-th element of the vector $\mathbf{v}$. The first element of a vector $\mathbf{v}$ is indexed as $\mathbf{v}[1]$. Moreover, we use $[n]$ for $n \in \mathbb{N}$ to identify the set $\{1, \ldots, n\}$.

Let $\mathbf{x} = ((p_0^1, p_1^1), \ldots, (p_0^n, p_1^n))$ be a vector containing n pairs of parameters corresponding to n instantiations of a 1-out-of-2 equivocal commitment scheme, where p_0^i represents the parameters of the first position of the i-th commitment instantiation, and p_1^i is the analogue for the second position. Consider the relations $\mathcal{R}_{T_0}, \mathcal{R}_{T_1}$, where $\mathcal{R}_{T_0} = \{(x = (p_0, p_1), w) : p_1 \text{ is a trapdoor parameter and } w \text{ is the corresponding trapdoor}\}$ and, similarly $\mathcal{R}_{T_1} = \{(x = (p_0, p_1), w) : p_0 \text{ is a trapdoor parameter and } w \text{ is the corresponding trapdoor}\}$.

We present a Σ-protocol Π_{ord}, that, given a vector $X = (\mathbf{x_1}, \ldots, \mathbf{x_k})$ of vectors each containing n pairs of parameters corresponding to n instantiations of a 1-out-of-2 equivocal commitment scheme, allows a prover P to efficiently prove knowledge of a witness for $X \in \mathcal{L}$ where

$$\begin{aligned}\mathcal{L} = \{&X : \exists W = (\mathbf{w_1}, \ldots, \mathbf{w_k}) \text{ such that} \\ &\forall\, i, j \in [k] \text{ with } i \neq j\ \exists z \in [n], \text{such that } b_{i,z} \neq b_{j,z}\}\end{aligned} \tag{1}$$

where $b_{i,z}$ and $b_{j,z} \in \{0, 1\}$ are such that $(\mathbf{x_i}[\mathbf{z}], \mathbf{w_i}[\mathbf{z}]) \in \mathcal{R}_{T_{b_{i,z}}}$ and $(\mathbf{x_j}[\mathbf{z}], \mathbf{w_j}[\mathbf{z}]) \in \mathcal{R}_{T_{b_{j,z}}}$.

[11]For this composition to work and to compress the communication complexity down to logarithmic, [GGHK21] points out that the size of the equivocal commitment must be independent of the size of the committed value. For example, the first message of the execution $\Sigma_{1,2,3,4}$ in Fig. 1 should be a commitment to two 1-out-of-2 equivocal commitments (i.e., the first messages of executions $\Sigma_{1,2}$ and $\Sigma_{3,4}$) which of course would not fit in the input space of the commitment scheme. To solve this issue, committed values have to be compressed down to a constant size with the aid of a collision-resistant hash function.

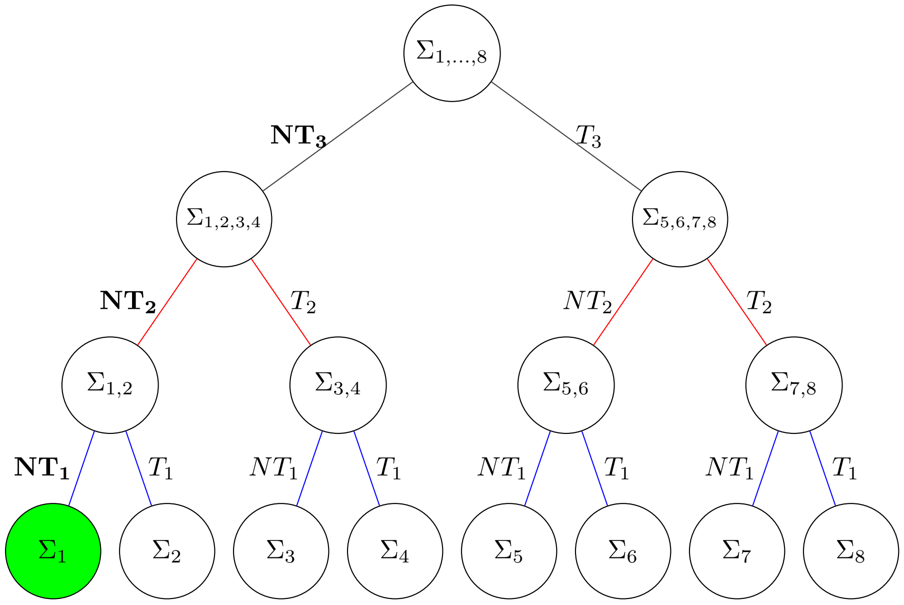

Figure 1: An example of a tree induced by the recursive application of the $(1,2)$-PTR of [GGHK21] in which 8 base Σ-protocols are composed to obtain a $(1,8)$-PTR (i.e., meaning that the prover knows at least one witness w_i for x_i, with $i \in [8]$). In this example, $\mathsf{P}_{1,\ldots,8}$ knows a witness for the statement x_1. This implies that, going from the root to the leaves, the left-most branch must be non-trapdoor. Additionally, commitment openings and parameters are re-used across the same level of the composition tree and this is emphasized by using the same index across all labels and the same color to draw all the edges within a level.

An efficient Σ-protocol. One could naively prove the above statement by separately proving that each vector of n pairs of commitment parameters differs in the way equivocal and binding parameters are laid out w.r.t. every other vector. Carrying out such proof would involve a quadratic (in k) amount of separate proofs, thus impairing the overall efficiency of the Σ-protocol, even for moderate values of k. We instead take a different path, that is introducing a strict total ordering among these k vectors of n pairs of commitment parameters. In particular, we map a vector $\mathbf{x}$ of n pairs of commitment parameters to a binary string $s \in \{0,1\}^n$ by setting $s = b_1||\ldots||b_n$, for which $(\mathbf{x}[\mathbf{z}], \mathbf{w}[\mathbf{z}]) \in \mathcal{R}_{T_{b_z}}$. Let s_m be the string resulting to applying the above mapping to a vector $\mathbf{x_m}$ of n pairs of commitment parameters with $m \in [k]$. W.l.o.g. consider the case where $s_1 > \ldots > s_k$.

If the above order relation holds, it follows that all the k vectors of n pairs of commitment parameters are logically different from each other in terms of how the trapdoor parameters are laid out in at least one position. Note that after having introduced such ordering among the k vectors of n pairs of commitment parameters, one can come up with the language $\mathcal{L}_{ord}$ described by only a linear number of comparisons. Let $X = (\mathbf{x_1}, \ldots, \mathbf{x_k})$ be a vector of k vectors each containing n pairs of parameters corresponding to n instantiations of a 1-out-of-2 commitment scheme, the language of Equation 1 can be equivalently rewritten as

$$\begin{aligned}\mathcal{L}_{ord} =& \{X : \exists W = (\mathbf{w_1}, \ldots, \mathbf{w_k}) \text{ such that} \\ & s_1 > s_2 > \ldots > s_k\},\end{aligned} \tag{2}$$

where for all $m \in [k]$, $s_m = b_1||\ldots||b_n$, for which $(\mathbf{x_m}[\mathbf{i}], \mathbf{w_m}[\mathbf{i}]) \in \mathcal{R}_{T_{b_i}}$.

Instantiation. Let us consider two binary strings $s_1 \in \{0,1\}^n$ and $s_2 \in \{0,1\}^n$, it is pretty straightforward to see that if $s_1 > s_2$, the following formula also holds[12]. We use $s[i]$ to indicate the i-th bit of the string s.

$$\bigvee_{i=1}^{n} \left(\left(\bigwedge_{j=0}^{i-1} (s_1[j] = s_2[j]) \right) \wedge (s_1[i] > s_2[i]) \right). \tag{3}$$

Indeed, this corresponds to performing a bit-wise comparison between s_1 and s_2, starting from the most significant bits. Namely, if $s_1 > s_2$ the first different bit between the two numbers is 1 in s_1 and 0 in s_2.

Building on this observation, we can construct a protocol $\Pi_{ord'}$ to prove that two binary strings, each representing a vector of n 1-out-of-2 equivocal commitment parameters, are such that one is greater than the other. Then, given k vectors of commitment parameters, one can prove that $X = (\mathbf{x_1}, \dots, \mathbf{x_k}) \in \mathcal{L}_{ord}$, where $|\mathbf{x_i}| = n$ for all $i \in [k]$, by using $\Pi_{ord'}$ $k-1$ times. We call the resulting protocol *proof of parameters ordering* Π_{ord}.

1-out-of-2-commitment of [GGHK21]. In [GGHK21] a t-out-of-ℓ equivocal commitment scheme based on the discrete logarithm assumption is defined. From now on, we call this commitment scheme GGHK. Here, we briefly describe GGHK for $t = 1$ and $\ell = 2$. As shown in [GGHK21], this choice of t and ℓ is what is used to get a $(1,\ell)$-PTR from Σ-protocols with logarithmic communication complexity.

GGHK uses the same SRS of the non-interactive version of the Pedersen commitment scheme. Namely, the SRS is composed by two generators g_0, h in a group $\mathbb{G}$ where solving the discrete logarithm is believed to be hard. We recall that a Pedersen commitment to the message $m \in \mathbb{G}$ is computed as $g_0^m h^r$ for a random value $r \in \mathbb{Z}_{|\mathbb{G}|}$. What makes the Pedersen commitment scheme binding is that the sender does not know the discrete logarithm of g_0 with base h. In GGHK, the SRS is used by the sender to compute new generators g_1, g_2 so that the sender knows the discrete logarithm (in base h) of at most one out of g_1 and g_2. To do that, the sender selects a random trapdoor y_1 and derives one of the two generators, say g_1, by computing $g_1 = h^{y_1}$. After having computed g_1,the other generator g_2 is computed by interpolating g_1 and h in the exponent, which is then evaluated in a concrete point in $\mathbb{Z}_{|\mathbb{G}|}$. To be more specific, consider the two points $(0, y_0)$ and $(1, y_1)$, the equation for the line crossing these two points is $y(x) = y_1 x - y_0 x + y_0$ for $x \in \mathbb{Z}_{|\mathbb{G}|}$. Then $g_2 = h^{y(2)}$ is evaluated as $h^{y(2)} = g_1^2 g_0^{-1}$ with $g_1 = h^{y_1}$ and $g_0 = h^{y_0}$. By doing so the intercept of such interpolated line corresponds to y_0, the discrete logarithm of h with base g_0.

To commit to two messages, the sender uses Pedersen commitment scheme with parameters (g_1, h) and (g_2, h), for the first and the second message respectively. To equivocate in the equivocal position, the sender uses the trapdoor associated to the equivocal position as in the regular Pedersen commitment scheme. Intuitively, what makes the other position binding is the fact that by knowing the discrete logarithm of g_2 (i.e., y_2) the sender could use it together with y_1 to reconstruct the equation of the line whose intercept is y_0. Having such equation, it would be trivial to compute the intercept y_0 (i.e., the discrete logarithm of h), thus breaking the discrete logarithm assumption. Whether one of the two parameters is trapdoor or not is perfectly hidden. Indeed, both g_i with $i \in \{1, 2\}$ can be recomputed by interpolating in the exponents the other generator and g_0. Whatever the trapdoor position is, g_0 and g_1 look as two uniformly random group generators.

Instantiating our Σ-protocol for the commitment of [GGHK21]. We now instantiate Π_{ord} for vectors of commitment parameters of GGHK.

Therefore, it suffices for the sender to prove that he knows the trapdoor of of at least of one of the two parameters (i.e, to prove that $(x, w) \in \mathcal{R}_{T_b}$ with $b \in \{0, 1\}$).

To do so, we just need to express Formula 3 in terms of the parameters of GGHK. Given (p_0^a, p_1^a), let $\mathcal{R}_{\mathsf{DL}}(p_b^a, w)$ be the function evaluating to 1 if w is the discrete logarithm of p_b^a w.r.t. h and 0 otherwise. Given two vectors of n commitment parameters of GGHK $\mathbf{x}_1 = ((p_0^1, p_1^1), \dots, (p_0^n, p_1^n))$ and $\mathbf{x}_2 = ((q_0^1, q_1^1), \dots, (q_0^n, q_1^n))$, and two vectors of corresponding witnesses $\mathbf{w}_p = (w_p^1, \dots, w_p^n)$ and $\mathbf{w}_q = (w_q^1, \dots, w_q^n)$, Formula 3 can be rewritten as follows:

[12]To make the formula consistent, we assign the index 1 to the first position within the vector and we say that $s_i[0] = 0$.

$$\bigvee_{i=1}^{n}\left(\left(\bigwedge_{j=0}^{i-1}\left(\left(\left(\mathcal{R}_{\mathsf{DL}}(p_0^j, w_p^j) \wedge \mathcal{R}_{\mathsf{DL}}(q_0^j, w_q^j)\right) \vee \left(\mathcal{R}_{\mathsf{DL}}(p_1^j, w_p^j) \wedge \mathcal{R}_{\mathsf{DL}}(q_1^j, w_q^j)\right)\right)\right)\right. \\ \left. \wedge \left(\mathcal{R}_{\mathsf{DL}}(p_1^i, w_p^i) \wedge \mathcal{R}_{\mathsf{DL}}(q_0^i, w_q^i)\right)\right). \quad (4)$$

Basically, for each bit of the strings s_1 and s_2 of Formula 3 such bits are equal if the corresponding parameters pairs have the same trapdoor position, meaning that either the sender knows the discrete log of both the first positions of the pairs, or that the same applies for the second position of both parameter pairs. In this case, a bit of the string s_i with $i \in \{1, 2\}$ is defined to be 1 if the corresponding parameters pair has in its first position a group element with a discrete log that is known to the sender, while it is defined to be to 0 if the corresponding parameters pair has in its second position a group element with a discrete log that is known to the sender.

On the communication efficiency of our instantiations of Π_{ord}. Our of Π_{ord} we have shown can be obtained by composing Σ-protocols of statements of varying complexity. Proving knowledge of a witness for each single basic instance over a parameter p_b^a can be done by relying on Schnorr-like Σ-protocols. We now analyze the communication complexity to prove Formula 4 by composing different Σ-protocols for its nested formulas. The AND clauses inside Formula 4 can be proven by the standard parallel repetition technique. The OR clauses, can be proven by using [GGHK21]. Π_{ord} has communication complexity $\mathcal{O}(k\lambda n)$[13]. More details are reported in Sec. 6.

Our communication-efficient (k, ℓ)-PTR. We build our (k, ℓ)-PTR by repeating the construction of [GGHK21] k times, each time using a different witness, and then we prove that the composition trees of such k instances are all different. Our new proof can be accomplished via a direct usage of Π_{ord} having as statements the k vectors of commitment parameters of length $\mathcal{O}(\log \ell)$ that constitutes the composition trees. Since such proof has a communication complexity[14] of $\mathcal{O}(k\lambda \log \ell)$), and the communication complexity of k repetitions of [GGHK21] is $\mathcal{O}(k(\lambda \log \ell + \mathsf{CC}(\Sigma_{base})))$ the combination of the two gives a (k, ℓ)-PTR from Σ-protocols, having communication complexity $\mathcal{O}(k(\lambda \log \ell + \mathsf{CC}(\Sigma_{base})) + k(\lambda \log \ell)) = \mathcal{O}(k(\lambda \log \ell + \mathsf{CC}(\Sigma_{base}))$.

Application to threshold ring signatures. A natural application of proofs over threshold relations is ring signatures. Given a set of ℓ signers, a ring signature is a digital signature that guarantees that a message was signed by a signer in the set (also called the ring) but at the same time protects the identity of the signer. There is a direct link between $(1, \ell)$-PTRs and ring signatures. For example, as pointed out in [GGHK21] one can get a ring signature whose size is logarithmic in the size of the ring by making their $(1, \ell)$-PTR non-interactive using a random oracle (see [GGHK21] Page 4 and Sec. 9.3). The ℓ public keys would be the statement of the proof, and the composed Σ-protocol would be a proof of knowledge of at least one signing key (i.e., the witness). Properly using the random oracle makes the signing process non-interactive and ties the proof to the signed message. Threshold ring signatures are a natural extension of ring signatures. In a threshold ring signature, k signers cooperate to sign a message while hiding their identity within the larger group of size ℓ. By using our construction for (k, ℓ)-PTR one can easily get a threshold ring signature.

