Robust Non-Interactive Zero-Knowledge Combiners

Michele Ciampi[1], Lorenzo Magliocco[2], Daniele Venturi[2], and Yu Xia[1]

[1]The University of Edinburgh, UK
{michele.ciampi, yu.xia}@ed.ac.uk
[2]Sapienza University of Rome, Italy
{magliocco, venturi}@di.uniroma1.it

February 15, 2025

Abstract

A t-out-of-n robust non-interactive zero-knowledge (NIZK) combiner is a construction that, given access to n candidate instantiations of a NIZK for some language, itself implements a NIZK for the same language. Moreover, the combiner is secure, assuming at least t of the given candidates are secure. In this work, we provide the first definition of combiners for NIZK, and prove that no robust NIZK combiner exists assuming $t \leq \lfloor n/2 \rfloor$ (unless the polynomial hierarchy collapses).

On the positive side, we provide different constructions of robust NIZK combiners for $t > \lfloor n/2 \rfloor$. In particular, we show how to obtain:

- A black-box combiner working for a special class of *homomorphic* languages where n, t are polynomial and $t > \lfloor n/2 \rfloor$.
- A non-black-box combiner working for any language, where n, t are constant and $t > \lfloor n/2 \rfloor$.
- A non-black-box combiner working for any language, where n, t are polynomial and $t > \lfloor 2n/3 \rfloor$.

1 Introduction

A short story. Once upon a time, there was a little girl called Cryptess. She was smart and had a passion for crypto puzzles. She loved walking through her favourite forest, especially while thinking about the next hard problem to attack. One day, while wandering through the forest without thinking about where to go, she found herself in front of a beautiful glade. She had the impression[1] she had been already there before, but could not remember, so she entered the glade.

Once inside, she found a fairy flapping her wings and staring at her. The fairy said: "Welcome back, little Cryptess. I am the fairy Cryptophia and since you have found my magical glade, I grant you one wish." Cryptess thought about it for a second, then she asked: "Do you know about zero-knowledge proofs?". "Of course I do!", replied the fairy. Then, Cryptess continued: "Here is my wish then. I give you two non-interactive zero-knowledge proof systems, out of which at least one is secure but you don't know which one. I wish you design a non-interactive zero-knowledge proof system, based on the two I gave you, that is always secure.". Cryptophia immediately replied: "I am sorry. Unfortunately, what you ask for is impossible."

[1]Indeed, she was right [Mit13], but for some reason she had forgotten about it.

Cryptess was surprised at first, but after some moment of thought something kicked in her head. Then, the glade disappeared and the little girl found herself on the ground, in the middle of the forest. "Maybe, if I assume three proof systems instead of two..." Then, she smiled, and ran home. Now, she knew what to do.

1.1 Robust NIZK combiners

A combiner C for a primitive $\mathcal{P}$ is a cryptographic construction that, given access to two or more candidate instantiations of $\mathcal{P}$, itself implements the primitive $\mathcal{P}$. In particular, C is called a t-out-of-n *robust* $\mathcal{P}$-combiner if C is *secure* assuming at least $t < n$ of the candidates $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are *secure*. Of course, depending on the actual primitive $\mathcal{P}$, security can be formalized in different ways, and, indeed, robust combiners have been studied in the literature for a plethora of cryptographic primitives including hash functions, oblivious transfer, functional encryption schemes, and more. We review some of these results in Section 1.4.

In this paper, we study robust non-interactive zero-knowledge (NIZK) combiners in the common reference string (CRS) model [BFM88]. This allows us to provide a formal framework to combine NIZK protocols with different potential weaknesses. As a concrete example, one could combine NIZKs relying on assumptions of different flavours, including heuristic assumptions widely used for real-world deployments and assumptions that may be less resilient to the test of time (*e.g.*, ones threatened by quantum computers). As another example, it could be meaningful to combine implementations developed from different vendors or open-source projects, as they might not match the respective theoretical specification. For instance, implementations may be tampered with by its maintainers (*e.g.*, with backdoors), or setup parameters may have been generated maliciously (*e.g.*, by revealing trapdoors to some adversarial entity).

A review of NIZK proof systems. A NIZK proof system for a language $\mathcal{L} \subseteq \{0,1\}^*$ allows a prover to convince a verifier (both modelled as PPT algorithms) about the veracity of a shared statement $x \in \mathcal{L}$ using a single message π (called the proof). The prover is additionally given as input a witness w to $x \in \mathcal{L}$, and both the prover and the verifier are also given a CRS—denoted σ—that is initialized at setup. A NIZK proof system is said to be *complete* if for all $x \in \mathcal{L}$, and for all honestly sampled CRSs, the honest prover always convinces the honest verifier about the veracity of the statement. As for security, NIZK proof systems are required to satisfy different[2] properties. We recall the most important ones below.

Soundness: No computationally[3] unbounded malicious prover (given the CRS σ) can produce an accepting proof π^* for a *false* statement $x \in \{0,1\}^*$ that is accepted by the verifier.

Proof of knowledge: For some languages, false statements simply do not exist, thus making soundness trivial to achieve. In such cases, a stronger form of soundness (called *proof of knowledge*) is required, which roughly says that any successful prover must in fact *know* a witness. The latter is formalized by the existence of a PPT extractor $\mathcal{E}(\xi, x, \pi)$ that for every (possibly adversarial) accepting proof π outputs a valid witness $\tilde{w}$ to $x \in \mathcal{L}$ with high probability; the extractor is facilitated by a trapdoor ξ that is generated along with the CRS σ.

Zero knowledge: For every true statement $x \in \mathcal{L}$, honestly computed proofs π reveal nothing about the witness. The latter is formalized by the existence of a PPT simulator

[2] In fact, one can consider further security properties (such as witness indistinguishability or simulation soundness), but in this paper we only focus on the properties above.

[3] When soundness holds computationally, we speak of *arguments* instead of proofs.

$\mathcal{S}(\tau, x)$ that outputs a proof $\tilde{\pi}$ that is computationally indistinguishable from an honestly computed proof π; the simulator is facilitated by a trapdoor τ that is generated along with the CRS σ. This corresponds to so-called black-box zero knowledge; in the non-black-box case, for every adversary there exists an adversary-dependent simulator.

Some care must be taken when defining robustness of combiners for any primitive $\mathcal{P}$: In fact, one cannot just say that a combiner C is robust if C is secure assuming at least t of the n candidates $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are secure. This is because the combiner C could always ignore all of the n candidates and construct the primitive $\mathcal{P}$ from scratch. For this reason, when defining robustness, one must always make the underlying reduction explicit, by asking that, for every adversary $\mathcal{A}$ breaking security of C, there exists a reduction Red (with running time at most a polynomial factor slower than the running time of $\mathcal{A}$) that breaks security of at least $n - t + 1$ of the given candidates. This way, we are ensured that C is secure assuming at least t of the candidates are secure, and moreover C cannot construct $\mathcal{P}$ from scratch (as ensured by the existence of the above reduction).

1.2 Our contributions

We give the first definitions, impossibilities, and constructions for t-out-of-n robust NIZK combiners. Our definition of robustness considers different levels of black-boxness along three dimensions: (i) whether the combiner C accesses $\mathcal{P}_1, \ldots, \mathcal{P}_n$ as a black-box; (ii) whether the knowledge extractor (in the case of knowledge soundness) accesses the adversary as a black-box; (iii) whether the zero-knowledge simulator accesses the adversary as a black-box. See Section 3 for the formal definitions. For simplicity, in the rest of this introduction, we only focus on dimension (i) and always assume black-box extraction and simulation (however, our results also cover non-black-box extraction and simulation).

Our notion of robust NIZK combiners does not require that t of the n candidate proof systems satisfy both (knowledge) soundness and zero-knowledge. Rather, it suffices that t candidates are (knowledge) sound and t candidates are zero-knowledge, but these two subsets may not be identical. Hence, ours are *multi-property combiners* as defined by Fischlin, Lehman and Pietrzak [FL08, FLP08, FLP14]. On the other hand, all of our constructions require that all of the candidates have a negligible completeness error; this assumption is common in the literature on combiners.[4] We also argue that any robust NIZK combiner satisfying our definition, when instantiated with at least t good NIZK candidates, yields a NIZK satisfying the standard definitions of (knowledge) soundness and zero-knowledge, and thus it can be employed directly in any application requiring these properties.