Let the statement $x = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ be a tuple containing the signers' verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$. Consider the associated $(1, \ell)$-PTR to prove knowledge of at least one signing key sk_i related to one of the verification keys $(\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$, and the (k, ℓ)-PTR to prove knowledge of at least k signing keys related to k of the ℓ verification keys. Let Σ_{base} be the base Σ-protocol to prove the knowledge of the signing key sk_i related to the verification key vk_i. Finally, let the aggregator A be a third party who aggregates the work of all the k signers to create the final signature. A is trusted for anonymity, but not for unforgeability (roughly,

[13]Now we are explicitly including the security parameter in the analysis of the communication complexity.

[14]Note that the parameter n previously used corresponds to $\log \ell$ in this case.

A knows the identities of the signers, but should not be able to produce a threshold ring signature on its own).

Let $\mathcal{K}$ be a set of indexes indicating the k signers within the ring of size ℓ. Consider a message m to be signed which is known to all signers and to A. The description of our approach follows.

1. For all $i \in \mathcal{K}$, each signer S_i computes the first-round message a_i of Σ_{base} for statement vk_i and witness sk_i, and sends a_i to A.

2. Using the first-round messages for Σ_{base} received in the previous step, A computes the first-round messages $a_i^{1,\ell} = (\mathtt{com}, (p_0^j, p_1^j)_{j \in [\log \ell]})$[15] for each of the k underlying $(1, \ell)$-PTR of [GGHK21] (cfr., App. C) for statement x. Moreover, A computes the first-round message a_T of the proof of parameters ordering Π_{ord} with the k tuples of commitment parameters pairs $(p_0^j, p_1^j)_{j \in [\log \ell]}$ (i.e., one for each $a_i^{1,\ell}$ computed by A) as statement[16].

   A sends all the first-round messages of the underlying $(1, \ell)$-PTR, (i.e., $(a_1^{1,\ell}, \dots, a_k^{1,\ell})$), together with Π_{ord}'s first-round message a_T to each S_i.

3. S_i computes the challenge c for (k, ℓ)-PTR compiler described in Sec. 6 by hashing the statement x together with the (k, ℓ)-PTR first-round message $a = (a_1^{1,\ell}, \dots, a_k^{1,\ell}, a_T)$ and the message m to be signed (i.e., $c = \mathsf{H}(x||a||m)$). Then, S_i computes the third-round message z_i of Σ_{base} from its first-round message a_i and the challenge c, and sends z_i to A.

4. A computes each third-round message $z_i^{1,\ell}$ of the [GGHK21] compiler from the statement x, a_i, $c = \mathsf{H}(x||a||m)$, and the values z_i received from each S_i. Then, A computes the third-round message z_T of the proof of parameters ordering Π_{ord} on the same challenge c and first-round message a_T. A publishes the (k, ℓ)-threshold ring signature $\sigma = (a = (a_1^{1,\ell}, \dots, a_k^{1,\ell}, a_T),\ c = \mathsf{H}(x||a||m),\ z = (z_1^{1,\ell}, \dots, z_k^{1,\ell}, z_T))$ of a message m under the signers' verification keys $x = (\mathsf{vk}_1, \dots, \mathsf{vk}_\ell)$.

Our above construction can be seen as A applying the Fiat-Shamir transform to our (k, ℓ)-PTR by delegating the computation of the underlying protocol Σ_{base} (to prove knowledge of the signing key sk_i relative to the verification key vk_i) to each signer S_i.

Informally, A is unable to forge a signature because of (1) the special soundness of the underlying Σ_{base} (i.e., each signer S_i cannot come up with an accepting transcript (a_i, c, z_i) without knowing sk_i), (2) the special soundness of the non-interactive version of our (k, ℓ)-PTR compiler of Sec. 6 (i.e., A cannot come up with an accepting transcript without knowing at least k accepting transcripts for Σ_{base} on the challenge $c = \mathsf{H}(x||a||m)$ and statement x, where each of those transcripts is computed from different signing keys), (3) the ZK property of the underlying non-interactive version of Σ_{base};[17] (i.e., A cannot learn the signing key of any signer S_i by interacting with S_i). Anonymity, instead, is guaranteed by the zero-knowledge property of the non-interactive version of our (k, ℓ)-PTR.

Extension to threshold-of-threshold ring signatures. We can go beyond threshold ring signatures. Consider the following scenario as a possible example. An organization is made of ℓ_2 different sub-groups, each of them containing ℓ_1 signers. Each sub-group can approve the content of a message if k_1 members agree to sign the message. Then, organization-wide, this message is considered approved if at least k_2 of the ℓ_2 sub-groups signed such message. We name such scenario threshold-of-threshold ring signature.

The aggregator A, in order to compute such a threshold-of-threshold ring signature, will compute a (k_2, ℓ_2)-PTR with the approach described in the previous paragraph using a (k_1, ℓ_1)-PTR as the base Σ-protocol: A will compute the first-round messages of the (k_2, ℓ_2)-PTR starting from the first-round messages

[15]In [GGHK21] $a_i^{1,\ell}$ is obtained from recursively composing $\ell/2$ instances of a 1-out-of-2 equivocal commitment. Therefore, $a_i^{1,\ell}$ is of the form $(\mathtt{com}, (p_0^j, p_1^j)_{j \in [\log \ell]})$, where $(p_0^j, p_1^j)_{j \in [\log \ell]}$ is a $\log \ell$ size tuple of pair of commitment parameters.

[16]A will use, for each a_i containing the commitment parameters tuple $(p_0^j, p_1^j)_{j \in [\log \ell]}$, the corresponding witness $W = (w_j)_{j \in [\log \ell]}$ to generate the first-round message a_T of Π_{ord}.

[17]Note that when a Σ-protocol with HVZK is compiled into a non-interactive argument system using the Fiat-Shamir transform it becomes ZK in the random oracle model.

of the k_2 underlying instances of the (k_1, ℓ_1)-PTR used as a base Σ-protocol. A, instead of directly communicating with the signers, will now talk to aggregators $A_1, \dots, A_{\ell_2}$, where each A_i, $i \in [\ell_2]$ is associated to a sub-group of signers $S^i_1, \dots, S^i_{\ell_1}$. At least k_2 of those A_i for $i \in \{1, \dots, \ell_2\}$ will, in turn, execute the base Σ-protocol for a (k_1, ℓ_1)-PTR by interacting with its own subset of signers $S^i_1, \dots, S^i_{k_1}$ using the technique described in the previous paragraph, with the difference that the challenge c now depends on the statement used by A for the (k_2, ℓ_2)-PTR. The very same technique can be easily extended to an arbitrary number of levels.

Considering two levels of aggregators, the size of the threshold-of-threshold ring signature computed by A is $\mathcal{O}(k_2(\lambda \log \ell_2 + \mathsf{CC}(\Sigma_{k_1,\ell_1}))) = \mathcal{O}(k_2(\lambda \log \ell_2 + k_1(\lambda \log \ell_1 + \mathsf{CC}(\Sigma_{\mathsf{base}}))))$, where $\mathsf{CC}(\Sigma_{k_1,\ell_1})$ is the communication complexity of the underlying (k_1, ℓ_1)-PTR computed by each A_i and $\mathsf{CC}(\Sigma_{\mathsf{base}})$ is the communication complexity of the base Σ-protocol used by each of the k_1 signers interacting with each A_i.

2 More on Related Work vs Our Results

Here, we give a more detailed discussion of how our results compare to the related work.

Σ-protocols. The breakthrough work by Cramer et al. [CDS94] devise a composition technique to obtain (k, ℓ)-PTR from Σ-protocols. This protocol obtains a communication complexity which is linear in ℓ, it achieves special soundness in the plain model. Inspired by the design principles of [CDS94], a rather recent line of works proposes new composition techniques to reduce the communication complexity or to obtain enhanced security guarantees.

Groth and Kohlweiss [GK15] devise a technique that achieves communication complexity which is logarithmic in ℓ specifically for the discrete-log relation and only for the value $k = 1$. Soundness holds computationally. The security is proven in the SRSmodel.

Recently, Attema and Cramer formulated compressed Σ-protocol theory [AC20] by appropriately recasting Bulletproof's [BCC$^+$16, BBB$^+$18] compression mechanism. Attema, Cramer and Fehr [ACF21] exploit compressed Σ-protocol theory to construct (k, ℓ)-PTR which has logarithmic communication complexity both in k and ℓ. Unfortunately, the composition of [ACF21] only applies to the discrete logarithm relation (or variations of it). Another drawback is that the resulting protocol has a logarithmic number of rounds. Soundness holds computationally. Security is proven in the shared random string model.

Ciampi et al. [CPS$^+$16] devise a (k, ℓ)-PTR by composing Σ-protocols with the aim of supporting delayed instances specification. They follow the blueprint of [CDS94] but they deploy a different technique to allow the prover to cheat on inactive clauses. This technique relies on the notion of k-out-of-ℓ equivocal commitment, namely a commitment to a vector of ℓ elements, k of which are binding and $\ell - k$ can be equivocated.

More recently, Goel et al. [GGHK21] crucially exploit the ideas presented in [CPS$^+$16] but with a different goal, that is reducing communication complexity. The new idea is recycling the same third-round message across all the instances. By applying their strategy recursively on 1-out-of-2 relations, [GGHK21] achieves logarithmic communication complexity when $k = 1$. The protocol enjoys special soundness which holds computationally. Their results are proven secure in the SRS model. Their composition is generic and applies to a broad class of Σ-protocols. Goel et al. also propose a technique to generalize their result to arbitrary k, however their approach falls back to a communication complexity that is linear both in k and in ℓ.

Our improvements. We improve the state of the art in the following dimensions:

- By taking advantage of approach used in [GGHK21], our technique applies to a large family of base protocols and preserves the honest-verifier zero-knowledge flavour of the base protocols.
- Our technique works for arbitrary value of k providing a communication complexity that is linear in k and logarithmic in ℓ. This strictly improves on the communication complexity of [GGHK21]. [ACF21] obtains a communication complexity that is logarithmic both in k and ℓ, but it only applies to the discrete logarithm relation (or variations of it).

Other approaches. Heath and Kolesnikov [HK20] extended the work of Jawureck et al. [JKO13] greatly optimizing the communication complexity of zero-knowledge proofs based on garbled circuits in the case of conditional branching. This approach leads to a communication complexity that is proportional to the longest branch. More recently, [HKP21] extended this result to (k, ℓ)-PTRs retaining the same communication advantage while optimizing computation efficiency. Mac'n'Cheese by Baum et al. [BMRS21] is an interactive commit-and-prove zero-knowledge proof system for binary and arithmetic circuits. As commitments it uses information-theoretic MACs based on vector oblivious linear evaluation (VOLE). For k-out-of-ℓ statements, the communication complexity is proportional to k times the longest circuit plus an additive term which is logarithmic in ℓ. Note that Mac'n'Cheese is inherently private coin since the soundness relies on the verifier keeping the MAC key secret. Therefore, it is not immediately clear whether it can be modified to support public verifiability. Instead all approaches producing Σ-protocols (like ours) can be turned into non-interactive publicly verifiable zero-knowledge proofs in the RO model applying the Fiat-Shamir transform.

Finally, one might leverage succinct proof techniques such as STARKs or SNARKs [CHM+20, CFQ19, GWC19, CFF+21, MBKM19, Set20, BBHR19] to get a communication-efficient (k, ℓ)-PTR. While succinctness could be heavily optimized achieving constant proof sizes, removing the dependency on k and ℓ. On the other hand, these techniques have several drawbacks such as a huge workload for provers and the use of strong assumptions and/or problematic trusted setups.

Although the approaches above apply to NP-complete languages, we note that they are not so obviously efficiently generalizable to arbitrary languages. Instead, our approach is more beneficial to protocol designers. Indeed, if there is a Σ-protocol for the base relation, a protocol designer can use our solution directly for the (k, ℓ) case without the need to run a possibly expensive NP-reduction.

Threshold ring signature. Several works construct threshold ring signature from a variety of assumptions. Our threshold ring signature achieves shorter signature tags than several known threshold ring signatures schemes [BSS02, YLA+13, OTYO18], other schemes have signature size which is linear in ℓ and thus are also outperformed when $k << \ell$ [ZZY+17, CHGa19, HS20].

In [ACR21], Attema et al. exploit the (k, ℓ)-PTR of [ACF21] to build a threshold ring signature. [ACR21] combines the (k, ℓ)-PTR of [ACF21] with BLS signatures [BLS01] to obtain a threshold ring signature whose size is logarithmic in ℓ and independent of k. Their scheme requires a shared random string as setup and pairing groups. A recent work by Munch-Hansen et al. [MOY21] achieves a signature size which is linear in k and does not depend on ℓ, it also achieves stronger anonymity guarantees than traditional definitions (i.e., signers' anonymity is preserved even within the smaller coalition of the k signers) but it requires a trusted setup. Another work by Haque et al. [HKSS20] does not require any trusted setup and achieves a signature size which is linear in k and does not depend on ℓ. However, to avoid forgeries and anonymity attacks, the keys need to be honestly generated by a trusted party different than the signers. It is interesting to note that our solution, while being very simple and straightforward, leads to a threshold ring signature scheme which has a good trade-off between size, assumptions and features:

- The signature size $\mathcal{O}(k \log \ell)$ outperforms several previous works.
- When comparing our construction to other works achieving more compact signature tags [HKSS20, MOY21, ACR21], our construction presents other advantages such as:
  (i) It can be instantiated from several Σ-protocols and commitment schemes, giving more flexibility in terms of used assumptions.
  (ii) It does not require pairing groups.
  (iii) It supports adversarially generated public keys.

The above advantages can be crucial in some contexts. For instance when considering Bitcoin, the requirement of a common hash function is for free and the random oracle model is well accepted. Moreover, current signatures schemes supported in Bitcoin do not use pairings. In some sense an update of Bitcoin that would allow to write scripts able to verify our threshold ring signatures would be much milder and less traumatic (and thus easier to be accepted by the community) than what is required by other threshold ring signature schemes.

3 Preliminaries

3.1 Notations

We use $\mathbb{N}$ to denote the set of all natural numbers and we let PPT stand for probabilistic polynomial time. For a probabilistic algorithm A, $A(x)$ denotes the probability distribution of the output of A when run with x as input. We use $A(x;r)$ instead to denote the output of A when run on input x and coin tosses r. We denote with $\lambda \in \mathbb{N}$ the security parameter and with $\texttt{poly}(\cdot)$ a positive polynomial $\texttt{poly}$. Every algorithm take in input the security parameter 1^λ in unary. When an algorithm takes more than one input, 1^λ is omitted. We say that a function $\nu : \mathbb{N} \to \mathbb{R}$ is negligible if every positive polynomial $\texttt{poly}(\cdot)$ and all sufficiently large λ it holds that $\nu(\lambda) \leq \frac{1}{\texttt{poly}(\lambda)}$.

A *polynomial-time* relation $\mathcal{R}$ is a relation for which membership of (x, w) to $\mathcal{R}$ can be decided in time polynomial in $|x|$. If $(x, w) \in \mathcal{R}$ then we say that w is a *witness* for the *instance* x. A polynomial-time relation $\mathcal{R}$ is naturally associated with the NP language $\mathcal{L}_\mathcal{R}$ defined as $\mathcal{L}_\mathcal{R} = \{x | \exists w : (x, w) \in \mathcal{R}\}$. Similarly, an NP language is naturally associated with a polynomial-time relation.

In this paper we consider the notion of *threshold relation.* Let $\mathcal{R}_1, \dots, \mathcal{R}_n$ be polynomial-time relations. A threshold relation with threshold k is a polynomial-time relation as well, with the following form

$$\begin{aligned}\mathcal{R}_{\mathsf{k},\ell} =& \{(x_1, \dots, x_n), (w_1, d_1) \dots, (w_k, d_k)) : \\ & 1 \leq d_1 < \cdots < d_k \leq n \wedge (x_{d_i}, w_i) \in \mathcal{R}_i, \text{ for } i = 1, \dots, k\}.\end{aligned}$$

Roughly, a threshold relation with threshold k contains a set of n NP statements $x_1, \dots, x_n$, and a set of witnesses $w_1, \dots, w_k$ with $k \leq n$, in which each w_i represents a valid witness for one and only one statement $x_{d_i} \in \mathcal{L}_{\mathcal{R}_{d_i}}$ with $d_i \in \{1, \dots, n\}$.