After having formally defined the notion of robust NIZK combiners, in Section 4, we prove two negative results. First, we show that there is no t-out-of-n (even non-black-box) robust NIZK combiner for $t \leq \lfloor n/2 \rfloor$ (unless the polynomial hierarchy collapses). Second, we prove that t-out-of-n black-box robust NIZK combiners for $t \leq \lfloor n/2 \rfloor$ only exist for trivial languages, even assuming that t of the underlying NIZK candidates are *both* sound and zero-knowledge (*i.e.*, the second impossibility result even holds for combiners that are not multi-property). To complement the above negative results, we provide constructions of robust NIZK combiners. In particular, we obtain three different (and incomparable) combiners (see Table 1):

- Our first construction (cf. Section 5) yields a t-out-of-n black-box robust NIZK combiner assuming $t > \lfloor n/2 \rfloor$, where $n, t = \mathsf{poly}(\lambda)$. This construction only works for NIZK proofs and arguments (of knowledge) supporting so-called *homomorphic* languages [Cra96], as

[4] But note that in the setting of NIZK proof systems, one can always amplify non-negligible completeness to overwhelming completeness (at the price of a larger running time for the prover).

Combiner	Threshold	Type	Input/Output Language	Proof Size
§5	$t > \lfloor n/2 \rfloor$	Black-box	Homomorphic	$n \cdot \lvert\pi\rvert$
§6	$t > \lfloor n/2 \rfloor$, $n, t = O(1)$	Non-black-box	NP	$\mathsf{poly}(\lvert\pi\rvert)$
§7	$t > \lfloor 2n/3 \rfloor$	Non-black-box	NP	$n \cdot \lvert\pi\rvert + \mathsf{poly}(\lvert x\rvert, \lvert w\rvert, \lambda)$

Table 1: Comparing our results for robust NIZK combiners in terms of threshold parameters, type of black-boxness, supported language, and proof size. In particular by *Input/Output Language*, we refer to both the language that must be supported by the input candidates and the language supported by the NIZK returned from the combiner. From the table, it can be seen that all our combiners are *language preserving*. We assume that all the input protocols issue proofs of size at most $|\pi|$ (where $|\pi|$ is at least polynomial in the security parameter λ). We note that the first and the second construction preserve the succinctness of the input candidates.

defined by Maurer [Mau09], which essentially means that the witness w belongs to a group $(\mathsf{G}, \star)$ and the statement x belongs to a group $(\mathsf{H}, \otimes)$, and moreover there is a function $f : \mathsf{G} \to \mathsf{H}$ such that $f(x) \otimes f(y) = f(x \star y)$ and f is one-way function (*i.e.*, it is infeasible to compute x from $f(x)$ for a randomly chosen x). Concrete examples of such languages include the language for proving the knowledge of a discrete logarithm, that a tuple is Diffie-Hellman, the equality of embedded values, and more.

- Our second construction (cf. Section 6) yields a t-out-of-n non-black-box robust NIZK combiner assuming $t > \lfloor n/2 \rfloor$, where $n, t = O(1)$. This construction works for arbitrary languages, and supports both NIZK arguments and proofs (of knowledge).
- Our third construction (cf. Section 7) yields a t-out-of-n non-black-box robust NIZK combiner assuming $t > \lfloor 2n/3 \rfloor$, where $n, t = \mathsf{poly}(\lambda)$. This construction also works for arbitrary languages, but only supports proofs (of knowledge).

Due to our impossibility results, our first and second constructions are essentially tight in terms of achievable parameters. We leave it as an interesting open problem to construct a robust NIZK combiner supporting all of NP for the parameters regime $t > \lfloor n/2 \rfloor$ with $n, t = \mathsf{poly}(\lambda)$.

1.3 Technical overview

Below, we give an overview of the main technical ideas behind our results.

1.3.1 Impossibility results

We start by explaining how to prove that no robust NIZK combiner exists without assuming the majority of the candidates being *good*. Let $n = 2$ and $t = 1$, and assume that there exists a 1-out-of-2 robust NIZK combiner C. Consider the following dummy NIZK candidates: (i) the prover in $\mathcal{P}_1$, upon input (x, w), always outputs 0 and the verifier only accepts 0 as a proof; (ii) the prover in $\mathcal{P}_2$, upon input (x, w), always outputs w and the verifier accepts if and only if w is a valid witness for x. Clearly, $\mathcal{P}_1$ is unconditionally zero-knowledge but not sound, whereas $\mathcal{P}_2$ is unconditionally sound but not zero-knowledge.

Yet, the definition of robust combiner implies that C must work for $\mathcal{P}_1$ and $\mathcal{P}_2$. In particular, given any unbounded adversary that breaks the zero-knowledge (resp. soundness) property of C, there exists a reduction, with at most a polynomial slowdown, that breaks the zero-knowledge (resp. soundness) property of both $\mathcal{P}_1$ and $\mathcal{P}_2$; but this is impossible, as $\mathcal{P}_1$ is unconditionally zero-knowledge (resp. $\mathcal{P}_2$ is unconditionally sound). It follows that C is both unconditionally sound and zero-knowledge, which is also impossible (unless the polynomial hierarchy

collapses [Ps05]). We conclude that C cannot exist. Note that, for the above argument, it is completely irrelevant whether the combiner is black-box or not. For more detail, we refer to Section 4.

Impossibility for robust combiners that are not multi-property. We note that the proof we have just sketched crucially relies on the fact that C is a multi-property combiner. This, informally, requires that the combiner works, even if one instance is zero-knowledge (but not sound) and the other instance is sound (but not zero-knowledge). It is natural to ask whether the impossibility holds when the combiner receives as input one scheme that is both zero-knowledge and sound (while the other scheme having no security at all). We prove that a combiner that makes black-box use of the input primitives can be secure in the setting where $t \leq \lfloor n/2 \rfloor$ only when proving instances of trivial languages. At a high level, (for the case $n = 2$, $t = 1$, and assuming that C has oracle access to the primitives) the proof works as follows. Let C be a secure combiner that works in the setting we are considering now. Clearly, such a combiner should work even when executed with two candidate schemes that are both zero-knowledge and sound. We denote these schemes with Π_1 and Π_2. Note that because C has only oracle access to the input instances, then it must be that $\mathsf{C}^{\mathcal{O}_1,\Pi_2}$ is still zero-knowledge and sound for any oracle $\mathcal{O}_1$ that acts as the insecure instance, fully controlled by the adversary.[5] This, in particular must hold if the oracle acts exactly like the zero-knowledge simulator of Π_1, which we denote with $\mathcal{S}_1$.

By assumption, $\mathsf{C}^{\mathcal{O}_1,\Pi_2}$ is zero-knowledge, then there exists $\mathcal{S}$, such that for any adversary $\{\mathcal{S}^{\mathcal{O}_1}\} \approx \{\mathsf{C}^{\mathcal{O}_1,\Pi_2}\}$. We consider now $\mathsf{C}^{\Pi_1,\mathcal{O}_2}$. For the same argument as before, such a combiner must be secure with respect to any oracle $\mathcal{O}_2$. In particular, consider the case where $\mathcal{O}_2$ behaves exactly like the zero-knowledge simulator $\mathcal{S}_2$ of Π_2. We now design an adversary $\mathcal{P}^\star$ who breaks the soundness of $\mathsf{C}^{\Pi_1,\mathcal{O}_2}$ as follows.

$\mathcal{P}^\star$ runs $\mathcal{S}^{\mathcal{O}_1}$ (the zero-knowledge simulator for the scheme $\mathsf{C}^{\mathcal{O}_1,\Pi_2}$ described above) on input a false instance for a membership hard language, and emulates $\mathcal{O}_1$ as follows. Whenever $\mathcal{S}$ makes a query to $\mathcal{O}_1$, $\mathcal{P}^\star$ queries Π_1 and returns the answer to $\mathcal{S}$. Whenever $\mathcal{S}$ wants to output something with respect to any of the algorithms of Π_2, $\mathcal{P}^\star$ mirrors this behavior via $\mathcal{O}_2$ (that we recall is fully controlled by $\mathcal{P}^\star$). When $\mathcal{S}$ returns a proof π, $\mathcal{P}^\star$ returns π and stops. We observe that π must be an accepting proof, as the output distribution of $\mathcal{P}^\star$ is computationally indistinguishable from that of $\mathcal{S}^{\mathcal{O}_1}$.

We finally note that in the above proof, we have assumed that C has oracle access to the schemes, but we can get rid of this assumption and make the arguments work for the case where C has only black-box access to the primitives. This can be done by assuming pseudo-random functions (PRFs) and by using the output of a PRF to obtain the randomness for the input schemes, instead of using randomness provided as input to the oracle from the caller. For a more formal proof, we refer the reader to Section 4.

1.3.2 Combiner for homomorphic languages

We start with the combiner for homomorphic languages. For simplicity, we describe the combiner for proving knowledge of the discrete logarithm $w \in \mathbb{Z}_q$ of a group element $x = g^w \in \mathsf{G}$ (where g is the generator of the cyclic group G with order q). In the formal part of the paper, we give a more general construction that works for a large class of homomorphic languages,

[5] The fact that one of the candidate is not secure, means that, potentially, it could be fully controlled by the adversary (i.e., the adversary can see all the inputs given to the protocols, and decide the output of such protocols).

as defined by Maurer [Mau09]. Concrete examples of such languages include Diffie-Hellman, Guillou-Quisquater, proofs for equality of embedded values, and more.

Assume we have n NIZK argument systems $\mathcal{P}_1, \ldots, \mathcal{P}_n$, each for proving knowledge of the discrete logarithm of a group element. We aim at constructing a combiner that returns a NIZK argument system $\mathcal{P}$ for the language of discrete logarithms and which guarantees security (*i.e.*, zero-knowledge and knowledge soundness) assuming that t out of the n input schemes are secure.[6] The prover takes a random polynomial p of degree $t-1$ such that $p(0) = w$, and computes the shares $(i, p(i))$ for each $i \in [n]$, thus obtaining a t-out-of-n Shamir's secret sharing of the witness w. Then, the prover derives n *sub-statements* $x_i \leftarrow g^{p(i)}$ and runs the prover algorithm of the scheme $\mathcal{P}_i$ on input the sub-statement x_i and its witness $p(i)$ for each $i \in [n]$, obtaining a proof π_i. The final proof π consists of all the generated proofs $\{\pi_i\}_{i\in[n]}$ and all the sub-statements $\{x_i\}_{i\in[n]}$. The verifier first checks that all the proofs are accepting (with respect to each individual scheme), and then runs the reconstruction of Shamir's secret sharing "in the exponent" in order to check that $g^{p(0)}$ equals x and that $g^{p(i)}$ equals x_i for all $i \in [n]$. The latter is possible because Shamir's reconstruction is based on Lagrange interpolation, which basically just requires the verifier to sum t polynomial evaluations (multiplied by some constant), and the latter can be done "in the exponent" by exploiting the homomorphic properties of the function $f(\alpha) = g^\alpha$ between the groups G and H (with $\mathsf{H} = \mathsf{G}$ in this case).

Zero-knowledge is guaranteed because the witnesses w_i of at least t of the sub-statements x_i are protected (thanks to the assumption that t candidates are zero-knowledge and that $t > \lfloor n/2 \rfloor$), hence the privacy property of Shamir's secret sharing implies that the witness w is protected. Knowledge soundness is guaranteed by the fact that, for at least t of the sub-statements x_i, the proof will be generated using a knowledge sound candidate, hence, it will be possible to extract t shares (polynomial evaluations), which we can use to reconstruct a candidate witness w'. It remains to argue that w' corresponds to w. But this must be the case, given that $g^{p(0)} = g^{w'}$.

1.3.3 Recursive proof combiner

The combiner we have just described works for a limited class of languages. Next, we propose a NIZK combiner that supports all of NP, assuming the underlying candidates also support all of NP. To simplify the description of our scheme, let us consider here the case where $n = 3$ and $t = 2$. The prover performs a recursive proof, using all possible combinations of the given candidates. In other words, the prover computes a proof that $x \in \mathcal{L}$ using the i-th candidate, thus obtaining a proof π_i; then, using the scheme j (with $j > i$) it proves the following: "I know a proof π_i for the statement $x \in \mathcal{L}$ which the verifier of the scheme $\mathcal{P}_i$ would accept". We denote this latter proof with $\pi_{i,j}$. The prover proceeds as above for each $i, j \in \{1, 2, 3\}$ with $j > i$, and finally it sends to the verifier $\pi_{1,2}, \pi_{1,3}, \pi_{2,3}$. The verifier accepts if all three proofs are accepting.

The above scheme is zero-knowledge, because every proof $\pi_{i,j}$ hides the witness: either the i-th candidate is zero-knowledge (and hence w is protected), or the j-th candidate is zero-knowledge (and hence the proof π_i, that could have leaked something about the witness, remains hidden to the adversary). Note that this holds given that $t > \lfloor n/2 \rfloor$. Knowledge soundness holds, because there exists at least one recursive proof $\pi_{i,j}$, generated using the i-th and the j-th candidates which are both knowledge sound (this again follows from the fact that $t > \lfloor n/2 \rfloor$). The construction can be extended to the case where n and t are arbitrary, but to keep the

[6] Here, for simplicity, when we say that a scheme is secure, we mean that it enjoys both zero-knowledge and (knowledge) soundness. However, we remark that our construction also works in case the subsets of candidates that are either zero-knowledge or (knowledge) sound are not identical.

running time of the prover polynomial, we must require that n and t are constant.

1.3.4 MPC-in-the-head combiner

With our third construction we want to remove the limitation on n and t, while still supporting all of NP. The construction is inspired by the MPC-in-the-head approach proposed in [IKOS07], and a similar approach has been used in [GJS19] to construct amplifiers for zero-knowledge from sub-exponentially secure public-key encryption.

Our combiner is based on an n-party information-theoretic multi-party computation (MPC) protocol for the following function g: *given as input n shares of a t-out-of-n secret sharing scheme, perform the reconstruction thus obtaining a value w, and return 1 if and only if w is a valid witness for x.* The MPC protocol guarantees security as long as k parties are corrupted (with $k < n/2$).

To simplify the description of the combiner, let us first assume that the combiner has access to a non-interactive commitment scheme; we will explain later how to remove this assumption. Let $\mathsf{pt}_1, \ldots, \mathsf{pt}_n$ be the n parties of the MPC protocol. The prover of our combiner computes a t-out-of-n secret sharing of the witness w, thus obtaining the shares $w_1, \ldots, w_n$, and executes the MPC protocol for the above function g in its head simulating the role of each party pt_i with input w_i. This process generates, for each party pt_i, a set of ingoing and outgoing messages that we denote with $\mathsf{in\text{-}out}_i$ for $i \in [n]$. The prover then, for each $i \in [n]$, commits to these messages and uses the candidate $\mathcal{P}_i$ to prove that the messages in $\mathsf{in\text{-}out}_i$ correspond to the view generated by running party pt_i of the above MPC protocol. The prover finally sends all the generated proofs and all the commitments to the values $\mathsf{in\text{-}out}_i$.

The verifier first checks that there is a match between the commitments corresponding to the ingoing messages of a party pt_i and the outgoing messages of all the other parties, and accepts the proof if all the proofs are valid. Intuitively, the zero-knowledge property is guaranteed given that t shares are protected by the zero-knowledge property of the t secure candidates, and moreover, the security of the MPC protocol guarantees that, even if the messages of k parties are controlled by the adversary (hence, the adversary knows the input of these parties), then the shares of the *honest parties* are still protected thanks to the hiding property of the commitment. Hence, due to the privacy of the secret sharing scheme, the witness itself is protected.

To argue (knowledge) soundness, ideally we would like to rely on the correctness of the MPC protocol, similarly to what is done in [IKOS07]. Unfortunately, we cannot do that, given that a dishonest prover can compute some of the messages in a malicious way, which rules out relying on the correctness or on the security of the MPC protocol. To solve this problem we require the MPC protocol to be *robust*. This property was used also in [IKOS07] to prove some of their results. The robustness property requires that even if the randomness used by the honest parties is chosen adversarially, and $n - t$ parties are fully controlled by the adversary, then the output of the honest parties is always correct. One possible instantiation of such an MPC protocol is [BGW88], which tolerates a corruption threshold of $k < \lfloor \frac{n-1}{3} \rfloor$, and, as claimed in [GMO16], is also robust.

It remains to explain how to remove the assumption that the combiner is given access to a secure non-interactive commitment scheme. Basically, we do that by having the combiner construct a non-interactive commitment scheme using the given NIZK candidates themselves. More precisely, we rely on the fact [HN24, LMP24] that one-way functions exist if: (i) every language in NP has a zero-knowledge proof argument (*i.e.*, $\mathsf{NP} \subseteq \mathsf{ZKA}$), and (ii) ZKA contains non-trivial languages (*i.e.*, $\mathsf{ZKA} \not\subseteq \mathsf{ioP/poly}$). Hence, our combiner can first obtain a one-way function candidate f_i using the NIZK candidate $\mathcal{P}_i$, combine[7] the one-way function candidates

[7] Robust combiners for one-way functions are well-known [Her05], for instance the classical concatenation

$\{f_i\}_{i\in[n]}$ into a secure one-way function construction f, and finally use f to obtain[8] a two-round commitment using Naor's construction [Nao91], that we can make non-interactive by plugging the first round of the commitment in the CRS. This makes our combiner unconditional under assumptions (i) and (ii); however, note that if assumption (i) does not hold it is impossible to obtain robust NIZK combiners for all of NP. Moreover, if (ii) does not hold, we have that $\mathsf{NP} \subseteq \mathsf{ioP/poly}$ (as $\mathsf{NP} \subseteq \mathsf{ZKA}$ must hold, and $\mathsf{ZKA} \subseteq \mathsf{ioP/poly}$). Also note that since Naor's commitment yields a non-interactive commitment (in the CRS model) with statistical binding (and computational hiding), the final combiner only works for proofs (and not for arguments); additionally, since the prover runs the NIZK candidates to prove a statement involving the commitments (which, in turn, are obtained using the NIZK candidates themselves), the combiner is non-black-box in the use of the candidates.