A distribution ensemble $\{X(a)\}_{a \in \{0,1\}^*}$ is an infinite sequence of probability distributions, where a distribution $X(a)$ is associated with each value of a. We say that two distribution ensembles $\{X(n)\}_{n\in\mathbb{N}}$ and $\{Y(n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for every PPT distinguisher $\mathcal{D}$, there exists a negligible function ν such that for all $n \in \mathbb{N}$,

$$\text{Prob}\,[\,\mathcal{D}(X(n), n) = 1\,] - \text{Prob}\,[\,\mathcal{D}(Y(n), n) = 1\,] \leq \nu(n).$$

We say that $\{X(n)\}_{n\in\mathbb{N}}$ and $\{Y(n)\}_{n\in\mathbb{N}}$ are statistically indistinguishable if the above holds for all $\mathcal{D}$. We can also use $\approx$ to indicate that two distributions are identically distributed.

We denote by $[n]$ for an $n \in \mathbb{N}$ the set of numbers $\{1, \dots, n\}$. We refer to vectors as $\mathbf{v}$ and to indicate the i-th position of $\mathbf{v}$, we write $\mathbf{v}[i]$, we do the same for binary strings. When using $\leftarrow$ we mean that the variable on the left side is assigned with the output value of the algorithm on the right side. With $\leftarrow\!\!\$$, we indicate that the variable on the left side is assigned a values sampled randomly according to the distribution on the right side.

3.2 Σ-Protocols

We consider a *3-round* public-coin protocol Π for an NP language $\mathcal{L}$ with a poly-time relation $\mathcal{R}_\mathcal{L}$. $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ is run by a prover running auxiliary algorithms $\mathsf{P}_0, \mathsf{P}_1$ and a verifier running an auxiliary algorithm V. The prover and the verifier receive common input x and the security parameter 1^λ in unary. The prover receives as an additional private input a witness w for x. Prover and verifier use the auxiliary algorithms $\mathsf{P}_0, \mathsf{P}_1, \mathsf{V}$ in the following way:

1. The prover runs P_0 on common input x, private input w, randomness R, and outputs a message a. The prover sends a to the verifier;

2. The verifier samples a random challenge $c \leftarrow_{\$} \{0,1\}^\lambda$ and sends c to the prover;
3. The prover runs P_1 on common input x, private input w, first-round message a, randomness R, and challenge c, and outputs the third-round message z, which is then sent to the verifier;
4. The verifier outputs 1 if $\mathsf{V}(x,a,c,z) = 1$, and rejects otherwise.

The transcript (a,c,z) for the protocol $\Pi = (\mathsf{P}_0,\mathsf{P}_1,\mathsf{V})$, and common statement x is called *accepting* if $\mathsf{V}(x,a,c,z) = 1$.

Definition 1. *A 3-round public-coin protocol* $\Pi = (\mathsf{P}_0,\mathsf{P}_1,\mathsf{V})$*, is a* Σ*-protocol for an NP language* $\mathcal{L}$ *with a poly-time relation* $\mathcal{R}_\mathcal{L}$ *iff the following properties are satisfied*

Completeness: *For all* $x \in \mathcal{L}$ *and* w *such that* $(x,w) \in \mathcal{R}_\mathcal{L}$ *it holds that:*

$$\text{Prob}\left[\mathsf{V}(x,a,c,z) = 1 \;\middle|\; \begin{array}{c} R \leftarrow_{\$} \{0,1\}^\lambda; c \leftarrow_{\$} \{0,1\}^\lambda; \\ a \leftarrow \mathsf{P}_0(x,w;R); \\ z \leftarrow \mathsf{P}_1(x,w,a,c;R) \end{array} \right] = 1.$$

Special Soundness: $\exists$ *PPT* $\mathsf{Extract}$*, such that on input* x *and two accepting transcripts* (a,c_0,z_0) *and* (a,c_1,z_1) *for* x*, where* $c_0 \neq c_1$*, it holds that*

$$\text{Prob}\left[\, (x,w) \in \mathcal{R}_\mathcal{L} | w \leftarrow \mathsf{Extract}(x,a,c_0,c_1,z_0,z_1) \,\right] = 1.$$

Special Honest-Verifier Zero-Knowledge (SHVZK): *There exists a PPT simulator* $\mathcal{S}$ *that, on input an instance* $x \in \mathcal{L}$ *and challenge* c*, outputs* (a,z) *such that* (a,c,z) *is an accepting transcript for* x*. Moreover, the distribution of the output of* $\mathcal{S}$ *on input* (x,c) *is computationally/statistically/perfectly indistinguishable from the distribution obtained when the verifier sends c as challenge and the prover runs on common input* x *and any private input* w *such that* $(x,w) \in \mathcal{R}_\mathcal{L}$*.*

We also define the weaker notion of computational special soundness. Indeed, in [GGHK21], the first message of the compiled Σ-protocol exploits the compression power of a collision-resistant hash function, thus the overall composition cannot achieve special soundness when instantiated with GGHK.

Definition 2. *A 3-round public-coin protocol* $\Pi = (\mathsf{P}_0,\mathsf{P}_1,\mathsf{V})$ *is a computational* Σ*-protocol for an NP language* $\mathcal{L}$ *with a poly-time relation* $\mathcal{R}_\mathcal{L}$ *if and only if it is complete, (computational/statistical/perfect) special honest-verifier zero knowledge, and computational special sound. Computational special soundness is specified below.*

Computational Special Soundness: $\exists$ *PPT* $\mathsf{Extract}$ *s.t.* $\forall$ *PPT* P^* $\exists$ *a negligible function* $\nu(\cdot)$ *such that* $\forall x \in \mathcal{L}$ *it holds that*

$$\text{Prob}\left[\, \mathsf{ExpExt}_{\mathsf{P}^*,\mathsf{Extract}}(x) = 1 \,\right] \leq \nu(|x|).$$

$\mathsf{ExpExt}_{\mathsf{P}^*,\mathsf{Extract}}(x)$

```
1. $(a,c_0,c_1,z_0,z_1) \leftarrow \mathsf{P}^*(x)$.
2. If $c_0 \neq c_1$, or $\mathsf{V}(x,a,c_0,z_0) = 0$, or $\mathsf{V}(x,a,c_1,z_1) = 0$ return 0.
3. $w \leftarrow \mathsf{Extract}(x,a,c_0,c_1,z_0,z_1)$.
4. Return 1 if $(x,w) \notin \mathcal{R}_\mathcal{L}$. Otherwise, return 0.
```

From now on, we will refer both to Σ-protocols and computational Σ-protocols simply as Σ-protocols. We will instead clearly state the considered flavour whenever it is useful for the discussion.

3.3 Stackable Σ-protocols

Similarly to [GGHK21], we define stackable Σ-protocols below.

Definition 3 (Computational/Statistical EHVZK)**.** *Let $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$, be a Σ-protocol for an NP language $\mathcal{L}$. Σ is extended honest-verifier zero knowledge (EHVZK) if there exists a PPT algorithm $\mathcal{S}^{\mathsf{EHVZK}}$ such that for all PPT/unbounded $\mathcal{D}$, and $c \in \{0,1\}^\lambda$, there exists an efficiently samplable distribution $D^{(z)}_{x,c}$ and a negligible function $\nu(\cdot)$ such that for all $x \in \mathcal{L}$*

$$\left|\text{Prob}\left[\ \mathsf{ExpEHVZK}_{(\mathsf{P}_0,\mathsf{P}_1),\mathcal{D}}(c) = 1\ \right] - \text{Prob}\left[\ \mathsf{ExpEHVZK}_{\mathcal{S}^{\mathsf{EHVZK}},\mathcal{D}}(c) = 1\ \right]\right| \leq \nu(|x|).$$

We say instead that a Σ-protocol is perfect EHVZK, if

$$\text{Prob}\left[\ \mathsf{ExpEHVZK}_{(\mathsf{P}_0,\mathsf{P}_1),\mathcal{D}}(c) = 1\ \right] - \text{Prob}\left[\ \mathsf{ExpEHVZK}_{\mathcal{S}^{\mathsf{EHVZK}},\mathcal{D}}(c) = 1\ \right] = 0,$$

and there is no constraint on the running time of $\mathcal{D}$. The experiment $\mathsf{ExpEHVZK}$ for EHVZK follows.

$\mathsf{ExpEHVZK}_{\mathsf{P}',\mathcal{D}}(c)$

1. $(x, w) \leftarrow \mathcal{D}(c)$.
2. If $(x, w) \notin \mathcal{R}_\mathcal{L}$, return 0.
3. If $\mathsf{P}' = \mathcal{S}^{\mathsf{EHVZK}}$, sample $z \leftarrow\$ D^{(z)}_{x,c}$ and compute $a \leftarrow \mathcal{S}^{\mathsf{EHVZK}}(x, c, z)$.
4. Otherwise, sample $R \leftarrow\$ \{0,1\}^\lambda$, compute $a \leftarrow \mathsf{P}_0(x, w; R)$ and $z \leftarrow \mathsf{P}_1(x, w, a, c; R)$.
5. Return $\mathcal{D}(x, w, a, c, z)$.

Definition 4 (Σ-protocol with recyclable third messages)**.** *Let $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ be a Σ-protocol for an NP language $\mathcal{L}$, Σ has recyclable third messages if for each $c \in \{0,1\}^\lambda$, there exists an efficiently samplable distribution $D^{(z)}_c$, such that for all $(x, w) \in \mathcal{R}_\mathcal{L}$, it holds that $D^{(z)}_c \approx \{z | R \leftarrow\$ \{0,1\}^\lambda; a \leftarrow \mathsf{P}_0(x, w; R); z \leftarrow \mathsf{P}_1(x, w, c; R)\}$.*

Definition 5 (Stackable Σ-protocol)**.** *We say that a Σ-protocol $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$, is stackable, if it is a EHVZK Σ-protocol and has recyclable third messages.*

4 1-out-of-2 Equivocal Commitment Schemes

We now define the notion of 1-out-of-2 equivocal commitment scheme. The sender commits to a pair of messages (m_0, m_1) with $m_0, m_1 \in \{0,1\}^\lambda$, where $\lambda \in \mathbb{N}$ is the security parameter. A 1-out-of-2 equivocal commitment scheme $CS = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{BindCom}, \mathsf{EquivCom}, \mathsf{Equiv}, \mathcal{R}_T)$ consists of five PPT algorithms and a polynomial-time relation $\mathcal{R}_T$. The algorithm Setup generates a common reference string pp. We denote by $\mathcal{Y}^{\mathsf{pp}}$ the space of well-formed commitment parameters w.r.t. pp and require that membership in $\mathcal{Y}^{\mathsf{pp}}$ can be checked efficiently. The above algorithms work as follows:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda; r)$: on input the security parameter, and randomness r, generates public parameters pp.
- $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta; r)$: on input public parameters pp, binding position $\beta \in \{0,1\}$, and randomness r, returns the commitment parameters $(p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}$ and the trapdoor td for parameter $p_{1-\beta}$ such that $(p_{1-\beta}, \mathsf{td})$ belongs to $\mathcal{R}_T$[18].

[18]The statement for $\mathcal{R}_T$ may also depend from pp. We will omit this dependence to simplify the notation.

- $\mathtt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0, m_1; r)$: on input public parameters pp, commitment parameters p_0, p_1, messages m_0, m_1, and randomness r outputs a commitment $\mathtt{com}$.
- $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m, p_0, p_1, \mathsf{td}; r)$: on input public parameters pp, binding position β, message of the binding position m, commitment parameters p_0, p_1, trapdoor td, and randomness r returns a commitment $\mathtt{com}$ and some auxiliary information aux.
- $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$: on input public parameters pp, binding position β, messages m_0, m_1, commitment parameters p_0, p_1, trapdoor td, and auxiliary information aux, deterministically returns an equivocation randomness r.

In the following, we assume that pp was already generated by a trusted third party using the algorithm Setup. Furthermore, we will omit the randomness from the algorithms, except when it is relevant. A sender and a receiver interact using the commitment scheme as follows.

Commit Phase: The sender, on private input m and binding position β, computes $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$, $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m, p_0, p_1, \mathsf{td})$. The sender sends $(\mathtt{com}, p_0, p_1)$ to the receiver.

Reveal Phase: The sender, on input m^*, computes $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$ where $m_\beta = m$ and $m_{1-\beta} = m^*$, and sends (r, m_0, m_1) to the receiver. The receiver computes $\mathtt{com}' \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0, m_1; r)$ and accepts if $\mathtt{com}' = \mathtt{com}$ and $(p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}$, rejects otherwise.

Definition 6 (1-out-of-2 Equivocal Commitment Scheme)**.** *The protocol CS described above is a 1-out-of-2 equivocal commitment scheme if and only if the following properties are satisfied:*

Partial Equivocation: *For all $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\beta \in \{0,1\}$, $(p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}$, $(m_0, m_1) \in \{0,1\}^{2\lambda}$, for all td such that $(p_{1-\beta}, \mathsf{td}) \in \mathcal{R}_T$ the following holds:*

$$\mathrm{Prob}\left[\begin{array}{c} \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, \\ m_0, m_1; r) = \mathtt{com} \end{array} \middle| \begin{array}{c} (\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m_\beta, p_0, p_1, \mathsf{td}); \\ r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux}). \end{array}\right] = 1.$$

Computational Fixed Equivocation: *Given the experiment $\mathsf{ExpFixEquiv}$ below, for every non-uniform PPT receiver $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for any $\lambda \in \mathbb{N}$:* $\mathrm{Prob}[\,\mathsf{ExpFixEquiv}_{\mathcal{A}}(\lambda) = 1\,] \leq \nu(\lambda)$.

$\mathsf{ExpFixEquiv}_{\mathcal{A}}(\lambda)$

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(p_0, p_1, r^1, r^2, r^3, r^4, m_0^1, m_0^2, m_1^1, m_1^2, m_0^3, m_0^4, m_1^3, m_1^4) \leftarrow \mathcal{A}(\mathsf{pp})$.
3. *Return 1 if* $\exists \beta \in \{0,1\}$ *such that*

$$\begin{aligned} &(\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^1, m_1^1; r^1) = \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^2, m_1^2; r^2) \wedge \\ &\big(\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^3, m_1^3; r^3) = \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0^4, m_1^4; r^4)\big) \wedge \\ &(m_{1-\beta}^1 \neq m_{1-\beta}^2) \wedge (m_\beta^3 \neq m_\beta^4) \wedge ((p_0, p_1) \in \mathcal{Y}^{\mathsf{pp}}). \end{aligned}$$

Return 0 otherwise.

Moreover, the protocol achieves perfect fixed equivocation if for any unbounded $\mathcal{A}$ it holds that $\mathrm{Prob}[\,\mathsf{ExpFixEquiv}_{\mathcal{A}}(\lambda) = 1\,] \leq \nu(\lambda) = 0$.

Computational Position Hiding: *Given the experiment* ExpHid *below, for every non-uniform PPT* $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$*:* $\text{Prob}[\,\mathsf{ExpHid}_{\mathcal{A}}(\lambda) = 1\,] \leq \frac{1}{2} + \nu(\lambda)$.

```
                    $\mathsf{ExpHid}_{\mathcal{A}}(\lambda)$

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. Sample $\beta \leftarrow_{\$} \{0,1\}$ and compute $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$.
3. $\beta' \leftarrow \mathcal{A}(\mathsf{pp}, p_0, p_1)$.
4. Return 1 if $\beta' = \beta$ and 0 otherwise.
```

Moreover, if $\mathcal{A}$ *is unbounded and* $\nu(\lambda) = 0$ *we say that the scheme is perfect position hiding.*

Computational Trapdoorness: *Given the experiment* $\mathsf{ExpTrap}$*, for every non-uniform PPT receiver* $\mathcal{A}$*, there exists a negligible function* $\nu(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$: $\text{Prob}[\,\mathsf{ExpTrap}_{\mathcal{A}}(\rho, \lambda) = 1\,] \leq \frac{1}{2} + \nu(\lambda)$.

```
                    $\mathsf{ExpTrap}_{\mathcal{A}}(\lambda)$

1. $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(m_0, m_1, p_0, p_1, \mathsf{td}, \beta) \leftarrow \mathcal{A}(\mathsf{pp})$.
3. If $(p_0, p_1) \notin \mathcal{Y}^{\mathsf{pp}}$ or $(p_{1-\beta}, \mathsf{td}) \notin \mathcal{R}_T$ abort the experiment.
4. Sample $b \leftarrow_{\$} \{0,1\}$. If $b = 0$, set $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, m_\beta, p_0, p_1, \mathsf{td})$ and set
   $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$. If $b = 1$, sample $r \leftarrow_{\$} \mathtt{D}$ and set $\mathtt{com} \leftarrow$
   $\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m_0, m_1; r)$.
5. $b' \leftarrow \mathcal{A}(\mathsf{pp}, m_0, m_1, p_0, p_1, \mathsf{td}, \beta, \mathtt{com}, r)$.
6. Return 1 if $b = b'$, return 0 otherwise.
```

Moreover, if $\mathcal{A}$ *is unbounded and* $\nu(\lambda) = 0$ *we say that the protocol achieves perfect trapdoorness.*

In App. B we show the instantiation of the commitment scheme GGHK [GGHK21] according to the above syntax. Our definition slightly differs from the one formulated in [GGHK21]. In particular, our fixed equivocation property implies the partial binding of [GGHK21]. We need this slightly stronger property to prove the soundness of our (k, ℓ)-PTR. We point out that natural instantiations such as GGHK also enjoy the fixed equivocation property. The remaining properties are just a restatement of the minimal requirements for a 1-out-of-2 commitment scheme in [GGHK21]. See App. C for more details.