1.4 Related works

Combiners. Several previous works have considered and applied the concept of combiners, either implicitly or explicitly. The first formal treatment of robust combiners is due to Harnik *et al.* [HKN+05], in the setting of oblivious transfer and key agreement protocols. In particular, this was the first work pointing out the necessity of making the reduction explicit in the definition, in order to rule out trivial solutions that ignore the underlying candidates.

Most relevant to our work is Sommer's Master Thesis [Som06] that studies robust combiners for interactive proof systems. This work rules out very specific forms of robust combiners (*e.g.*, those that are deterministic or that work by splitting the statement into multiple parts).

A long line of research studies robust combiners for hash functions [FL08, FLP08, Mit13, MP14]. In this context, it is particularly challenging to obtain *short* combiners, in which the output length of the combiner is significantly shorter than the concatenation of the candidates' outputs. While there are lower bounds for short robust hash functions combiners [BB06, CRS+07, Pie07, Pie08, Mit12], recent work has shown that such lower bounds can be overcome in the random oracle model [DFG+23]. Robust combiners have also been considered for other cryptographic primitives, including public-key encryption [HKN+05, DK05], oblivious transfer [HKN+05, Som06, MPW07, CDFR17, FR24], commitment schemes [Som06], private information retrieval [MP06], key encapsulation mechanisms [GHP18], authenticated encryption with associated data [PR20], obfuscators [AJS17, FHNS16], and functional encryption [AJN+16, ABJ+19, JMS20].

Amplifiers. A recent line of work studies amplification of NIZK proof and argument systems [GJS19, BKP+24, BG24]. Intuitively, a NIZK amplifier takes as input an $(\varepsilon_s, \varepsilon_z)$-weak NIZK—where the soundness and zero-knowledge errors are, respectively, $\varepsilon_s, \varepsilon_z$ for $\varepsilon_s + \varepsilon_z < 1$—and turns it into a NIZK (for the same language) with negligible soundness and zero-knowledge errors. NIZK amplifiers imply NIZK combiners: by selecting at random one of the NIZK proof systems given to the combiner, we get a weak NIZK that we can then amplify. Following this approach, one obtains [BG24]:

- A robust combiner for NIZK arguments assuming polynomially-secure public-key encryption.
- A robust combiner for NIZK proofs assuming polynomially-secure one-way functions.

In contrast, our combiners are unconditional in the sense that they do not require any additional assumptions besides the fact that t out of n of the given NIZK proofs systems are secure.

combiner with input splitting would work.

[8] Naor's construction requires a pseudorandom generator, which can be constructed from one-way functions [HILL99].

Another limitation is that the approach based on amplifiers currently does not seem to preserve the proof/argument of knowledge property of the input protocols, while all of our combiners do. We also note that if we want to design a combiner that works when the input candidates are *proof* systems (with unconditional soundness) and we do not care about the property of PoK/AoK, then we can obtain the following combiner that works for any threshold $t > n/2$. On input n NIZK proof system candidates the combiner constructs a OWF using the technique described in Section 1.3 (§*MPC-in-the-head combiner*), picks a random NIZK input candidate and applies the amplifier of [BG24] (we can use this amplifier given that it only requires the existence of OWFs and a weak NIZK as input). However, such an approach does not work if the input candidates are argument systems, as in this case the best-known combiner [BG24] requires the existence of public-key encryption. In summary, the combiner we have just sketched works only for proof systems, while in this paper we propose (unconditional) combiners that work for both argument and proof systems while preserving the PoK/AoK property (if any) of the input protocols.

2 Notation

We denote the set of natural numbers with $\mathbb{N}$, and the set of integers of prime order q with $\mathbb{Z}_q$. We denote "$\leftarrow$" as the assigning operator (*e.g.*, to assign to a the value of b we write $a \leftarrow b$), and "$\xleftarrow{\$}$" as the sample operator (*e.g.*, $x \xleftarrow{\$} Q$ denotes the sampling of x from a distribution Q). We use "$=$" to check equality of two different elements. We use $\mathsf{poly}(\cdot)$ to indicate a generic polynomial function, $\mathsf{negl}(\cdot)$ to denote a negligible function, and $\mathsf{nonnegl}(\cdot)$ to denote a non-negligible function. We use $\approx_c$ (resp. $\not\approx_c$) to denote computational indistinguishability (resp. distinguishability) of two ensembles. We use $\approx_p$ to denote that two ensembles are identical. While referring to a set T, we denote T_j as the subset containing the first j indices of T. With respect to an NP language $\mathcal{L}$, we denote $\mathscr{R}_{\mathcal{L}}(x, w)$ as its corresponding NP-relation.

3 Defining combiners

Our combiner definitions follow the spirit of the notion for hash-function combiners proposed in [Pie08]. At a high level, in [Pie08], the authors defines black-box combiners for collision resistant hash functions as a tuple of algorithms $(\mathsf{C}, \mathcal{A})$, where C is the algorithm combining n *candidates* (the hash function protocols in the case of [Pie08]). At a high level, the security required from a t-out-of-n combiner requires that if an adversary breaks the security of the combiner, then there exists a reduction that breaks the security of at least $n - t + 1$ candidates. This captures the fact that the security of the combiner relies only on the security of at least t input candidates, and no additional cryptographic assumption is used (*i.e.*, it prevents the combiner from just ignoring the input candidate and creating from scratch a hash function). In this work, we follow a similar blueprint and extend it to the case of non-interactive zero-knowledge proofs, formally defining combiners for each of the properties that one expects from a zero-knowledge proof: correctness, soundness, zero-knowledge, and proof/argument of knowledge. In our definition, we capture the case where the combiner could have *black-box* or *non-black-box* access to the underlying primitives. Our definitions are general enough to also capture both notions of black-box and non-black-box simulation (resp. extraction) for the zero-knowledge (resp. argument/proof of knowledge) case.

Finally, we argue that combiners that satisfy our definition also satisfy the *standard* notions of zero-knowledge, soundness, and argument/proof of knowledge if sufficiently many (t) of the input candidates satisfy the *standard* notions of zero-knowledge, soundness, and argument/proof

of knowledge. Below, we provide formal definitions and the formal arguments of the above claims.

3.1 Formalizing correctness

From a correctness standpoint, our goal consists of combining non-interactive complete protocols, which we formally define as follows.

Definition 1 (Set of Non-Interactive Complete Protocols $\mathcal{F}_{\mathsf{NI}}$ for the language $\mathcal{L}$)**.** *Let* $\Pi = (\mathtt{setup}, \mathcal{P}, \mathcal{V})$ *be a tuple of algorithms defined as follows:*

- $\mathtt{setup}(1^n, 1^\lambda)$ *takes as input a statement of length* n *and the unary description of the security parameter* λ*. It outputs a public parameter* pp*.*
- $\mathcal{P}(pp, x, w)$ *takes as the input a public parameter* pp*, a statement* x *and a witness* w*, s.t.* $(x, w) \in \mathscr{R}_{\mathcal{L}}$*. It outputs a proof* π*.*
- $\mathcal{V}(pp, x, \pi)$ *takes as the input a public parameter* pp*, a statement* x *and a proof* π*. It outputs 1 to accept and 0 to reject.*

Then, $\Pi \in \mathcal{F}_{\mathsf{NI}}$ *if, for all* $\lambda \in \mathbb{N}$ *and all* $(x, w) \in \mathscr{R}_{\mathcal{L}}$*, where* $\mathcal{L}$ *is an* NP *language, it holds that:*

$$\Pr\left[\mathcal{V}(pp, x, \mathcal{P}(pp, x, w)) = 1 \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda).$$

With that, we can properly characterize correctness for $\mathcal{F}_{\mathsf{NI}}$ combiners. Roughly, whenever one defines a combiner C that combines $\mathcal{F}_{\mathsf{NI}}$ candidates, C itself has to be a $\mathcal{F}_{\mathsf{NI}}$.

Definition 2 (Correctness of an n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner)**.** *Let* C *be a PPT algorithm.* C *is a correct* n*-candidate* $\mathcal{F}_{\mathsf{NI}}$*-combiner if the following holds:*

$$\forall PPT\ f_1, \ldots, f_n \in \mathcal{F}_{\mathsf{NI}}, \mathsf{C}^{f_1, \ldots, f_n} \in \mathcal{F}_{\mathsf{NI}}.$$

3.2 Combiners for soundness, ZK, and AoK

In what follows, we provide separate definitions of combiners for soundness, zero-knowledge, and argument-of-knowledge. All our definitions have a label $\alpha \in \{\mathsf{B}, \mathsf{N}\}$ that specifies whether the combiner has black-box access ($\alpha = \mathsf{B}$) or non-black-box access ($\alpha = \mathsf{N}$) to its candidates. Moreover, for zero-knowledge and argument-of-knowledge, we extend our label to $\alpha\beta \in \{\mathsf{B}, \mathsf{N}\}^2$ in order to capture whether the primitive being realized is black-box or non-black-box in the use of the adversary. For instance, a NB combiner has non-black-box access to its candidates and realizes black-box zero-knowledge (*i.e.*, there exists a universal simulator).

3.2.1 Combiners for soundness

To define combiners for soundness, we find it convenient to define the relation $\mathcal{R}_{\mathsf{Sound}}$. This relation contains all the tuple *(protocol, adversary)*, such that the adversary breaks the soundness of the protocol. We will then define a combiners for soundness, as follows. Let C be a *correct* combiner that works having as input n candidates $f_1, \ldots, f_n \in \mathcal{F}_{\mathsf{NI}}$. If there exists $\mathcal{A}$ such that $(\mathsf{C}, \mathcal{A}) \in \mathcal{R}_{\mathsf{Sound}}$ (*i.e.*, C is not sound), then there must exist $n - t + 1$ adversaries (reductions) $\mathsf{Red}_{i_1}, \ldots, \mathsf{Red}_{i_{n-t+1}}$ such that $(f_{i_j}, \mathsf{Red}_{i_j}) \in \mathcal{R}_{\mathsf{Sound}}$ for all $j \in [n - t + 1]$ (*i.e.*, there are at least $n - t + 1$ candidates that are not sound).

Below, we provide the formal definitions of the relation and of a sound combiner.

Definition 3 ($\mathcal{R}_{\mathsf{Sound}}$)**.** *Let* $\Pi \in \mathcal{F}_{\mathsf{NI}}$*, and let* $\mathcal{A}$ *(i.e., the adversary) be a PPT algorithm. We say that* $(\Pi, \mathcal{A}) \in \mathcal{R}_{\mathsf{Sound}}$ *if the following holds:*

$$\Pr\left[\mathcal{V}(pp, x, \pi) = 1 \wedge x \notin \mathcal{L} \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^{\lambda}); (x, \pi) \xleftarrow{\$} \mathcal{A}(pp)\right] \geq \mathsf{nonnegl}(\lambda).$$

Definition 4 (t-out-of-n $\mathsf{type} \in \{\mathsf{B}, \mathsf{N}\}$ Sound Combiners)**.** *Let* C *be a correct* n*-candidate* $\mathcal{F}_{\mathsf{NI}}$*-combiner, and let* Red *be a PPT algorithm (i.e., the reduction). We say that* C *is a* t*-out-of-*n $\mathsf{type} \in \{B, N\}$ *sound combiner if, after ordering the quantifiers according to* type *in Table 2, the following holds:*

$$(\mathsf{C}^{f_1,\ldots,f_n}, \mathcal{A}^{f_1,\ldots,f_n}) \in \mathcal{R}_{\mathsf{Sound}} \implies$$
$$\exists I = \{i_1, \ldots, i_{n-t+1}\} \subseteq [n] \ s.t., \forall i_j \in I, \ (f_{i_j}, \mathsf{Red}^{\mathcal{A}, f_{i_j}}) \in \mathcal{R}_{\mathsf{Sound}}.$$

type	Structural quantifiers	
B	$\exists \mathsf{C}$	$\forall f_1, \ldots, f_n \in \mathcal{F}$
N	$\forall f_1, \ldots, f_n \in \mathcal{F}$	$\exists \mathsf{C}$

Table 2: Classification of sound combiners.

Connections with the *standard* soundness notion. We note that the above definition implies that if the combiner is run having at least t input candidates that are sound according to the standard definition of soundness (we recall this in Definition 16), then it must be that C satisfies the standard notion of soundness as well. This holds because, if t of the input candidates $(f_1, \ldots, f_n)$ are sound, it means that for each of these t candidates no adversary can ever prove a false statement. More formally, if the j-th candidate is sound, then there exists no adversary Red_j, such that $(f_j, \mathsf{Red}_j) \in \mathcal{R}_{\mathsf{Sound}}$. Hence, if there exists an adversary $\mathcal{A}$ such that our combiner is not sound (i.e., $(\mathsf{C}, \mathcal{A}) \in \mathcal{R}_{\mathsf{Sound}}$), then it must be possible to design $n - t + 1$ reductions that contradict the soundness of $n - t + 1$ of the input candidates. But this would contradict the fact that t input candidates are sound.

In a nutshell, if the combiner is executed with t input instances that are sound (under the standard definition of soundness recalled in Definition 16), then the combiner is also sound as per Definition 16.

3.2.2 Combiners for zero-knowledge

Following a similar approach, we define combiners for the zero-knowledge property. First, we define the relation $\mathcal{R}_{\mathsf{ZK}}$. A tuple $(protocol, simulator, adversary)$ belongs to $\mathcal{R}_{\mathsf{ZK}}$ if the adversary distinguishes between proofs generated using the honest prover of the protocol and proofs generated from the simulator.

We then use this relation to formalize the notion of combiners for black-box and non-black-box zero-knowledge by properly ordering the quantifiers for the adversary and the simulator. For example in the case of black-box zero-knowledge combiners, we say that a combiner C, running on input n candidates $f_1, \ldots, f_n$ is secure if the following happens: if for all simulators $\mathcal{S}$ there exists an $\mathcal{A}$ such that $(\mathsf{C}, \mathcal{S}, \mathcal{A})$ belongs to $\mathcal{R}_{\mathsf{ZK}}$, then for all possible choice of simulators $\mathcal{S}_{i_1}, \ldots, \mathcal{S}_{i_{n-t+1}}$ there exists a sequence of adversaries $\mathsf{Red}_{i_1}, \ldots, \mathsf{Red}_{i_{n-t+1}}$ such $(f_{i_j}, \mathcal{S}_{i_j}, \mathsf{Red}_{i_j}) \in \mathcal{R}_{\mathsf{ZK}}$. In other words, if the combiner is not zero-knowledge, then it must be that at least $n-t+1$ input candidates are not zero-knowledge as well. The formal definitions follow.

Definition 5 ($\mathcal{R}_{\mathsf{ZK}}$)**.** *Let* $\Pi \in \mathcal{F}_{\mathsf{NI}}$*, and let* $\mathcal{A}$ *and* $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ *be PPT algorithms. Let* $\texttt{REAL} = \{(pp, x, w, \pi) \mid pp \xleftarrow{\$} \texttt{setup}(1^{|x|}, 1^\lambda);\ (x, w) \xleftarrow{\$} \mathcal{A}(pp);\ \pi \xleftarrow{\$} \mathcal{P}(pp, x, w)\}_{\lambda \in \mathbb{N}}$ *and* $\texttt{IDEAL}_{\mathcal{S}} = \{(pp, x, w, \pi) \mid (pp, \tau) \xleftarrow{\$} \mathcal{S}_0(1^{|x|}, 1^\lambda);\ (x, w) \xleftarrow{\$} \mathcal{A}(pp);\ \pi \xleftarrow{\$} \mathcal{S}_1(pp, \tau, x)\}_{\lambda \in \mathbb{N}}$*. We say that* $(\Pi, \mathcal{A}, \mathcal{S}) \in \mathcal{R}_{\mathsf{ZK}}$ *if the following holds:*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow \mathcal{A}(pp, x, w, \pi) \end{array}; (pp, x, w, \pi) \xleftarrow{\$} \begin{cases} \texttt{REAL} & \text{if } b = 0 \\ \texttt{IDEAL}_{\mathcal{S}} & \text{if } b = 1 \end{cases}\right] \geq \frac{1}{2} + \mathsf{nonnegl}(\lambda).$$