5 The $(1, \ell)$-Proof over Threshold Relations of [GGHK21]

In [GGHK21], the authors directly present the $(1, \ell)$-PTR system by leveraging a 1-out-of-ℓ equivocal commitment. Then they argue on how the complexity of the protocol can be reduced recursively composing $(1, 2)$-PTRs obtained with their compiler. Therefore, for the sake of simplicity, we directly focus on the $(1, 2)$-PTR.

To describe the $(1, 2)$-PTR system we use the 1-out-of-2 equivocal commitment scheme as defined in Sec. 4.

Let $\Pi' = (\mathsf{P}'_0, \mathsf{P}'_1, \mathsf{V}')$ be a stackable Σ-protocol for language $\mathcal{L}$ and let $\mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}$ be the simulator for Π'. Let $x = (x_0, x_1)$ be the public input. Let $w = (w_\alpha, \alpha)$, with $\alpha \in \{0, 1\}$, so that $(x_\alpha, w_\alpha) \in \mathcal{R}_{\mathcal{L}}$ be the prover's private input. [GGHK21] proposes a compiler that generates a stackable Σ-protocol $\Pi = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$, starting from Π' for proving the relation $\mathcal{R} = \{(x = (x_0, x_1), w_\alpha) | (x_0, w_\alpha) \in \mathcal{R}_\mathcal{L} \ \vee (x_1, w_\alpha) \in \mathcal{R}_\mathcal{L}\}$. We report the compiler in Fig. 2. In the compiled protocol Π, we set the sender S of the 1-out-of-2 commitment scheme to be the prover $(\mathsf{P}_0, \mathsf{P}_1)$, while the verifier V is the receiver R.

Given the compiler in Fig. 2 it is straightforward to extend this protocol to a $(1, \ell)$-PTR using recursion. Since in [GGHK21] the proof of computational special soundness and EHVZK is done w.r.t. a non-interactive commitment scheme, we report in App. C a sketch of the proof using an interactive commitment scheme.

The $(1, \ell)$-PTR of [GGHK21] is a stackable Σ-protocol with computational special soundness. If the employed 1-out-of-2 equivocal commitment scheme satisfies perfect trapdorness and perfect position hiding, then the EHVZK flavour (i.e., perfect, statistical or computational) of the base Σ-protocol is preserved.

6 Our Compact (k, ℓ)-PTR from Stackable Σ-Protocols

6.1 Proof of Parameters Ordering for DL Parameters

We now introduce Π_{ord} that can be used to prove that there is a strict total ordering among vectors of 1-out-of-2 equivocal commitment parameters of GGHK. Given k vectors $\mathbf{p}_1, \dots, \mathbf{p}_k$ of n pairs of parameters of a 1-out-of-2 equivocal commitment and k vectors $\mathbf{w}_1, \dots, \mathbf{w}_k$ of witnesses, the relation proved by Π_{ord}, can be defined starting from Formula 3. The first step is to rewrite Formula 3, obtaining a formula $\mathcal{R}_{ord'}$ for $\Pi_{ord'}$ as follows. Given two vectors of parameters $\mathbf{p} = ((p_0^1, p_1^1), \dots, (p_0^n, p_1^n))$ and $\mathbf{q} = ((q_0^1, q_1^1), \dots, (q_0^n, q_1^n))$, two vectors of parameters trapdoors $\mathbf{w}_p = (w_p^1, \dots, w_p^n)$ and $\mathbf{w}_q = (w_q^1, \dots, w_q^n)$, $\mathcal{R}_{ord'}((\mathbf{p}, \mathbf{q}), (\mathbf{w}_p, \mathbf{w}_q))$ is[19]

$$\bigvee_{i=1}^{n} \left(\left(\bigwedge_{j=0}^{i-1} (((\mathcal{R}_{\mathsf{DL}}(p_0^j, w_p^j) \wedge \mathcal{R}_{\mathsf{DL}}(q_0^j, w_q^j)) \vee (\mathcal{R}_{\mathsf{DL}}(p_1^j, w_p^j) \wedge \mathcal{R}_{\mathsf{DL}}(q_1^j, w_q^j))) \right) \wedge (\mathcal{R}_{\mathsf{DL}}(p_1^i, w_p^i) \wedge \mathcal{R}_{\mathsf{DL}}(q_0^i, w_q^i)) \right). \tag{5}$$

We define the relation for Π_{ord} as follows:

$$\mathcal{R}_{ord}((\mathbf{p}_1, \dots, \mathbf{p}_k), (\mathbf{w}_1, \dots, \mathbf{w}_k)) = \bigwedge_{i=1}^{k-1} \mathcal{R}_{ord'}((\mathbf{p}_i, \mathbf{p}_{i+1}), (\mathbf{w}_i, \mathbf{w}_{i+1})). \tag{6}$$

We point out that the first part of formula $\mathcal{R}_{ord}$ can be expressed as a composition of subformulas. In turn, each of those subformulas can be described in the same way. We describe the subformulas $\mathcal{R}_i(x, w)$ as follows:

$$\begin{aligned}
\mathcal{R}_{ord'}((\mathbf{p}, \mathbf{q}), (\mathbf{w}_p, \mathbf{w}_q)) &= \bigvee_{i=1}^{n} \Big(R_1^i((\mathbf{p}, \mathbf{q}), (\mathbf{w}_p, \mathbf{w}_q)) \wedge \mathcal{R}_3((p_1^i, q_0^i), (w_i^p, w_i^q)) \Big). \\
\mathcal{R}_1^i((\mathbf{p}, \mathbf{q}), (\mathbf{w}_p, \mathbf{w}_q)) &= \bigwedge_{j=0}^{i-1} \Big(\mathcal{R}_2((p_0^j, p_1^j, q_0^j, q_1^j), (w_p^j, w_q^j)) \wedge \mathcal{R}_3((p_0^i, q_1^i), (w_p^i, w_q^i)) \Big). \\
\mathcal{R}_2((p_0, p_1, q_0, q_1), (w_p, w_q)) &= \Big(\mathcal{R}_3((p_0, q_0), (w_p, w_q)) \vee \mathcal{R}_3((p_1, q_1), (w_p, w_q)) \Big). \\
\mathcal{R}_3((p, q), (w_p, w_q)) &= \Big(\mathcal{R}_{\mathsf{DL}}(p, w_p) \wedge \mathcal{R}_{\mathsf{DL}}(q, w_q) \Big).
\end{aligned}$$

We point out that that all the relations described above can be realized with a stackable Σ-protocol. In particular, we can prove $\mathcal{R}_3$ via the AND composition of two Σ-protocols for $\mathcal{R}_{\mathsf{DL}}$ (i.e., with parallel

[19]In the following formula the AND ranging from $j = 0$ to $j = i - 1$ is evaluated as true for $j = 0$. Indeed, according to the notation used in this paper, there are no parameters pair in the position 0 of the vector.

Let $\Pi' = (\mathsf{P}'_0, \mathsf{P}'_1, \mathsf{V}')$ be a stackable Σ-protocol for relation $\mathcal{R}_\mathcal{L}$ and CS be a 1-out-of-2 equivocal commitment scheme, the following compiler produces a stackable Σ-protocol $\Pi^{1,2} = (\mathsf{P}_0^{1,2}, \mathsf{P}_1^{1,2}, \mathsf{V}^{1,2})$ for relation $\mathcal{R}_{OR} = \{((x_0, x_1), w) | (x_0, w) \in \mathcal{R}_\mathcal{L} \vee (x_1, w) \in \mathcal{R}_\mathcal{L}\}$. Let pp be public parameters generated by a trusted party running Setup.

First round: The prover, on input $(x, (w, \beta), \mathtt{rand}_\mathsf{P})$, runs $\mathsf{P}_0^{1,2}(x, w, \beta; \mathtt{rand}_\mathsf{P})$ as follows:

- Parse $\mathtt{rand}_\mathsf{P} = (\mathtt{rand}^\beta || \mathtt{rand}_0 || \mathtt{rand}_1)$;
- Set $v_\beta \leftarrow \mathsf{P}'_0(x_\beta, w_\beta; \mathtt{rand}^\beta)$;
- Run $\mathsf{Gen}(\mathsf{pp}, \beta; \mathtt{rand}_0)$ to obtain (p_0, p_1, td);
- Run $\mathsf{EquivCom}(\mathsf{pp}, \beta, v_\beta, p_0, p_1, \mathsf{td}; \mathtt{rand}_1)$ to obtain $(\mathtt{com}, \mathsf{aux})$;
- Output (σ_1, td) where $\sigma_1 = (\mathtt{com}, p_0, p_1)$.

Finally, the prover sends $a = \sigma_1$ to the verifier.

Second round: The verifier samples a challenge $c \leftarrow_\$ \{0,1\}^\lambda$ and sends c to the prover.

Third round: The prover runs $\mathsf{P}_1^{1,2}(x, (w, \beta), a, v_\beta, c; \mathtt{rand}_\mathsf{P})$ as follows:

- Parse $\mathtt{rand}_\mathsf{P} = (\mathtt{rand}^\beta || \mathtt{rand}_0 || \mathtt{rand}_1)$ and $a = (\mathtt{com}, p_0, p_1)$;
- Compute $z^* \leftarrow \mathsf{P}'_1(x_\beta, w_\beta, v_\beta, c; \mathtt{rand}^\beta)$;
- Set $v^*_{1-\beta} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_{1-\beta}, c, z^*)$;
- Set $v^*_\beta = v_\beta$;
- Run $\mathsf{Equiv}(\mathsf{pp}, \beta, v^*_0, v^*_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$ (where aux is computed with $\mathtt{rand}_1$) and receive r;
- Output $z = (z^*, r, p_0, p_1)$.

Finally, the prover sends z to $\mathsf{V}^{1,2}$.

Verification: $\mathsf{V}^{1,2}$, in input (x, a, c, z), does as follows:

- Parse $a = (\mathtt{com}, p_0, p_1)$ and $z = (z^*, r, p'_0, p'_1)$;
- If $(p_0, p_1) \neq (p'_0, p'_1)$ return 0, otherwise continue to next step;
- If $(p_0, p_1) \notin \mathcal{Y}^{\mathsf{pp}}$ return 0, otherwise continue to next step;
- Set $v_i \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_i, c, z^*)$, for $i \in \{0, 1\}$;
- Compute $\mathtt{com}' \leftarrow \mathsf{BindCom}(p_0, p_1, v_0, v_1, c; r)$. If $\mathtt{com}' = \mathtt{com}$, return $\mathsf{V}'(x_0, v_0, c, z^*) \wedge \mathsf{V}'(x_1, v_1, c, z^*)$, otherwise output 0.

Figure 2: $(1, 2)$-PTR of [GGHK21] from stackable Σ-protocols.

repetition). Given the composed Σ-protocol $\Pi^{\mathcal{R}_3}$ for proving $\mathcal{R}_3$, we can apply, in turn, an OR composition (e.g. with [GGHK21]) of Σ-protocols for $\mathcal{R}_3$ in order to prove $\mathcal{R}_2$. Then, the protocol for $\mathcal{R}_1$ can be obtained by composing Σ-protocols for $\mathcal{R}_2$ and $\mathcal{R}_3$. The protocol for $\mathcal{R}_{ord'}$ can be obtained by composing (in OR) n Σ-protocols for $\mathcal{R}_1$. Finally, the Σ-protocol Π_{ord} for $\mathcal{R}_{ord}$ can be realized by composing (in AND) k Σ-protocols for $\mathcal{R}_{ord'}$. We remark that $\mathcal{R}_2$ is evaluated only $n-1$ times since when $j=0$ and $\mathcal{R}_2$ is satisfied since in this case there are no parameters pairs to check.

We note that this instantiation is a perfect EHVZK computational stackable Σ-protocols.

On the communication complexity of Π_{ord}. The communication complexity of the protocol to prove $\mathcal{R}_{\mathsf{DL}}$ is $\mathcal{O}(\lambda)$. The complexity of protocol $\Pi_{\mathcal{R}_3}$ realizing the AND composition of two protocols for $\mathcal{R}_{\mathsf{DL}}$ is still $\mathcal{O}(\lambda)$ when using parallel repetition. The same reasoning applies to the OR composition of two instances of $\Pi_{\mathcal{R}_3}$ with [GGHK21]. Now, to realize a proof $\Pi_{\mathcal{R}_1}$ for $\mathcal{R}_1$, parallel repetition shall be applied i times to the AND of the two Σ-protocols $\Pi_{\mathcal{R}_2}$ and $\Pi_{\mathcal{R}_3}$. The communication complexity of $\Pi_{\mathcal{R}_1}$ is $\mathcal{O}(n\lambda)$. $\mathcal{R}_{ord'}$ can be proved by composing in OR n instances of the Σ-protocol $\Pi_{\mathcal{R}_1}$. By using [GGHK21], the protocol $\Pi_{ord'}$ proving $\mathcal{R}_{ord'}$ achieves communication complexity $\mathcal{O}(n\lambda + \lambda \log n) = \mathcal{O}(n\lambda)$. Finally, Π_{ord} can be obtained by repeating $\Pi_{ord'}$ $k-1$ times in parallel, obtaining a communication complexity of $\mathcal{O}((k-1)n\lambda)$.

6.2 Our (k,ℓ)-Proof over Threshold Relations

We now describe our communication-efficient (k,ℓ)-PTR (cfr., Fig. 3) starting from the compiler of [GGHK21] for $(1,\ell)$-PTR, which is obtained by recursively composing ℓ instances of the $(1,2)$-PTR described in Sec. 5, and our communication-efficient Σ-protocol described in the previous section for proving a strict total ordering relation among vectors of commitment parameters. Notice that Π_{ord} is defined over ordered vectors of pairs of commitment parameters. Each $(1,\ell)$-PTR $\Pi^i_{1,\ell}$, for $i \in [k]$, induces a vector $\mathbf{v_i}$ of pairs of commitment parameters. Informally, $\mathbf{v_i}$, for $i \in [k]$, represents the position in which the prover of $\Pi^i_{1,\ell}$ holds the witness. While running Π_{ord}, the prover of the (k,ℓ)-PTR must use $(\mathbf{v_1}, \ldots, \mathbf{v_k})$ already sorted according to the positions of the witnesses used in the k $\Pi_{1,\ell}$ executions. The prover has all this information. Therefore, the prover would just need to sort the k underlying $\Pi_{1,\ell}$ executions according to such indexes when interacting with the verifier.

Let $(\mathsf{P}^{1,\ell}_0, \mathsf{P}^{1,\ell}_1)$ be the prover algorithms of $(1,\ell)$-PTR from [GGHK21]. W.l.o.g., we also assume that the algorithm $\mathsf{P}^{1,\ell}_0$ outputs, together with the first-round message a to be sent to $\mathsf{V}^{1,\ell}$, the tuple of commitment parameters $((p^1_0, p^1_1), \ldots, (p^{\log \ell}_0, p^{\log \ell}_1))$ and the related witnesses $(\mathsf{td}_1, \ldots, \mathsf{td}_{\log \ell})$.