Definition 6 (t-out-of-n $\mathsf{type} \in \{\mathsf{B}, \mathsf{N}\}^2$ Non-Interactive Zero-Knowledge Combiners)**.** *Let* C *be a correct* n*-candidate* $\mathcal{F}_{\mathsf{NI}}$*-combiner. We say* C *is a* t*-out-of-*n $\mathsf{type} \in \{B, N\}^2$ *zero-knowledge combiner if, after having ordered the quantifiers according to* type *in Table 3, the following holds (*$\mathsf{combZKQuant}$*,* $\mathsf{candZKQuant}$ *are also defined in Table 3):*

$$\begin{array}{c} \mathsf{combZKQuant}\ s.t.\ (\mathsf{C}^{f_1,\ldots,f_n}, (\mathcal{A}^{f_1,\ldots,f_n}, \mathcal{S})) \in \mathcal{R}_{\mathsf{ZK}} \implies \\ \exists I = \{i_1, \ldots, i_{n-t+1}\} \subseteq [n]\ s.t.\ \textit{the following holds:} \\ \forall i_j \in I, \mathsf{candZKQuant}\ s.t.\ (f_{i_j}, (\mathsf{Red}^{\mathcal{A}, f_{i_j}}, \mathcal{S}_{i_j})) \in \mathcal{R}_{\mathsf{ZK}}. \end{array}$$

type	Structural quantifiers	combZKQuant	candZKQuant
BB	$\exists \mathsf{C} \forall f_1, \ldots, f_n \in \mathcal{F}$	$\forall \mathcal{S} \exists \mathcal{A}$	$\forall \mathcal{S}_{i_j}\ \exists \mathsf{Red}^{\mathcal{A}, f_{i_j}}$
BN	$\exists \mathsf{C} \forall f_1, \ldots, f_n \in \mathcal{F}$	$\exists \mathcal{A} \forall \mathcal{S}$	$\exists \mathsf{Red}^{\mathcal{A}, f_{i_j}} \forall \mathcal{S}_{i_j}$
NB	$\forall f_1, \ldots, f_n \in \mathcal{F} \exists \mathsf{C}$	$\forall \mathcal{S} \exists \mathcal{A}$	$\forall \mathcal{S}_{i_j}\ \exists \mathsf{Red}^{\mathcal{A}, f_{i_j}}$
NN	$\forall f_1, \ldots, f_n \in \mathcal{F} \exists \mathsf{C}$	$\exists \mathcal{A} \forall \mathcal{S}$	$\exists \mathsf{Red}^{\mathcal{A}, f_{i_j}} \forall \mathcal{S}_{i_j}$

Table 3: Classification of zero-knowledge combiners.

Connections with the *standard* ZK notion. In this discussion, we focus on the BB case, but the same arguments hold for any case.

We observe that, if a combiner that satisfies our definition is executed with at least t input candidates that are *secure*, then the combiner satisfies the *standard* notion of zero-knowledge. When we say a candidate is secure, we mean that it satisfies the standard notion of zero-knowledge (we recall this in Definition 17). This standard (black-box) notion of zero-knowledge requires that there exists a simulator $\mathcal{S}$ such that no adversary can distinguish between transcripts generated using the prover and transcripts generated using $\mathcal{S}$. This, in particular, means that no reduction can exist that contradicts the security of t of the input candidates.

Hence, if our combiner is executed using at least t candidates that are *secure*, then any adversary that contradicts the zero knowledge of our combiner would necessarily contradict the security of $n - t + 1$ of the input candidates. But this would contradict the security of at least one of the candidates that are believed to be *secure*. This implies that, whenever t input candidates satisfy the standard notion of zero knowledge, the protocol obtained by running a combiner satisfies the standard zero-knowledge security notion as well. As such, our combiner can be used in any application where it is needed to have a zero-knowledge protocol.

3.2.3 Combiners for argument-of-knowledge

Following the same approach, we define below the notion of a secure combiner for arguments of knowledge.

Definition 7 ($\mathcal{R}_{\mathsf{AoK}}$)**.** *Let $\Pi \in \mathcal{F}_{\mathsf{NI}}$, and let $\mathcal{A}$ and $\mathcal{E} = (\mathcal{E}_0, \mathcal{E}_1)$ be PPT algorithms. We say that $(\Pi, (\mathcal{A}, \mathcal{E})) \in \mathcal{R}_{\mathsf{AoK}}$ if the following holds:*

$$\Pr\left[\mathcal{A}(pp) = 1 \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^\lambda)\right] \not\approx_c \Pr\left[\mathcal{A}(pp) = 1 \mid (pp, \xi) \xleftarrow{\$} \mathcal{E}_0(1^{|x|}, 1^\lambda)\right],$$

or

$$\Pr\left[(x, w) \in \mathscr{R}_{\mathcal{L}}, \mathcal{V}(pp, x, \pi) = 1 \;\middle|\; \begin{array}{l}(pp, \xi) \xleftarrow{\$} \mathcal{E}_0(1^n, 1^\lambda); (x, \pi) \leftarrow \mathcal{A}(pp) \\ w \leftarrow \mathcal{E}_1(pp, \xi, x, \pi)\end{array}\right] < 1 - \mathsf{negl}(\lambda).$$

Definition 8 (t-out-of-n $\mathsf{type} \in \{\mathsf{B}, \mathsf{N}\}^2$ Non-Interactive Argument-of-Knowledge combiner)**.** *Let C be a correct n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner. We say C is a t-out-of-n type zero-knowledge combiner if, after having ordered the quantifiers according to type in in Table 4, the following holds (*$\mathsf{combZKQuant}$, $\mathsf{candZKQuant}$ *are also defined in Table 4):*

$$\begin{aligned}&\mathsf{combAoKQuant}\ s.t.\ (\mathsf{C}^{f_1,\ldots,f_n}, (\mathcal{A}^{f_1,\ldots,f_n}, \mathcal{E})) \in \mathcal{R}_{\mathsf{AoK}} \implies \\ &\exists I = \{i_1, \ldots, i_{n-t+1}\} \subseteq [n]\ s.t.\ \text{the following holds:} \\ &\forall i_j \in I, \mathsf{candAoKQuant}\ s.t.\ (f_{i_j}, (\mathsf{Red}^{\mathcal{A}, f_{i_j}}, \mathcal{E}_{i_j})) \in \mathcal{R}_{\mathsf{AoK}}.\end{aligned}$$

type	Structural quantifiers	combAoKQuant	candAoKQuant
BB	$\exists \mathsf{C} \forall f_1, \ldots, f_n \in \mathcal{F}$	$\forall \mathcal{E} \exists \mathcal{A}$	$\forall \mathcal{E}_{i_j}\ \exists \mathsf{Red}^{\mathcal{A}, f_{i_j}}$
BN	$\exists \mathsf{C} \forall f_1, \ldots, f_n \in \mathcal{F}$	$\exists \mathcal{A} \forall \mathcal{E}$	$\exists \mathsf{Red}^{\mathcal{A}, f_{i_j}} \forall \mathcal{E}_{i_j}$
NB	$\forall f_1, \ldots, f_n \in \mathcal{F} \exists \mathsf{C}$	$\forall \mathcal{E} \exists \mathcal{A}$	$\forall \mathcal{E}_{i_j}\ \exists \mathsf{Red}^{\mathcal{A}, f_{i_j}}$
NN	$\forall f_1, \ldots, f_n \in \mathcal{F} \exists \mathsf{C}$	$\exists \mathcal{A} \forall \mathcal{E}$	$\exists \mathsf{Red}^{\mathcal{A}, f_{i_j}} \forall \mathcal{E}_{i_j}$

Table 4: Classification of argument-of-knowledge combiners.

Following similar arguments used in the soundness and zero-knowledge cases, we can claim that an AoK combiner executed on input t primitives that satisfies the *standard* notion of AoK (which we recall in Definition 19) also satisfies the *standard* AoK notion.

Remark 1. *Sometimes we also consider statistically sound combiners and proof-of-knowledge combiners. We do not provide explicit definitions for these cases, as it suffices to consider unbounded adversaries within the respective relations (*i.e.*, $\mathcal{R}_{\mathsf{AoK}}, \mathcal{R}_{\mathsf{ZK}}$).*

4 Impossibility Result for $t \leq \lfloor \frac{n}{2} \rfloor$

In this section, we show that no t-out-of-n NIZK combiner exists for NP unless the polynomial hierarchy collapses for $t \leq \lfloor \frac{n}{2} \rfloor$. We also provide an impossibility result for a stronger setting, showing that no t-out-of-n NIZK combiner exists for NP exists even if we allow t of the input candidates to be both sound and zero-knowledge at the same time. The latter impossibility result applies only to combiners that have black-box access to candidates.