Consider the relation $\mathcal{R}_{\mathsf{k},\ell} = \{((x_1,\ldots,x_\ell),((w_1,\alpha_1),\ldots,(w_k,\alpha_k)))|1 \le \alpha_1 < \ldots < \alpha_k \le \ell \wedge \forall\, j \in [k] : (x_{\alpha_j}, w_j) \in \mathcal{R}_{\mathcal{L}}\}$.

Theorem 1. *Let $\Pi_{1,\ell}$ be the stackable Σ-protocol of [GGHK21], and let Π_{ord} be the stackable Σ-protocol of Sec. 6.1. The protocol $\Pi_{\mathsf{k},\ell} = (\mathsf{P}^{\mathsf{k},\ell}, \mathsf{V}^{\mathsf{k},\ell})$ described in Fig. 3 is a stackable Σ-protocol for relation $\mathcal{R}_{\mathsf{k},\ell}$ with computational special soundness. Furthermore, $\Pi_{\mathsf{k},\ell}$ preserves the EHVZK flavour of the underlying $\Pi_{1,\ell}$.*

We remark that, since the EHVZK flavour of our Π_{ord} instantiation is perfect, Π_{ord} does not affect the EHVZK flavour of $\Pi_{\mathsf{k},\ell}$. We will go through the proof by proving lemmas for completeness, computational special soundness, and EHVZK.

Lemma 1 (Completeness)**.** $\Pi_{\mathsf{k},\ell}$ *is complete.*

Proof. It follows from the completeness of the underlying protocols. □

Before proving that $\Pi_{\mathsf{k},\ell}$ is computational special sound we will first make some observations and introduce an additional lemma that we will use later on in the actual proof.

From now on, we consider the following notion of composition tree $\mathcal{T}$ for an accepting transcript of $\Pi_{1,\ell}$. Given an accepting transcript (a,c,z) of $\Pi_{1,\ell}$, for an instance $(x_1,\ldots,x_\ell)$, where a contains a 1-out-of-2 equivocal commitment to values a_{left} and a_{right}, we label the nodes of $\mathcal{T}$ as follows. Every node of $\mathcal{T}$ is inductively labeled with an instance and an accepting transcript for that instance:

In our (k,ℓ)-PTR $\Pi_{\mathsf{k},\ell} = (\mathsf{P}_0^{\mathsf{k},\ell}, \mathsf{P}_1^{\mathsf{k},\ell}, \mathsf{V}^{\mathsf{k},\ell})$, the prover takes as input a tuple of statements $\mathbf{x} = (x_1, \ldots, x_\ell)$ and k witnesses $\mathbf{w} = ((w_1, \alpha_1), \ldots, (w_k, \alpha_k))$ in which $\alpha_j \in [\ell]$ is the position of the j-th witness. $\Pi_{\mathsf{k},\ell}$ uses the $(1,\ell)$-PTR $\Pi_{1,\ell} = (\mathsf{P}_0^{1,\ell}, \mathsf{P}_1^{1,\ell}, \mathsf{V}^{1,\ell})$, and the proof of parameters ordering $\Pi_{ord} = (\mathsf{P}_0^{ord}, \mathsf{P}_1^{ord}, \mathsf{V}^{ord})$.

First Round: The prover invokes $\mathsf{P}_0^{\mathsf{k},\ell}$ that, on input $(\mathbf{x}, \mathbf{w}; \texttt{rand})$ computes a as follows:

1. Parse $\texttt{rand}$ as $\texttt{rand}_{\mathsf{P}_1} || \ldots || \texttt{rand}_{\mathsf{P}_k} || \texttt{rand}_{ord}$;
2. For all $j \in [k]$: Run $(a_j, \mathbf{p}_j, \mathbf{td}_j) \leftarrow \mathsf{P}_0^{1,\ell}(\mathbf{x}, (w_j, \alpha_j); \texttt{rand}_{\mathsf{P}_j})$, where $\mathbf{p}_j = ((p_0^{(1,j)}, p_1^{(1,j)}), \ldots, (p_0^{(\log \ell, j)}, p_1^{(\log \ell, j)}))$ and $\mathbf{td}_j = (\mathsf{td}^{(1,j)}, \ldots, \mathsf{td}^{(\log \ell, j)})$;
3. Generate $a_{ord} \leftarrow \mathsf{P}_0^{ord}((\mathbf{p}_j)_{j\in[k]}, (\mathbf{td}_j)_{j\in[k]}; \texttt{rand}_{ord})$;

The prover sends $a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ to the verifier.

Second Round: The verifier samples $c \in \{0,1\}^\lambda$ and sends c to the prover.

Third Round: The prover invokes $\mathsf{P}_1^{\mathsf{k},\ell}$ that computes z as follows: For each $j \in [k]$ run $z_j \leftarrow \mathsf{P}_1^{1,\ell}(\mathbf{x}, (w_j, \alpha_j), c; \texttt{rand}_{\mathsf{P}_j})$ and $z_{ord} \leftarrow \mathsf{P}_1^{ord}((\mathbf{p}_j)_{j\in[k]}, (\mathbf{td}_j)_{j\in[k]}, c; \texttt{rand}_{ord})$;
Then, the prover sends $z = (z_1, \ldots, z_k, z_{ord})$ to the verifier.

Verification: The verifier invokes $\mathsf{V}^{\mathsf{k},\ell}$ that, on input $(\mathbf{x}, a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k), c, z = (z_1, \ldots, z_k, z_{ord}))$, returns a bit b as follows:

1. For $i \in [k]$ and $j \in [\log \ell]$ check that $(p_0^{(i,j)}, p_1^{(i,j)}) \in \mathcal{Y}^{\mathsf{pp}}$, where the pairs $(p_0^{(i,j)}, p_1^{(i,j)})$ are taken from z_i;
2. Check that $\mathbf{p}_i = (p_0^{(i,j)}, p_1^{(i,j)})_{j\in[\log \ell]}$;
3. For all $i \in [k]$ check that $\mathsf{V}^{1,\ell}(\mathbf{x}, a_i, c, z_i) = 1$;
4. Check that $\mathsf{V}^{ord}((\mathbf{p}_j)_{j\in[k]}, a_{ord}, c, z_{ord}) = 1$;
5. If all the previous checks are successful, output 1. Otherwise, output 0.

Figure 3: Our communication-efficient (k,ℓ)-PTR from stackable Σ-protocols.

- The root is labeled with $\Sigma_{\mathsf{root}} = ((x_1, \dots, x_\ell), a, c, z)$.
- Given a node m at level q, with $q \in [\log \ell]$, labeled with $\Sigma_m = ((x_i, \dots, x_j), a_m, c, z_m = (\tilde{z}_m, r_m, (p_0^q, p_1^q)))$, for $1 \le i < j \le \ell$, the left child node of m is at level $q+1$ and is labeled with $\Sigma_{\mathsf{left}} = ((x_i, \dots, x_{(j+i-1)/2}), a_{\mathsf{left}}, c, \tilde{z}_m)$, and the right child node is at level $q+1$ and is labeled with $\Sigma_{\mathsf{right}} = ((x_{((j+i+1)/2)+1}, \dots, x_j), a_{\mathsf{right}}, c, \tilde{z}_m)$[20], where a_{left} is the first-round message of the transcript for statement $(x_i, \dots, x_{(j+i-1)/2})$ and a_{right} is the first-round message of the transcript for statement $(x_{((j+i+1)/2)+1}, \dots, x_j)$.

Moreover, for each node m at level q in $\mathcal{T}$ labeled with $\Sigma_m = ((x_i, \dots, x_j), a_m, c, z_m = (\tilde{z}_m, r_m, (p_0^q, p_1^q)))$, the edge going from m to its left child is labeled with p_0^q and the edge from m to its right child is labeled with p_1^q. We say that the edges from m to their children are edges at level q.

Let (a, c, z) be an accepting transcript for $\Pi_{1,\ell}$, we notice that $\mathbf{p} = ((p_0^1, p_1^1), \dots, (p_0^{\log \ell}, p_1^{\log \ell}))$ contained in z represents the labeling of the edges of the composition tree $\mathcal{T}$ for (a, c, z). Indeed, each node m in level q, for $q \in [\log \ell]$, has the edge to the left child labeled with p_0^q and the edge to the right child labeled with p_1^q. We denote with $\mathcal{T}^{\mathbf{p}}$ the composition tree with edge labeling defined by $\mathbf{p}$. An example of a composition tree is illustrated in Fig. 1.

The extractor of $\Pi_{1,\ell}$ in terms of composition trees. Let us now review how $\mathsf{Extract}_{\Pi_{1,\ell}}$, the extractor of $\Pi_{1,\ell}$ proposed in [GGHK21], works in terms of composition trees. $\mathsf{Extract}_{\Pi_{1,\ell}}$ takes in input two composition trees whose roots have accepting transcripts with the same first-round massage and different challenges. Due to stackability, we can look at every node of a composition tree as representing the transcript for the Σ-protocol for an OR relation on a subset of all the ℓ involved statements. The root of the tree Σ_{root} represents the transcript for the Σ-protocol for the whole relation $\mathcal{R}_{1,\ell}$. For each node with statements $(x_i, \dots, x_j)$, its left child Σ_{left} and right child Σ_{right} represent the Σ-protocol transcripts for a relation $\mathcal{R}_{1,(\mathsf{j-i+1})/2}$ on the first and second half of the subset of instances respectively. It follows that the leaves represent the Σ-protocols transcripts for the individual instances x_i with $i \in [\ell]$. $\mathsf{Extract}_{\Pi_{1,\ell}}$ is a recursive composition of the extractor $\mathsf{Extract}_{\Pi_{1,2}}$ of the underlying $\Pi_{1,2}$ (see App. C for details).

Consider the following two accepting transcripts Σ^1_{root} and Σ^2_{root} (with associated composition trees $\mathcal{T}^1$ and $\mathcal{T}^2$) having the same first-round message $a_{\mathsf{root}} = (\mathtt{com}, p_0, p_1)$, $c^1 \ne c^2$, $z^1 = (\tilde{z}^1, r, p_0, p_1)$, $z^2 = (\tilde{z}^2, r, p_0, p_1)$. Given these accepting transcripts, it is possible to compute transcripts for Σ_{left} and Σ_{right} as follows: $a^1_{\mathsf{left}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{left}}(x_{\mathsf{left}}, c^1, \tilde{z}^1)$, $a^1_{\mathsf{right}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{right}}(x_{\mathsf{right}}, c^1, \tilde{z}^1)$, $a^2_{\mathsf{left}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{left}}(x_{\mathsf{left}}, c^2, \tilde{z}^2)$, $a^2_{\mathsf{right}} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{right}}(x_{\mathsf{right}}, c^2, \tilde{z}^2)$, where $x_{\mathsf{left}} = (x_1, \dots, x_{\ell/2})$ and $x_{\mathsf{right}} = (x_{\ell/2+1}, \dots, x_\ell)$. Therefore, whenever $a^1_{\mathsf{left}} = a^2_{\mathsf{left}}$ or $a^1_{\mathsf{right}} = a^2_{\mathsf{right}}$ it is possible to call again $\mathsf{Extract}_{\Pi_{1,2}}$ on the two composition sub-trees of $\mathcal{T}^1$ and $\mathcal{T}^2$ rooted at the respective children nodes that have the same-first round messages (i.e., either the left or the right children nodes). For each pair of nodes in $\mathcal{T}^1$ and $\mathcal{T}^2$ having the same first-round message, looking again at the children nodes, either a^1_{left} is equal to a^2_{left} or a^1_{right} is equal to a^2_{right} with overwhelming probability, otherwise it is possible to break the computational special soundness of $\Pi_{1,2}$ and thus the computational special soundness of $\Pi_{1,\ell}$ (see App. C for details). Therefore, the latter leads to at least one extracted witness for an instance x_j with $j \in [\ell]$. In general, up to ℓ witnesses could be extracted.

Trapdoor equivalence class. We now introduce the concept of trapdoor equivalence class. Informally, a trapdoor equivalence class identifies all vectors of trapdoors having the same trapdoor (and non-trapdoor) position for all parameters pairs.

Definition 7 (Trapdoor Equivalence Class). *Let $\mathbf{p} = ((p_0^1, p_1^1), \dots, (p_0^n, p_1^n))$ be a vector of parameters of a 1-out-of-2 equivocal commitment scheme (see Def. 6), and let $\mathcal{R}_T$ be its associated poly-time relation. Let $\mathbf{td} = (\mathsf{td}_1, \dots, \mathsf{td}_n)$ be a vector of trapdoors such that for every $i \in [n]$, there exists $\beta \in \{0,1\}$ such that $\mathcal{R}_T(p^i_\beta, \mathsf{td}_i) = 1$. The trapdoor equivalence class $[\mathbf{td}]^{\mathcal{R}_T}_{\mathbf{p}}$ is the set containing all vectors $\mathbf{td}' = (\mathsf{td}'_1, \dots, \mathsf{td}'_n)$ in which for every $i \in [n]$, and $\beta \in \{0,1\}$, $\mathcal{R}_T(p^i_\beta, \mathsf{td}_i) = \mathcal{R}_T(p^i_\beta, \mathsf{td}'_i)$.*

[20]We recall that $\tilde{z}_m$ contains the commitment parameters used to generate the first-round message of the children of node m. Therefore, all the children of node m use the same commitment parameters pair.

We now define the notion of *valid* trapdoor equivalence class. Informally, a trapdoor equivalence class is said to be valid w.r.t. two composition trees with the same edge labels if there is no level in such trees where a trapdoor is related to a parameter (e.g., in the first position) while an equivocation is performed in the position of the other parameter (e.g., the second position).

Definition 8 (Valid Trapdoor Equivalence Classes)**.** *Let* $\mathbf{p} = ((p_0^1, p_1^1), \ldots, (p_0^{\log \ell}, p_1^{\log \ell}))$ *be a vector of parameters of a 1-out-of-2 equivocal commitment scheme (see Def. 6) with associated poly-time relation* $\mathcal{R}_T$*. Let* (a, c, z) *and* (a, c', z')*, with* $c \neq c'$*, be two accepting transcripts of* $\Pi_{1,\ell}$ *for the same instance* $x = (x_1, \ldots, x_\ell)$*, and* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *be the composition trees associated with those accepting transcripts. Let us take* $(\mathsf{td}_1, \ldots, \mathsf{td}_{\log \ell}) \in [\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$*. A trapdoor equivalence class* $[\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$ *is valid w.r.t.* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *if, for every level* $i \in [\log \ell]$ *(of both* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$*), there does not exist a node* m *at level* i *in which*

$$((\mathcal{R}_T(p_0^i, \mathsf{td}_i) = 1) \wedge (a_{\mathsf{right}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{right}})) \vee ((\mathcal{R}_T(p_1^i, \mathsf{td}_i) = 1) \wedge (a_{\mathsf{left}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{left}}))$$

where $a_{\mathsf{left}}^{(i,m)}$ *(resp.* $a_{\mathsf{right}}^{(i,m)}$*) is the first-round message associated to the left (resp. right) child of node* m *which is at level* i *of* $\mathcal{T}^{\mathbf{p}}$*. The same holds w.r.t* $a'^{(i,m)}_{\mathsf{left}}$*,* $a'^{(i,m)}_{\mathsf{right}}$*, and* $\mathcal{T}'^{\mathbf{p}}$*.*

Lemma 2. *Let* (a, c, z) *and* (a, c', z') *be two transcripts for* $\Pi_{1,\ell}$ *for the same statement* $(x_1, \ldots, x_\ell)$ *and* $c \neq c'$*. Let* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *be the two composition trees associated with* (a, c, z) *and* (a, c', z') *respectively. Let* CS *be the 1-out-of-2 equivocal commitment scheme with associated relation* $\mathcal{R}_T$ *used in* $\Pi_{1,\ell}$*. If* $\mathsf{Extract}_{\Pi_{1,\ell}}$ *extracts* s *different witnesses, then the number of valid trapdoor equivalence classes* $[\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$ *w.r.t.* $\mathcal{T}^{\mathbf{p}}$ *and* $\mathcal{T}'^{\mathbf{p}}$ *is upper-bounded by* s*.*

Proof. Let i be a level of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$ and t be the number of *valid* trapdoor equivalence classes $[\mathbf{td}]_{\mathbf{p}}^{\mathcal{R}_T}$ (see Def. 8). Let us call $\mathbf{td}^j = (\mathsf{td}_1^j, \ldots, \mathsf{td}_{\log \ell}^j)$ the representative of the j-th valid trapdoor equivalence class $[\mathbf{td}^j]_{\mathbf{p}}^{\mathcal{R}_T}$ for $j \in [t]$. We define L_i, E_i and s_i as follows:

- $L_i = \left\lceil \frac{\sum_{j \in [t]} (\mathcal{R}_T(p_0^i, \mathsf{td}_i^j))}{t} \right\rceil + \left\lceil \frac{\sum_{j \in [t]} (\mathcal{R}_T(p_1^i, \mathsf{td}_i^j))}{t} \right\rceil$. L_i can either be 1 or 2. L_i is 1 if, at level i, either $\mathcal{R}_T(p_0^i, \mathsf{td}_i^j) = 1$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^k) = 0$ or $\mathcal{R}_T(p_0^i, \mathsf{td}_i^k) = 0$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^j) = 1$ for some $j, k \in [t]$. Otherwise, L_i is 2.