Theorem 1. *Let $\{\Pi_i, \ldots, \Pi_n\}$ be a set of non-interactive complete candidates (i.e., $\forall i \in [n], \Pi_i \in \mathcal{F}_{\mathsf{NI}}$). If there exists a t-out-of-n combiner $\mathsf{C}^{\Pi_1,\ldots,\Pi_n}$ for the language $\mathcal{L}$ that is both α sound (according to Definition 4) and $\alpha\beta$ zero-knowledge (according to Definition 6), with $\alpha, \beta \in \{B, N\}$ (i.e., the combiner returns a protocol that is both sound and zero-knowledge), then either $t > \lfloor \frac{n}{2} \rfloor$ or $\mathcal{L} \subseteq \mathsf{coAM} \cap \mathsf{AM}$.*

Proof. Let C be a 1-out-of-2 combiner for some NP-language $\mathcal{L}$. This combiner works assuming that the input protocols are for two languages $\mathcal{L}_{\mathsf{zk}}$ and $\mathcal{L}_{\mathsf{pok}}$. Let us consider the following two NIZK candidates $\Pi_{\mathsf{zk}} = (\mathtt{setup}_{\mathsf{zk}}, \mathcal{P}_{\mathsf{zk}}, \mathcal{V}_{\mathsf{zk}})$ and $\Pi_{\mathsf{pok}} = (\mathtt{setup}_{\mathsf{pok}}, \mathcal{P}_{\mathsf{pok}}, \mathcal{V}_{\mathsf{pok}})$ respectively for the languages $\mathcal{L}_{\mathsf{zk}}$ and $\mathcal{L}_{\mathsf{pok}}$. The setup algorithms of both Π_{zk} and Π_{pok} return nothing. The prover $\mathcal{P}_{\mathsf{zk}}$, on input a statement and a witness, returns 0. The verifier accepts a proof π_{zk} for a statement x only if $\pi_{\mathsf{zk}} = 0$. The prover $\mathcal{P}_{\mathsf{pok}}$ on input a statement x and a witness w returns the witness w. The verifier accepts a proof π only if w is a valid witness for x. In summary, we have that Π_{zk} is unconditional zero-knowledge (but not sound), and Π_{pok} is unconditional sound (but not zero-knowledge).

By assumption, C is a valid combiner that on input Π_{zk} and Π_{pok} returns a new scheme Π. By the definition of secure combiner, we have that if there exists an adversary, running in time t, that attacks successfully the zero-knowledge (the soundness) of Π, then we can design $n - t + 1 = 2$ reductions to the zero-knowledge (the soundness) properties of both schemes, each (the reductions) running in time $\mathsf{poly}(t)$. We are going to prove that it is impossible to construct these two reductions. Proving this implies that the scheme produced by C is unconditional sound and unconditional zero-knowledge, which implies that $L \subseteq \mathsf{coAM} \cap \mathsf{AM}$ due to [Ps05]. To conclude the proof we now need to argue that no adversary (even unbounded) can break the security of C. Assume that an unbounded adversary breaks the zero-knowledge of C. By the definition of secure combiners, this must imply the existence of two (unbounded) reductions, one that breaks the zero-knowledge property of Π_{zk}, and one that breaks the zero-knowledge of Π_{pok}. Let us focus on the reduction to the security for Π_{zk}. A successful reduction $\mathsf{Red}_{\mathsf{zk}}$ needs to tell apart if a proof of Π_{zk} is simulated or not. However, no such $\mathsf{Red}_{\mathsf{zk}}$ can exist, as the zero-knowledge property of Π_{zk} is unconditional. Hence, it is impossible to have two reductions, which implies that the scheme obtained by running C retains its zero knowledge even against unbounded adversaries. Let us now turn our attention to the soundness of C. Similarly, in this case, we need to argue that if there exists a corrupted prover that generates proofs for false statements, then we have a reduction $\mathsf{Red}_{\mathsf{pok}}$ to the soundness of Π_{pok} and a reduction to the soundness of Π_{zk}. We observe that the soundness of Π_{pok} is unconditional, hence, no such reduction $\mathsf{Red}_{\mathsf{pok}}$ can exist. This concludes the proof, as we have argued that the scheme obtained by running C on inputs Π_{zk} and Π_{pok} retains both the zero-knowledge and the sound property against unbounded adversaries. But due to [Ps05], such a scheme exists only for languages in $\mathsf{coAM} \cap \mathsf{AM}$. □

Theorem 2. *Let $T = \{\Pi_i, \ldots, \Pi_n\} \in$ with $\Pi_i \in \mathcal{F}_{\mathsf{NI}}$ for each $i \in [n]$, and let $K \subseteq T$ with $|K| = t$ be such that each $\Pi \in T$ is both sound and zero-knowledge (according to Definitions 16 and 17, respectively). If PRFs exist (as per Definition 11), and there exists a combiner $\mathsf{C}^{\Pi_1,\ldots,\Pi_n}$ that is both a t-out-of-n B sound combiner (according to Definition 4) and a t-out-of-n BB ZK combiner (according to Definition 6) for the language $\mathcal{L}$, then either $t > \lfloor \frac{n}{2} \rfloor$ or $\mathcal{L} \subseteq \mathsf{BPP/poly}$.*

Proof. The high-level idea of the proof consists of arguing that a malicious prover is able to run a valid simulator in a real-world execution of the combiner, thus contradicting the soundness of the combiner. In what follows, we exhibit a proof for $(n, t) = (2, 1)$, which can be trivially extended to the general case of (n, t) with $t \leq \lfloor \frac{n}{2} \rfloor$. We start the proof by assuming that the combiners have oracle access instead of black-box access to the candidates. We then argue how to get rid of such simplification.

Let C be a combiner as per the theorem statement. C has access to two oracles $\mathcal{O}_1$ and $\mathcal{O}_2$, representing the input candidates. If one of the candidates is compromised, this means that the adversary has full control over the oracle representing that primitive (*i.e.*, the adversary can see all the inputs to the oracle and program the replies). We denote with $\mathcal{O}_i^\star$ with $i \in [1, 2]$ the oracle controlled by the adversary. We denote the distribution of honest proofs generated via

the combiner (*i.e.*, proofs generated using the honest prover provided by the combiner) with $\{\mathsf{C}^{\mathcal{O}_1,\mathcal{O}_2}\}$.

Fact 1. Let now $\Pi_1 \in \mathcal{F}_{\mathsf{NI}}$ and $\Pi_2 \in \mathcal{F}_{\mathsf{NI}}$ be both sound and zero-knowledge according to Definition 16 and Definition 17, respectively. By the assumption that Π_1 is zero-knowledge, $\{\mathsf{C}^{\Pi_1,\Pi_2}\} \approx_c \{\mathsf{C}^{\mathcal{S}_1,\Pi_2}\}$.

Fact 2: "The adversary runs the simulated oracle in the real-world". Assume the first oracle is corrupted: $\mathcal{O}_1^\star$ and consider the distribution $\{\mathsf{C}^{\mathcal{O}_1^\star,\Pi_2}\}$. The adversary can program the behaviour of $\mathcal{O}_1$ to match that of $\mathcal{S}_1$ (*e.g.*, by controlling the respective trapdoors). Hence, $\{\mathsf{C}^{\mathcal{S}_1,\Pi_2}\} \equiv \{\mathsf{C}^{\mathcal{O}_1^\star,\Pi_2}\}$.

Fact 3: "Characterizing the simulator of the combiner". By contradiction, C is a 1-out-of-2 BB ZK combiner. This means, in particular, that $\{\mathsf{C}^{\mathcal{O}_1^\star,\Pi_2}\}$ admits a simulator $\mathcal{S}^{\mathcal{O}_1^\star}$ such that $\{\mathsf{C}^{\mathcal{O}_1^\star,\Pi_2}\} \approx_c \{\mathcal{S}^{\mathcal{O}_1^\star}\}$. Crucially, $\mathcal{S}^{\mathcal{O}_1^\star}$ succeeds in simulating the protocol by only leveraging on the secure scheme Π_2 (*e.g.*, by generating its respective trapdoors).

Breaking soundness of the combiner. Let us consider the case in which the adversary, instead, breaks scheme Π_2 and programs its respective oracle $\mathcal{O}_2$ to behave as $\mathcal{S}_2$. For the same argument provided in Facts 1 and 2, it holds that $\{\mathsf{C}^{\Pi_1,\Pi_2}\} \approx_c \{\mathsf{C}^{\Pi_1,\mathcal{S}_2}\}$. Moreover, from Facts 1 and 2, we also have that $\{\mathsf{C}^{\Pi_1,\Pi_2}\} \approx_c \{\mathcal{S}^{\mathcal{O}_1}\}$. Hence, $\{\mathsf{C}^{\Pi_1,\mathcal{O}_2}\} \approx_c \{\mathcal{S}^{\mathcal{O}_1}\}$.