- $E_i = \prod_{j=1}^{i} L_j$ E_i represents the number of valid equivalence classes for the sub-trees of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$ having only the first i levels. $E_{\log \ell}$ is equal to t.

- s_i represents the number of witnesses that can be extracted from the i-th level of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$. It is trivial to see that $s_i \geq s_{i-1}$ for every $i \in [\log \ell]$.

In the following, we use $a^{(i,m)}$ to indicate the first-round message contained in the node m at level i in composition tree $\mathcal{T}^{\mathbf{p}}$. Additionally,"$a_{\mathsf{left}}^{(i,m)}$" indicatse the first-round message contained in the left child of node m at level i of composition tree $\mathcal{T}^{\mathbf{p}}$. The pedix "right" indicates a right child node. The apex " $'$ " is added to indicate that a node is from composition tree $\mathcal{T}'^{\mathbf{p}}$.

We prove the lemma by induction. For every level $i \in [\log \ell]$, $E_i \leq s_i$.

Base case: Let us consider the root m of $\mathcal{T}^{\mathbf{p}}$ and $\mathcal{T}'^{\mathbf{p}}$. By inspection, it is clear that if $s_0 = 1$ then either $a_{\mathsf{left}}^{(0,m)} = a'^{(0,m)}_{\mathsf{left}}$ or $a_{\mathsf{right}}^{(0,m)} = a'^{(0,m)}_{\mathsf{right}}$ but not both at the same time. Indeed, in this last case s_0 would be equal to 2. Therefore, fixed one of the two cases above, there exists only one *valid* equivalence class, thus $L_0 = 1$. Let us assume that $a_{\mathsf{left}}^{(0,m)} = a'^{(0,m)}_{\mathsf{left}}$ and $a_{\mathsf{right}}^{(0,m)} \neq a'^{(0,m)}_{\mathsf{right}}$. Indeed, if $L_0 = 2$ and $s_0 = 1$, then there exists td_1^j and td_1^k from two different equivalence classes with representative td^j and td^k respectively such that either $\mathcal{R}_T(p_0, \mathsf{td}_1^j) = 1$ or $\mathcal{R}_T(p_0, \mathsf{td}_1^k) = 1$ violating the validity property of Def. 8. If $s_1 = 2$ the base case is trivially true since $L_1 \leq 2$. Therefore, in the base case it holds that $E_2 = L_2 \leq s_2$.

Inductive case: We now show that $E_{i-1} \leq s_{i-1}$ implies $E_i \leq s_i$. We notice that for increasing values of i the values of E_i and s_i are monotone non-decreasing. Indeed, E_i is a product of non-zero integers. Regarding s_i, extracting a witness from a node at level $i-1$ requires extracting a witness from at least one of its child nodes at level i. Therefore, if $E_i = E_{i-1}$ then $E_i \leq s_i$ trivially follows from $E_{i-1} \leq s_{i-1}$. We now show that if $E_i = 2E_{i-1}$, then $s_i \geq 2s_{i-1}$. If $E_i = 2E_{i-1}$ then it holds that $L_{i-1} = 2$ which means that there exists td_i^j and td_i^k in equivalence classes with representatives td^j and td^k satisfying either $\mathcal{R}_T(p_0^i, \mathsf{td}_i^j) = 1$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^k) = 1$ or $\mathcal{R}_T(p_0^i, \mathsf{td}_i^k) = 1$ and $\mathcal{R}_T(p_1^i, \mathsf{td}_i^j) = 1$. Let us assume by contradiction that $s_i < 2s_{i-1}$. In this case there must be at least a witness at level $i-1$ such that $a_{\mathsf{left}}^{(i,m)} = a'^{(i,m)}_{\mathsf{left}}$ and $a_{\mathsf{right}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{right}}$ or $a_{\mathsf{right}}^{(i,m)} = a'^{(i,m)}_{\mathsf{right}}$ and $a_{\mathsf{left}}^{(i,m)} \neq a'^{(i,m)}_{\mathsf{left}}$ for some node m. From the above observation, it follows that at least one of the trapdoor equivalence classes does not satisfy the validity requirement of Def. 8. Then, it holds that $s_i \geq 2s_{i-1}$, from which it follows that $E_i \leq s_i$.

□

Lemma 3 (Computational Special Soundness). *$\Pi_{\mathsf{k},\ell}$ is computational special sound.*

Proof. Computational special soundness follows from the computational special soundness of $\Pi_{1,\ell}$, the computational special soundness of Π_{ord}, and the fixed equivocation property of CS. To prove computational special soundness of $\Pi_{\mathsf{k},\ell}$ we define the extractor $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ based on the extractor $\mathsf{Extract}_{\Pi_{1,\ell}}$ of $\Pi_{1,\ell}$ and the extractor $\mathsf{Extract}_{\Pi_{ord}}$ of Π_{ord} as follows.

Given two accepting transcripts with the same first-round message $a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$, $c \neq c'$, $z = (z_1, \ldots, z_k, z_{ord})$, and $z' = (z'_1, \ldots, z'_k, z'_{ord})$ for statement $(x_1, \ldots, x_\ell)$, $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ runs, for each $i \in [k]$, the extractor of each $(1,\ell)$-PTR, i.e. $\mathsf{Extract}_{\Pi_{1,\ell}}((x_1 \ldots, x_\ell), a_i, c, c', z_i, z'_i)$.

From each execution of $\mathsf{Extract}_{\Pi_{1,\ell}}((x_1 \ldots, x_\ell), a_i, c, c', z_i, z'_i)$, $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ obtains with overwhelming probability at least a witness (w_1^i, j_1^i), otherwise it is possible to break the computational special soundness of $\Pi_{1,\ell}$. Indeed, (a_i, c, z_i) and (a_i, c', z'_i) are two accepting transcripts for statement $(x_1 \ldots, x_\ell)$ for $\Pi_{1,\ell}$ otherwise $\mathsf{V}^{\mathsf{k},\ell}$ cannot accept the transcripts (a, c, z) and (a, c', z'). If $\mathsf{Extract}_{\Pi_{1,\ell}}((x_1 \ldots, x_\ell), a_i, c, c', z_i, z'_i)$ does not return a valid witness, then $\mathsf{ExpExt}_{\mathsf{P}^{1,\ell*}, \mathsf{Extract}_{\Pi_{1,\ell}}}(x_1, \ldots, x_\ell)$ returns 1, that happens only with negligible probability.

Let us consider the case in which, for all $i \in [k]$, a set of witnesses $\mathbf{w}^i = \{(w_1^i, j_1^i), \ldots, (w_k^i, j_{f^i}^i)\}$ with $f^i < k$ is extracted from each $(1,\ell)$-PTR. Recall that for $u \in [f^i]$, each index $j_u^i \in [\ell]$ simply specifies the base instance the extracted witness corresponds to. Additionally, consider $I^i = \{j_1^i, \ldots, j_{f^i}^i\}$ and the case for which $|\bigcup_{i=1}^k I^i| < k$. Namely, this is the case in which less than k witnesses for different statements in $(x_1, \ldots, x_\ell)$ are extracted from the (k,ℓ)-PTR. We only need to consider this case since otherwise we would have already extracted a witness for relation $\mathcal{R}_{\mathsf{k},\ell}$. This case implies that from each $(1,\ell)$-PTR less than k witnesses are extracted, since the witness extracted by $\mathsf{Extract}_{\Pi_{1,\ell}}$ are always related to different elementary statements by construction.

We run $\mathsf{Extract}_{\Pi_{ord}}$ on input $((\mathbf{p}_i)_{i \in [k]}, a_{ord}, c, c', z_{ord}, z'_{ord})$ obtaining, with overwhelming probability, a witness for $(\mathbf{p}_i)_{i \in [k]}$ being in $\mathcal{R}_{ord}$. Otherwise, it is possible to break the computational special soundness of Π_{ord}. The reduction follows the same blueprint of the one shown for the computational special soundness of $\Pi_{1,\ell}$. We now argue that there must be two composition trees $\mathcal{T}^{\mathbf{p}_i}$ and $\mathcal{T}'^{\mathbf{p}_i}$, with $i \in [k]$, in which there exists a node m at level $q \in [\log \ell]$ in which $\mathcal{R}_T(p_0^q, \mathsf{td}_q^i) = 1$, $a_{\mathsf{left}}^{(q,m)} = a'^{(q,m)}_{\mathsf{left}}$ and $a_{\mathsf{right}}^{(q,m)} \neq a'^{(q,m)}_{\mathsf{right}}$ (or equally $\mathcal{R}_T(p_1^q, \mathsf{td}_q^i) = 1$, $a_{\mathsf{right}}^{(q,m)} = a'^{(q,m)}_{\mathsf{right}}$, and $a_{\mathsf{left}}^{(q,m)} \neq a'^{(q,m)}_{\mathsf{left}_q}$), where td_q^i is an element of the vector $\mathbf{td}^i = (\mathsf{td}_1^i, \ldots, \mathsf{td}_{\log \ell}^i)$ related to $\mathbf{p}_i$, and $\mathbf{td}^i$ was extracted using $\mathsf{Extract}_{\Pi_{ord}}$. Namely, $\mathbf{td}^i$ is a representative of a trapdoor equivalence class which is not valid according to Def. 8. Indeed, by Lemma 2 the number of valid trapdoor equivalence classes is upper-bounded by the number of extracted witnesses, which is strictly less than k in this case. Nevertheless, thanks to the computational special soundness of Π_{ord}, $\mathsf{Extract}_{\Pi_{\mathsf{k},\ell}}$ extracts, with overwhelming probability, k vectors of trapdoors, all belonging to different trapdoor equivalence classes[21]. Therefore, one of such vectors must belong to a non-valid trapdoor equivalence class,

[21]Indeed, this is the requirement imposed by the relation $\mathcal{R}_{ord}$.

thus allowing the following reduction. Consider the transcripts associated to node m at level q in both $\mathcal{T}^{\mathbf{p}_i}$ and $\mathcal{T}'^{\mathbf{p}_i}$. They are of the form $(a^{(q,m)}, c, z), (a'^{(q,m)}, c', z')$ with $a^{(q,m)} = a'^{(q,m)}$, $c \neq c'$, $z = (z^*, r, p_0^q, p_1^q)$, and $z' = (z^{*\prime}, r', p_0^q, p_1^q)$. We now use these two accepting transcripts to break the fixed equivocation property of the 1-out-of-2 equivocal commitment scheme. $\mathcal{A}$ does the following:

- $(m_0, m_0', m_1) \leftarrow\$ \{0,1\}^{3\lambda}$ with $m_0 \neq m_0'$.
- $(\mathtt{com}', \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta = 1, m_1, p_0^q, p_1^q, \mathsf{td}_q^i)$;
- $r^* \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta = 1, m_0, m_1, p_0^q, p_1^q, \mathsf{td}_q^i, \mathsf{aux})$
- $r^{*\prime} \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta = 1, m_0', m_1, p_0^q, p_1^q, \mathsf{td}_q^j, \mathsf{aux})$

$\mathcal{A}$ outputs $(p_0^q, p_1^q, r, r', r^*, r^{*\prime}, a_{\mathsf{left}}^{(q,m)}, a_{\mathsf{left}}'^{(q,m)}, a_{\mathsf{right}}^{(q,m)}, a_{\mathsf{right}}'^{(q,m)}, m_0, m_0', m_1, m_1)$.

Thanks to the partial equivocation property of the commitment scheme, $\mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, m_0, m_1, r^*) = \mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, m_0, m_1', r^{*\prime})$. Additionally, since the two transcripts are accepting, we have that $\mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, a_{\mathsf{left}}^{(q,m)}, a_{\mathsf{right}}^{(q,m)}, r) = \mathsf{BindCom}(\mathsf{pp}, p_0^q, p_1^q, a_{\mathsf{left}}'^{(q,m)}, a_{\mathsf{right}}'^{(q,m)}, r')$. Therefore, $\mathcal{A}$ breaks the fixed equivocation property with the same probability that less than k witnesses for different elementary statements are extracted, thus reaching a contradiction. □

Lemma 4 (Extended Honest-Verifier Zero Knowledge). $\Pi_{\mathsf{k},\ell}$ *is Extended Honest-Verifier Zero Knowledge.*

Proof. Let $D_{x,c}^{(z)^*}$ be the third-round message distribution of $\Pi_{1,\ell}$, and $D_c^{(z)'}$ be the third-round message distribution of Π_{ord}. Let $\mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}$ be the EHVZK simulator of $\Pi_{1,\ell}$, for $i \in [k]$, and $\mathcal{S}_{ord}^{\mathsf{EHVZK}}$ be the EHVZK simulator of Π_{ord}. Let $D_c^{(z)} = \left\{(z_1, \ldots, z_k, z_{ord}) | \, \forall i \in [k] \; z_i \leftarrow\$ D_c^{(z)^*}, z_{ord} \leftarrow\$ D_c^{(z)'}\right\}$.

We define simulator $\mathcal{S}_{\mathsf{k},\ell}^{\mathsf{EHVZK}}(\mathbf{x} = (x_1, \ldots, x_\ell), c, (z_1, \ldots, z_k, z_{ord}))$ as follows:

1. Compute $a_i \leftarrow \mathcal{S}_{1,\ell}^{\mathsf{EHVZK}}(\mathbf{x}, c, z_i)$, for all $i \in [k]$;
2. Parse $z_i = (\tilde{z}_j, \{r_j\}_{j \in [\log \ell]}, \mathbf{p}_i)$, for all $i \in [k]$;
3. Compute $a_{ord} \leftarrow \mathcal{S}_{ord}^{\mathsf{EHVZK}}((\mathbf{p}_1, \ldots, \mathbf{p}_k), c, z_{ord})$;
4. Return $(a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$.

We prove that $\Pi_{\mathsf{k},\ell}$ is EHVZK with the following hybrid arguments.

$\mathcal{H}_0$: this is equal to the real game with honest prover, except that the prover of hybrid $\mathcal{H}_0$ takes in input $(\mathbf{x}, \mathbf{w}, c, (z_1, \ldots, z_k, z_{ord}))$, where $z_i \in D_c^{(z)}$ for all $i \in [k]$ and $z_{ord} \in D_c^{(z)'}$. The additional inputs c and $(z_1, \ldots, z_k, z_{ord})$ are ignored during the execution by the prover of hybrid $\mathcal{H}_0$. We notice that $\mathcal{H}_0$ is distributed identically to the real game.

$\mathcal{H}_{1_0}$: It is identical to $\mathcal{H}_0$ except that a_{ord} is computed using $\mathcal{S}_{ord}^{\mathsf{EHVZK}}((\mathbf{p}_1, \ldots, \mathbf{p}_k), c, z_{ord})$ where $(\mathbf{p}_1, \ldots, \mathbf{p}_k)$, c and z_{ord} are taken from the prover's additional input specified in $\mathcal{H}_0$. Recall that z_i, for $i \in [k]$, contains also $\mathbf{p}_i$. If there exists a distinguisher $\mathcal{D}_{EHVZK}$ that distinguishes $\mathcal{H}_0$ from $\mathcal{H}_{1_0}$ with non-negligible advantage, we can construct a distinguisher $\mathcal{D}_{ord}$ that distinguishes an execution of $\mathcal{S}_{ord}^{\mathsf{EHVZK}}$ from an execution of Π_{ord} with the same advantage. The reduction works as follows:

- $\mathcal{D}_{ord}$ samples randomness $\mathtt{rand}$, computes $(a_j, \mathbf{td}_j) \leftarrow \mathsf{P}_0^{1,\ell}(\mathbf{x}, (w_j, i_j); \mathtt{rand})$ and $z_j = (\tilde{z}_j, \{r_i\}_{i \in [\log \ell]}, \mathbf{p}_j) \leftarrow \mathsf{P}_1^{1,\ell}(\mathbf{x}, (w_j, i_j), c; \mathtt{rand})$, for all $j \in [k]$. Then, $\mathcal{D}_{ord}$ outputs the statement/witness pair $((\mathbf{p}_1, \ldots, \mathbf{p}_k), (\mathbf{td}_1, \ldots, \mathbf{td}_k))$ in the experiment (cfr., Sec. 3.3) $\mathsf{ExpEHVZK}_{\mathsf{P}', \mathcal{D}_{ord}}(c)$ (where P' is either $(\mathsf{P}_0^{ord}, \mathsf{P}_1^{ord})$ or $\mathcal{S}_{ord}^{\mathsf{EHVZK}}$) receiving back (a_{ord}, z_{ord}).
- $\mathcal{D}_{ord}$, on input $((\mathbf{p}_1, \ldots, \mathbf{p}_k), (\mathbf{td}_1, \ldots, \mathbf{td}_k), a_{ord}, c, z_{ord})$, forwards $(\mathbf{x}, \mathbf{w}, a, c, z)$ to $\mathcal{D}_{EHVZK}$, where $a = (a_1, \ldots, a_k, a_{ord}, \mathbf{p}_1, \ldots, \mathbf{p}_k)$ and $z = (z_1, \ldots, z_k, z_{ord})$. $\mathcal{D}_{ord}$ outputs whatever $\mathcal{D}_{EHVZK}$ outputs.

$\mathcal{H}_{1_i}$**:** for each $i \in [k]$, this is equal to $\mathcal{H}_{1_{i-1}}$ except that all values $a_1, \dots, a_i$ are computed using $\mathcal{S}^{\mathsf{EHVZK}}_{1,\ell}(\mathbf{x}, c, z_i)$ where c and z_i are taken from the prover's additional input specified in $\mathcal{H}_0$.

If there exists a distinguisher $\mathcal{D}_{EHVZK}$ that distinguishes $\mathcal{H}_{1_i}$ from $\mathcal{H}_{1_{i+1}}$ with non-negligible advantage, we can construct a distinguisher $\mathcal{D}_{1,\ell}$ that distinguishes a simulated execution $\mathcal{S}^{\mathsf{EHVZK}}_{1,\ell}$ from an execution of $\Pi_{1,\ell}$ with the same advantage. The reduction works as follows:

- $\mathcal{D}_{1,\ell}$ samples randomness $\mathtt{rand}$, $\mathcal{D}_{1,\ell}$ computes $(a_j, \mathbf{td}_j) \leftarrow \mathsf{P}^{1,\ell}_0(\mathbf{x}, (w_j, i_j); \mathtt{rand})$ and $z_j = (\tilde{z}_j, \{r_i\}_{i \in [\log \ell]}, \mathbf{p}_j) \leftarrow \mathsf{P}^{1,\ell}_1(\mathbf{x}, (w_j, i_j), c; \mathtt{rand})$, for each $j < i$. Then, $\mathcal{D}_{ord}$ samples $z_j \leftarrow\$ D^z_c$ and computes $a_j \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{1,\ell}(\mathbf{x}, c, z_j)$ for each $j > i$. The value a_{ord} is computed from $\mathcal{S}^{\mathsf{EHVZK}}((\mathbf{p}_1, \dots, \mathbf{p}_k), c, z_{ord})$ for $z_{ord} \leftarrow\$ D^{(z)'}_c$. Then $\mathcal{D}_{1,\ell}$, outputs the statement/witness pair $(\mathbf{x}, w_i)$ in the experiment $\mathsf{ExpEHVZK}_{\mathsf{P}', \mathcal{D}_{1,\ell}}(c)$ (where P' is either $(\mathsf{P}^{1,\ell}_0, \mathsf{P}^{1,\ell}_1)$ or $\mathcal{S}^{\mathsf{EHVZK}}_{1,\ell}$), receiving back (a_i, z_i).
- $\mathcal{D}_{1,\ell}$, on input $(\mathbf{x}, w_i, a_i, c, z_i)$, sends $(\mathbf{x}, \mathbf{w}, a, c, z)$ to $\mathcal{D}_{EHVZK}$, where $a = (a_1, \dots, a_k, a_{ord}, \mathbf{p}_1, \dots, \mathbf{p}_k)$ and $z = (z_1, \dots, z_k, z_{ord})$. $\mathcal{D}_{1,\ell}$ outputs whatever $\mathcal{D}_{EHVZK}$ outputs.

$\mathcal{H}_2$**:** this is equal to $\mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{k},\ell}(\mathbf{x}, c, (z_1, \dots, z_k, z_{ord}))$. $\mathcal{H}_2$ is identical to $\mathcal{H}_{1_k}$.

□

7 Conclusion

In this work, we proposed an efficient and modular transformation that starting from stackable Σ-protocols and a corresponding threshold relation $\mathcal{R}_{\mathsf{k},\ell}$, provides an efficient Σ-protocol for $\mathcal{R}_{\mathsf{k},\ell}$ with communication complexity $\mathcal{O}(k(\mathsf{CC}(\Sigma) + \lambda \log \ell))$, where $\mathsf{CC}(\Sigma)$ is the communication complexity of the base Σ-protocol. Moreover, our transformation preserves statistical/perfect honest-verifier zero knowledge.

Previous approaches broke the linear barrier of communication in ℓ only for specific languages and thus, even if producing more compact proofs, are for from being a general tool to be used as a building block for a variety of protocols. Interestingly, the flexibility of our approach also allows to get other primitives such as threshold ring signatures from a variety of assumptions depending on the underlying Σ-protocol. Our approach is based on the $(1, \ell)$-PTR of [GGHK21], equipped with an efficient proof system about the equivocal commitment scheme used in the $(1, \ell)$-PTR. Our technique is not tied to a specific instantiation but it can be adapted to different instantiations, as soon as there exists a relation with associated Σ-protocol for a parameter being equivocal. Although we broke the linear barrier in ℓ of the communication complexity of (k, ℓ)-PTR for a vast class of Σ-protocols, removing the linear dependency in k for a vast class of Σ-protocols is still an interesting open problem.

Acknowledgements. We thank all the anonymous reviewers for their insightful comments. In particular, we thank an anonymous reviewer of CRYPTO 2022 for pointing out a possible alternative approach.

References

[AC20] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, pages 513–543, 2020.

[ACF21] Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, pages 65–91, 2021.

[ACR21] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 526–556. Springer, 2021.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Advances in Cryptology — ASIACRYPT 2001*, pages 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[BMRS21] Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, pages 92–122, 2021.

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer, 2002.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[CFF$^+$21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zksnarks and commit-and-prove extensions. In *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2021.

[CFQ19] Matteo Campanelli, Dario Fiore, and Anaïs Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2075–2092. ACM, 2019.

[CHGa19] Jiangshan CHEN, Yupu HU, Wen GAO, and Hongmei Liang and. Lattice-based threshold ring signature with message block sharing. *KSII Transactions on Internet and Information Systems*, 13(2):1003–1019, February 2019.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zksnarks with universal and updatable SRS. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 738–768. Springer, 2020.

[CPS$^+$16] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Online/offline OR composition of sigma protocols. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 63–92. Springer, 2016.

[GGHK21] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking sigmas: A framework to compose Σ-protocols for disjunctions. *IACR Cryptol. ePrint Arch.*, 2021:422, 2021. Version accessed 09 November 2021.

[GGHK22] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking Sigmas: A Framework to Compose Σ-Protocols for Disjunctions. In *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 458–487. Springer, 2022.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In *Advances in Cryptology - EUROCRYPT 2015*, pages 253–280, 2015.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, page 953, 2019.

[HK20] David Heath and Vladimir Kolesnikov. Stacked garbling for disjunctive zero-knowledge proofs. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 569–598. Springer, 2020.

[HKP21] David Heath, Vladimir Kolesnikov, and Stanislav Peceny. Garbling, stacked and staggered - faster k-out-of-n garbled function evaluation. In *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 245–274. Springer, 2021.

[HKSS20] Abida Haque, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Logarithmic-size (linkable) threshold ring signatures in the plain model. *IACR Cryptol. ePrint Arch.*, 2020:683, 2020.

[HS20] Abida Haque and Alessandra Scafuro. Threshold ring signatures: New definitions and post-quantum security. In *Public-Key Cryptography – PKC 2020*, pages 423–452, Cham, 2020. Springer International Publishing.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 955–966. ACM, 2013.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2111–2128. ACM, 2019.

[MOY21] Alexander Munch-Hansen, Claudio Orlandi, and Sophia Yakoubov. Stronger notions and a more efficient construction of threshold ring signatures. In *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*, volume 12912 of *Lecture Notes in Computer Science*, pages 363–381. Springer, 2021.

[OTYO18] Takeshi Okamoto, Raylin Tso, Michitomo Yamaguchi, and Eiji Okamoto. A k-out-of-n ring signature with flexible participation for signers. Cryptology ePrint Archive, Report 2018/728, 2018. https://ia.cr/2018/728.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252, 1989.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *Annual International Cryptology Conference*, pages 704–737. Springer, 2020.

[YLA$^+$13] Tsz Hon Yuen, Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Efficient linkable and/or threshold ring signature without random oracles. *The Computer Journal*, 56(4):407–421, 2013.

[ZZY$^+$17] Guomin Zhou, Peng Zeng, Xiaohui Yuan, Siyuan Chen, and Kim-Kwang Raymond Choo. An efficient code-based threshold ring signature scheme with a leader-participant model. *Security and Communication Networks*, 2017:1915239, Aug 2017.

A Standard Tools

A.1 Discrete Logarithm (DL) Assumption

Assumption 1 (DL)**.** *There exists a PPT algorithm* $\mathsf{GG}(1^\lambda)$ *which returns the description of a cyclic group* $\mathbb{G}$ *in which the sampling of the elements is efficient, such that for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* $\nu(\lambda)$*:*

$$\text{Prob}\left[\ \mathcal{A}(1^\lambda, \mathbb{G}, g, h) = y | \mathbb{G} \leftarrow \mathsf{GG}(1^\lambda); h \leftarrow \mathbb{G}; y \leftarrow \mathbb{Z}_{|\mathbb{G}|}; g \leftarrow h^y\ \right] = \nu(\lambda).$$

A.2 Collision-Resistant Hash Functions

Since [GGHK21] compresses the messages exchanged between the prover and the verifier, we verbatim report from [KL14] the definition of collision-resistant hash functions.

Definition 9. *A hash function* Hf *is a pair of PPT algorithms* $(\mathsf{Gen}; \mathsf{H})$ *fulfilling the following:*

- Gen *is a PPT algorithm which takes as input a security parameter* λ *and outputs a key* s*. We assume that* 1^λ *is included in* s*.*
- *There exists a polynomial* `poly` *such that* H *is (deterministic) polynomial-time algorithm that takes as input a key* s *and any string* $x \in \{0,1\}^*$*,and outputs a string* $\mathsf{H}^s(x) \in \{0,1\}^{\mathtt{poly}(\lambda)}$*.*

The security property of $\mathsf{Hf} = (\mathsf{Gen}, \mathsf{H})$ is defined using an experiment $\mathsf{ExpHashColl}$ for Hf, an adversary $\mathcal{A}$ and a security parameter λ.

$\mathsf{ExpHashColl}_{\mathsf{Hf},\mathcal{A}}(\lambda)$

1. $s \leftarrow \mathsf{Gen}(1^\lambda)$.
2. The adversary $\mathcal{A}$ on input s computes x and x'.
3. The output of the experiment is 1 if and only if $x \neq x'$ and $\mathsf{H}^s(x) = \mathsf{H}^s(x')$. In such a case we say that $\mathcal{A}$ has found a collision for Hf.

Definition 10. *A hash function* $\mathsf{Hf} = (\mathsf{Gen}, \mathsf{H})$ *is collision resistant if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\text{Prob}\left[\ \mathsf{ExpHashColl}_{\mathsf{Hf},\mathcal{A}}(\lambda) = 1\ \right] \leq \nu(\lambda).$$

B 1-out-of-2 Equivocal Commitment from DL [GGHK21]

We now report the instantiation of the 1-out-of-2 equivocal commitment scheme GGHK from [GGHK21]. GGHK uses the same CRS of the (non-interactive) Pedersen commitment scheme. Namely, the CRS contains two random group elements g_0, h in a group $\mathbb{G}$ where solving the discrete logarithm is believed to be hard (i.e., it can be seen as an SRS). The parameters space $\mathcal{Y}^{\mathsf{pp}}$ is $\{(p_0, p_1) | p_0, p_1 \in \mathbb{G}, p_1 = p_0^2 g_0^{-1}\}$. The polynomial-time relation $\mathcal{R}_T$ related to this instantiation is $\mathcal{R}_T = \{(x, w) : x = h^w\}$. The randomness space for $\mathsf{BindCom}$ and $\mathsf{EquivCom}$ is $\mathtt{D} = \{0,1\}^{2\lambda}$.

In the following, we describe the algorithms of GGHK. For conciseness, we omit the appropriate parsing of the CRS pp from the below algorithms.

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter, generate the appropriate group $\mathbb{G}$. Sample $g_0, h \leftarrow\!\!\$\ \mathbb{G}$. Output $\mathsf{pp} = (g_0, h)$.

- $\mathsf{Gen}(\mathsf{pp}, \beta; \mathtt{rand}_0)$: Parse pp as (p_0, h). Sample $y_{1-\beta} \leftarrow\!\!\$\ \mathbb{Z}_{|\mathbb{G}|}$ according to randomness $\mathtt{rand}_0$. Generate trapdoor parameter $p_{1-\beta} = h^{y_{1-\beta}}$. Generate binding parameter p_β by interpolating a degree one polynomial in the exponent base h, and evaluating it in point $\beta + 1$. In this interpolation, the exponent of g_0 is associated with point 0, while parameter p_i is associated with point $i+1$. In particular, if $\beta = 1$ compute $p_1 = p_0^2 g_0^{-1}$, otherwise compute $p_0 = p_1^{1/2} g_0^{1/2}$. Output (p_0, p_1, td), where $\mathsf{td} = y_{1-\beta}$.
- $\mathsf{EquivCom}(\mathsf{pp}, \beta, m, p_0, p_1, \mathsf{td}; \mathtt{rand}_1)$: Parse pp as (p_0, h) and $\mathtt{rand}_1$ as (r_0, r_1). Set $m_\beta = m$ and $m_{1-\beta} = 0$. Compute $\mathtt{com}_0 = h^{r_0} p_0^{m_0}$ and $\mathtt{com}_1 = h^{r_1} p_1^{m_1}$. Set $\mathtt{com} = (\mathtt{com}_0, \mathtt{com}_1)$ and $\mathsf{aux} = (r_0, r_1)$. Output $(\mathtt{com}, \mathsf{aux})$.
- $\mathsf{Equiv}(\mathsf{pp}, \beta, m_0, m_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$: Parse aux as (r_0, r_1). Compute $r'_{1-\beta} = r_{1-\beta} - \mathsf{td} \cdot m_{1-\beta}$, $r'_\beta = r_\beta$ and output r, where $r = (r'_0, r'_1)$.
- $\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, m'_0, m'_1; r)$: Parse pp as (p_0, h) and r as (r_0, r_1). Compute $\mathtt{com}_b = h^{r'_b} p_b^{m'_b}$ for $b \in \{0, 1\}$. Output $\mathtt{com} = (\mathtt{com}_0, \mathtt{com}_1)$.

Remark 1. GGHK is a 1-out-of-2 equivocal commitment. Indeed, GGHK satisfies the partial equivocation property since a valid witness for $\mathcal{R}_{\mathsf{DL}}$ on one of the parameters always allows successful equivocation in the corresponding position (i.e., as in the Pedersen commitment scheme). It has computational fixed equivocation. In [GGHK21], a weaker notion called partial binding involving only one commitment equivocated in both positions is proved. However, their proof directly involves the underlying Pedersen commitments that make up a 1-out-of-2 equivocal commitment. Therefore, one can carry essentially the same reduction for the case of two (distinct) 1-out-of-2 equivocal commitments w.r.t. the same parameters. GGHK is perfect position hiding. In [GGHK21], a stronger notion in which the commitment must hide both the committed values and the binding position is proved. In [GGHK21], the trapdoorness property is not formulated. Nevertheless, GGHK achieves perfect honest-receiver trapdoorness. All these properties are achieved in the SRS model.

C Details on the $(1, \ell)$-PTR of [GGHK21]

Here, we report a sketch of the security proofs using the commitment of Sec. 4. Completeness trivially follows from the completeness of the underlying protocols and the partial equivocation of the commitment scheme.

Computational special soundness (sketch). It is possible to create an extractor $\mathsf{Extract}$ for Π using the extractor $\mathsf{Extract}'$ of the underlying protocol Π'. $\mathsf{Extract}$ receives in input two accepting transcripts of Π $(a = (\mathtt{com}, p_0, p_1), c_0, z_0 = (z'_0, r_0, p_0, p_1))$ and $(a = (\mathtt{com}, p_0, p_1), c_1, z_1 = (z'_1, r_1, p_0, p_1))$ for the same statement x. $\mathsf{Extract}$ can derive the first-round messages of the underlying underlying protocol Π' by running its EHVZK simulator. Let such messages be $a_0^1 \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_0, c_0, z'_0)$, $a_1^1 \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_1, c_0, z'_0)$, $a_0^2 \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_0, c_1, z'_1)$, $a_1^2 \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_1, c_1, z'_1)$.

With non-negligible probability either $a_0^1 = a_0^2$ or $a_1^1 = a_1^2$, otherwise it is possible to break the computational fixed equivocation of CS. GGHK has computational fixed equivocation, therefore there is a negligible probability that a PPT P^* breaks the fixed equivocation property. Assume $a_0^1 \neq a_0^2$ and $a_1^1 \neq a_1^2$, the adversarial sender $\mathcal{A}$ breaks the fixed equivocation property of CS as follows:

- $\mathcal{A}$ sends $(p_0, p_1, r_0, r_1, r_0, r_1, a_0^1, a_0^2, a_1^1, a_1^2, a_0^1, a_0^2, a_1^1, a_1^2)$ to the challenger of fixed equivocation.

Since the two transcripts are accepting, it holds that $\mathsf{BindCom}(\mathsf{pp}, p_0, p_1, a_0^1, a_1^1; r_0) = \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, a_0^2, a_1^2; r_1) = \mathtt{com}$ such that $a = (\mathtt{com}, p_0, p_1)$. Thus $\mathcal{A}$ wins the fixed equivocation experiment with the same probability that $a_0^1 \neq a_0^2$ and $a_1^1 \neq a_1^2$. Since the probability that $\mathcal{A}$ wins the binding experiment is negligible, the following holds with non-negligible probability: either $a_0^1 = a_0^2$ or $a_1^1 = a_1^2$. It follows that $\mathsf{Extract}$ has, with non-negligible probability, at least one pair of accepting transcripts for Π' sharing the same first-round message to give in input to $\mathsf{Extract}'$.

$\mathsf{Extract}$ runs $\mathsf{Extract}'$ on such transcripts and outputs whatever $\mathsf{Extract}'$ outputs. Recall that $x = (x_0, x_1)$ and $\mathsf{Extract}'$ outputs a witness w such that $(x_i, w) \in \mathcal{R}_{\mathcal{L}}$ with $i \in \{0,1\}$, which is also a valid witness for relation $\mathcal{R}_{OR}$ (cfr., Fig. 2) If Π' is special sound, $\mathsf{Extract}'$ successfully extracts a witness with probability 1. Instead, if special soundness holds only computationally $\mathsf{Extract}'$, $\mathsf{Extract}$ will have a negligible probability of failure.

In [GGHK21] an hash function is used to compress the first message of Π. We consider this in the rest of the proof. Let $\mathsf{Hf} = (\mathsf{Gen}, \mathsf{H})$ be a collision resistant hash function (cfr., App. A.2), now the message a produced by P_0 is $a = \mathsf{H}^s((\mathtt{com}, p_0, p_1))$, where s is a random value defined in a common reference string CRS[22] and generated using algorithm Gen on the same security parameter used in Π. Let us assume that even if the adversarial prover P^* of Π cannot break the partial binding of CS and cannot break the special soundness of the underlying protocol Π', P^* can still break computational special soundness of Π. We can show that in this case there exists an adversary $\mathcal{A}$ with access to P^* that wins in $\mathsf{ExpHashColl}$ with non-negligible probability. $\mathcal{A}$ obtains the value s in the CRS and the message (a, c_0, c_1, z_0, z_1) from P^*. Looking at the transcripts (a, c_0, z_0) and (a, c_1, z_1), it cannot be that a is equal to $\mathsf{H}^s((\mathtt{com}, p_0, p_1))$ for the same $(\mathtt{com}, p_0, p_1)$, otherwise, as previously shown, it is possible to break either the special soundness of Π' or the partial binding of CS. Then there must exists two values $(\mathtt{com}, p_0, p_1)$ and $(\mathtt{com}', p_0', p_1')$ such that $\mathsf{H}^s((\mathtt{com}, p_0, p_1)) = \mathsf{H}^s(\mathtt{com}', p_0', p_1'))$. This means that $\mathcal{A}$ found a collision for Hf, thus reaching a contradiction. Since $\mathcal{A}$ can win in this experiment only with negligible probability, it follows that P^* can break computational special soundness with the same negligible probability.

EHVZK (sketch). Let $D_c^{(z)} = \left\{(z^*, r, p_0, p_1) | r \leftarrow\!\!\$\, \mathtt{D}; z^* \leftarrow\!\!\$\, D_c^{(z^*)}; (p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta = 1)\right\}$, where $\mathtt{D}$ is the randomness space of the underlying 1-out-of-ℓ equivocal commitment scheme[23] and pp the common random string previously generate in a trusted manner by running $\mathsf{Setup}(1^\lambda)$. It is straightforward to see that Π is EHVZK with recyclable third-round messages.

The simulator $\mathcal{S}^{\mathsf{EHVZK}}((x_0, x_1), c, (z^*, r, p_0, p_1))$ is defined as follows.

1. Compute $v_b \leftarrow \mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_b, c, z^*)$, for $b \in \{0,1\}$;
2. Compute $\mathtt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, v_0, v_1, c; r)$;
3. Return $(\mathtt{com}, p_0, p_1)$.

Consider the following hybrids:

$\mathcal{H}_0$**:** it is equivalent to the real game with the honest prover P.

$\mathcal{H}_1$**:** it is equivalent to $\mathcal{H}_0$ except that the protocol is not interactive and uses a challenge c randomly sampled by $\{0,1\}^\lambda$ as challenge from V. The distribution of those two hybrids is identical since c is a random value.

$\mathcal{H}_2$**:** it is equivalent to $\mathcal{H}_1$ except that the prover P uses c and z^* from $D_c^{(z^*)}$ as input for $\mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}$ to compute v_β in the first round. Additionally, the execution of P_1' in the third round is removed since the value z^* used is the one taken in input from $D_c^{(z)}$. If there exists a distinguisher $\mathcal{D}_{EHVZK}$ that distinguishes $\mathcal{H}_1$ from $\mathcal{H}_2$ with non-negligible probability, we can define a distinguisher $\mathcal{D}'_{EHVZK}$ that breaks the EHVZK property of Π'. $\mathcal{D}'_{EHVZK}$ sends (x_β, w) to P' in the experiment $\mathsf{ExpEHVZK}_{\mathsf{P}', \mathcal{D}'_{EHVZK}}(c)$. P' returns $(v_\beta, c, \tilde{z})$.

$\mathcal{D}'_{EHVZK}$ computes a transcript as follows:

- $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$;
- $(\mathtt{com}, \mathsf{aux}) \leftarrow \mathsf{EquivCom}(\mathsf{pp}, \beta, v_\beta, p_0, p_1, \mathsf{td})$;

[22]We recall that the CRS does not need to be generated by a trusted entity since it is composed only of the description of a CRHF (e.g., SHA256 in practice).

[23]For GGHK $\mathtt{D} = \{0,1\}^{2\lambda}$.

- $v_{1-\beta} \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_{1-\beta}, c, \tilde{z})$;
- $r \leftarrow \mathsf{Equiv}(\mathsf{pp}, \beta, v_0, v_1, p_0, p_1, \mathsf{td}, \mathsf{aux})$.

$\mathcal{D}'_{EHVZK}$ sends $(a = (\mathtt{com}, p_0, p_1), c, z = (\tilde{z}, r, p_0, p_1))$ to $\mathcal{D}_{EHVZK}$ and returns the value returned by $\mathcal{D}_{EHVZK}$.

Since Π' is perfect EHVZK, $\mathcal{H}_1$ and $\mathcal{H}_2$ are identically distributed.

$\mathcal{H}_3$: it is the same as $\mathcal{H}_2$ excepts that P computes $\mathtt{com} \leftarrow \mathsf{BindCom}(\mathsf{pp}, p_0, p_1, v_0, v_1; r)$, where r is not generated by first calling $\mathsf{EquivCom}(\mathsf{pp}, \beta, v_\beta, p_0, p_1, \mathsf{td})$ and then $\mathsf{Equiv}(\mathsf{pp}, \beta, v_0, v_1, p_0, p_1, \mathsf{td}, \mathsf{aux}; \mathtt{rand}_1)$ but it is taken from the input of P. If there exists a distinguisher $\mathcal{D}_{EHVZK}$ that distinguishes between $\mathcal{H}_2$ and $\mathcal{H}_3$ with non-negligible probability, we can define an adversary $\mathcal{A}_{trap}$ that breaks the honest receiver trapdoorness property of the underlying 1-out-of-2 equivocal commitment scheme.

- Given $(x = (x_0, x_1), w)$, $\mathcal{A}_{trap}$ computes $v_b \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_b, c, z^*)$, for each $b \in \{0,1\}$, and $(p_0, p_1, \mathsf{td}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \beta)$ where β is from the input of P;
- $\mathcal{A}_{trap}$ sends $(v_0, v_1, p_0, p_1, \mathsf{td}, \beta)$ to $\mathsf{ExpTrap}(\lambda)$, and receives $(\mathtt{com}, r)$ from $\mathsf{ExpTrap}$.
- $\mathcal{A}_{trap}$ sends $(a = (\mathtt{com}, p_0, p_1), c, (z^*, r, p_0, p_1))$ to $\mathcal{D}_{EHVZK}$ and receives back a bit b. $\mathcal{A}_{trap}$ outputs b.

Note that $\mathcal{A}_{trap}$ perfectly simulates $\mathcal{H}_2$ when $\mathsf{ExpTrap}$ uses $\mathsf{EquivCom}$ to generate $\mathtt{com}$ and then Equiv to generate r, and perfectly simulates $\mathcal{H}_3$ when $\mathsf{ExpTrap}$ uses $\mathsf{BindCom}$ together with a uniformly random r to generate $\mathtt{com}$.

$\mathcal{H}_4$: it is the same as $\mathcal{H}_3$ except that the prover of $\mathcal{H}_4$ fixes the binding position to $\beta' = 1$ when calling the parameter generation algorithm Gen. If $\beta' = \beta$, the two hybrids are identically distributed, then $\mathcal{D}_{EHVZK}$ cannot distinguish $\mathcal{H}_3$ from $\mathcal{H}_4$. Let us consider the case in which $\beta' \neq \beta$. If there exists a distinguisher $\mathcal{D}_{EHVZK}$ that distinguishes between $\mathcal{H}_3$ and $\mathcal{H}_4$ with non-negligible probability, we can define an adversary $\mathcal{A}_{hid}$ that breaks the position hiding property of the 1-out-of-2 equivocal commitment scheme with non-negligible probability.

- Given $(x = (x_0, x_1), w)$, $\mathcal{A}_{hid}$ computes $v_b \leftarrow \mathcal{S}^{\mathsf{EHVZK}}_{\mathsf{base}}(x_b, c, z^*)$, for each $b \in \{0,1\}$.
- $\mathcal{A}_{hid}$ receives (p_0, p_1, td) from ExpHid.
- $\mathcal{A}_{hid}$ sets $(a = (\mathtt{com}, p_0, p_1), c, z = (z^*, r, p_0, p_1))$, where c and z^* are the value in input to P, and $\mathtt{com}$ is generated by running $\mathsf{BindCom}$ with the messages (v_0, v_1) and the randomness r taken from the input of P. $\mathcal{A}_{hid}$ sends (a, c, z) to $\mathcal{D}_{EHVZK}$ and receives a bit b. $\mathcal{A}_{hid}$ outputs b.

Note that $\mathcal{A}_{hid}$ perfectly simulates $\mathcal{H}_3$ given $\beta = 0$ when ExpHid calculates (p_0, p_1, td) from $\beta = 0$, and perfectly simulates $\mathcal{H}_4$ when ExpHid calculates (p_0, p_1, td) from $\beta = 1$.

$\mathcal{H}_5$: it is equal to $\mathcal{H}_4$ except that the prover P takes additional inputs $(z^*, r, p_0, p_1))$ taken from $D_c^{(z)}$, and uses such inputs instead of sampling them by itself. $\mathcal{H}_4$ and $\mathcal{H}_5$ are indistinguishable since the values passed in input in $\mathcal{H}_5$ are identically distributed to the values computed in $\mathcal{H}_4$.

Note that $\mathcal{H}_5$ is identically distributed to the simulator $\mathcal{S}^{\mathsf{EHVZK}}((x_0, x_1), c, (z^*, r, p_0, p_1))$. This concludes the proof.

Achieved properties. As we said, it is possible to instantiate the compiler in Fig. 2 with GGHK described in App. B. Since the commitment scheme in App. B achieves computational fixed equivocation, perfect position hiding and perfect honest-receiver trapdoorness in the SRS model, the resulting protocol Π is computational special sound and perfect EHVZK in the SRS model.

Remark 2. We notice that in [GGHK21] they refer to the standard formulation of Σ-protocols requiring special soundness to hold for an unbounded prover. Nevertheless, they then proceed by proposing an extractor that fails only with negligible probability assuming the commitment scheme is computational partial binding (i.e., a weaker version of the fixed equivocation property stating that it is infeasible to generate a commitment and accepting openings to $(m_0^1, m_1^1), (m_0^2, m_1^2)$ such that $m_0^1 \neq m_0^2$ and $m_1^1 \neq m_1^2$.). It is straightforward to notice that their construction cannot achieve statistical/perfect special soundness in the classical sense since it relies on a security property that only holds computationally. We took care of this subtlety by formally defining computational special soundness in Sec. 3.2 and using such definition in the reduction presented above.

Remark 3. We also notice an issue in the proof for EHVZK proposed in [GGHK21]. They prove EHVZK using the following hybrids:

- $\mathcal{H}^\beta$ that is equivalent to the real protocol, except that in the first round a_β is generated using $\mathcal{S}_{\mathsf{base}}^{\mathsf{EHVZK}}(x_\beta, c, z)$;
- $\mathcal{H}^{\beta,p_0,p_1}$ that is equivalent to $\mathcal{H}^\beta$, except that the commitment is computed setting $\beta = 1$.

In [GGHK21], they argue that $\mathcal{H}^\beta$ and $\mathcal{H}^{\beta,p_0,p_1}$ are perfectly indistinguishable for the perfect hiding of their commitment, but this property is not enough to prove the indistinguishability of the two hybrids. Indeed, a 1-out-of-2 equivocal commitment scheme fulfilling the definitions presented in [GGHK21] does not necessarily guarantee the trapdoorness property. Therefore, the verifier may be able to discover the value of β at the end of the Reveal Phase.

It is worth noting that it is possible to define a commitment scheme achieving all the properties required in [GGHK21], but that reveals the binding position in the Reveal Phase. To see this, consider a commitment scheme in which the Sender, in the Reveal Phase, sends the message M to the Receiver, where $M = M^*||\beta$, with $||$ the concatenation operator, M^* the message to be sent by the Sender to the Receiver to compute the opening, and β the binding position used by the sender in the Commitment Phase. For example, considering the Reveal Phase of the commitment scheme in App. B, we can modify the last message sent by the sender to the receiver as described before, concatenating the index of the binding position to it. It is straightforward to see that the resulting scheme achieves computational binding, perfect hiding, but does not achieve trapdoorness.