Next, we show that a malicious prover $\mathcal{P}^*$ that controls the oracle $\mathcal{O}_2$ can run $\mathcal{S}^{\mathcal{O}_1}$. To do so, we first observe that $\mathcal{P}^*$ has the same control over $\mathcal{O}_2$ that $\mathcal{S}^{\mathcal{O}_1}$ has. Moreover, any query that $\mathcal{S}^{\mathcal{O}_1}$ would have issued to its oracle $\mathcal{O}_1$ is simply forwarded to the oracle Π_1 of the combiner. Finally, we observe that $\mathcal{S}^{\mathcal{O}_1}$ always outputs accepting proofs even on false statements, unless $\mathcal{L} \in \mathsf{BPP/poly}$. But this contradicts the assumption that C is a 1-out-of-2 B sound combiner.

From oracle access to black-box access. In the proof above, we exclusively provide combiners oracle access to candidates. This means that the oracles are allowed to sample their own internal randomness. Next, we show how to extend our results to the black-box setting by additionally assuming a PRF F. We redefine each oracle $\mathcal{O}$ to a different oracle $\mathcal{O}'$ that, upon input of some randomness r, simply evaluates $F_k(r)$, where k is hardwired inside the $\mathcal{O}'$, and F is a pseudo-random function.

□

5 Combiner for Homomorphic Languages

In this section, we define a t-out-of-n BB NIZKAoK/PoK combiner, with $t > n/2$, that may be used to prove the class of languages described in [Mau09] which are denoted as *homomorphic languages* [Cra96]. A homomorphic language $\mathcal{L}_{\mathsf{HL}}$ is defined with respect to the groups $(\mathsf{G}, \star), (\mathsf{H}, \otimes)$ and a homomorphism $f : \mathsf{G} \to \mathsf{H}$ (*i.e.*, $f(x) \otimes f(y) = f(x \star y)$), and is represented by the set $\{x \mid \exists w \text{ s.t. } f(w) = x\}$.

Some examples of these languages are the languages of discrete logarithm instances, the language of Diffie-Hellman tuples, the languages of valid Pedersen commitments, and many more. We refer to [Mau09] for more details, and also to Appendix B for a few concrete examples. Before presenting how our combiner works, we need to introduce some additional notation and technical tools.

Hyperoperations. Notationally, we use hyperoperations as in [Goo47]. Let w be an element in group G. We denote $w\{2\star\}v$ as the result of $\underbrace{w \star w \star \ldots \star w}_{v \text{ copies of } w}$, with $v \in \mathbb{Z}_q$. We compactly represent $a_0 \star a_1 \star \ldots \star a_n$ with the symbol $\bigstar_{i=0}^{n} a_i$, and $a_0 \otimes a_1 \otimes \ldots \otimes a_n$ with the symbol $\bigotimes_{i=0}^{n} a_i$. Finally, if f is a group homomorphism as defined above, the following facts hold:

$$f(w)\{2\otimes\}v = f(w\{2\star\}v) \tag{1}$$

$$\bigotimes_{j\in[n]} f(w_j) = f(\bigstar_{j\in[n]} w_j) \tag{2}$$

Fact 1 holds, as:

$$f(w)\{2\otimes\}v = \underbrace{f(w) \otimes f(w) \otimes \ldots \otimes f(w)}_{v \text{ copies of } f(w)} = f\,\underbrace{(w \star w \star \ldots \star w)}_{v \text{ copies of } w} = f(w\{2\star\}v)$$

Fact 2 holds, as:

$$\bigotimes_{j\in[n]} f(w_j) = f(w_1) \otimes \ldots \otimes f(w_n) = f(w_1 \star \ldots \star w_n) = f(\bigstar_{j\in[n]} w_j)$$

Shamir's secret sharing scheme. We propose a version of Shamir's secret-sharing scheme that works in $(\mathsf{G}, \star)$, which allows performing the standard sharing and reconstruction operations. Moreover, we design a special reconstruction algorithm that we denote with $\overline{\texttt{reconstruct}}_{\mathsf{H}}$, which allows performing the reconstruction *homomorphically*, exploiting the properties of the homomorphic language. More details follow.

The sharing algorithm $\texttt{share}_{\mathsf{G}}$ takes as input w, where w is an element of group G, and samples $t-1$ random group elements $a_1, \ldots, a_{t-1}$ by using the generation algorithm $\texttt{Gen}_{\mathsf{G}}(1^\lambda)$ (this algorithm just samples uniformly at random elements from G). A polynomial p of degree $t-1$ is defined by using coefficient a_i, and by setting $a_0 = w$. The shares are computed by evaluating $p(i)$, for $i \in [n]$:

$$\texttt{share}_{\mathsf{G}}(w)\text{: } a_1, \ldots, a_{t-1} \xleftarrow{\$} \texttt{Gen}_{\mathsf{G}}(1^\lambda). \text{ Output } \{w_i \leftarrow w \star \bigstar_{j=1}^{t-1} (a_j\{2\star\}i^j)\}_{i\in[n]}$$

The $\texttt{reconstruct}_{\mathsf{G}}$ algorithm takes as input a subset $S = \{s_i\}_{i\in I}$ of t shares, where $I = i_1, \ldots, i_t \in \mathbb{Z}_q^t$ is a set of evaluation points of the polynomial, and uses Lagrange interpolation to recompute $p(0)$ (*i.e.*, the secret):

$$\texttt{reconstruct}_{\mathsf{G}}(\{s_i\}_{i\in I})\text{: Output } w \leftarrow \bigstar_{j\in I} (s_i\{2\star\}(\prod_{k\in I, k\neq j} \frac{k}{k-j}))$$

Since Shamir's secret sharing scheme relies on Lagrange interpolation, it is possible to reconstruct points in the polynomial that are different from the secret (*i.e.*, points evaluated in $l \neq 0$). To this extent, we denote the modified version of $\texttt{reconstruct}_{\mathsf{G}}$ as $\overline{\texttt{reconstruct}}_{\mathsf{G}}$. $\overline{\texttt{reconstruct}}_{\mathsf{G}}$ takes as input a subset $S = \{s_i\}_{i\in I}$ of t shares and an index $l \notin I$ of the point to be reconstructed, and works as follows:

$$\overline{\texttt{reconstruct}}_{\mathsf{G}}(\{s_i\}_{i\in I}, l)\text{: Output } w_l \leftarrow \bigstar_{j\in I} (s_i\{2\star\}(\prod_{k\in I, k\neq j} \frac{l-k}{j-k})).$$

We remark that $\overline{\texttt{reconstruct}}_{\mathsf{G}}(\cdot, 0) = \texttt{reconstruct}_{\mathsf{G}}(\cdot)$.

Finally, if f is a group homomorphism as defined above, having access to a subset S of size t of functions of the shares (*i.e.*, $\{f(s_i)\}_{i\in I}$) allows the reconstruction of a function of the point

in the polynomial (*i.e.*, $x_l = f(s_l)$ for $l \notin I$) as follows:

$$x_l \leftarrow \bigotimes_{j \in I} (f(s_i)\{2\otimes\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k}))$$

We denote the reconstruction operation above as $\overline{\texttt{reconstruct}}_{\mathsf{H}}(\{x_i\}_{i \in I}, l)$. The correctness of this operation follows by applying Fact 1 and Fact 2:

$$\begin{aligned} x_l \leftarrow \bigotimes_{j \in I} (f(s_i)\{2\otimes\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) &= f(\bigstar_{j \in I} (s_i\{2\star\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k}))) \\ &= f(\overline{\texttt{reconstruct}}_{\mathsf{G}}(\{s_i\}_{j \in I}, l) = f(w_l) \end{aligned}$$

In the rest of the paper, whenever we talk about homomorphic languages $\mathcal{L}_{\mathsf{HL}}$ with respect to the groups $(\mathsf{G}, \star), (\mathsf{H}, \otimes)$ and a homomorphism $f : \mathsf{G} \to \mathsf{H}$, we implicitly require that the algorithms $(\texttt{share}_{\mathsf{G}}, \texttt{reconstruct}_{\mathsf{G}})$, as defined here, satisfy correctness and perfect security of Shamir's secret sharing scheme (respectively, Definition 14 and Definition 15). Looking ahead, the correctness property will be used within the proof of Theorem 3, and the perfect security property will be used within the proof of Theorem 5. In Appendix B we give a few concrete examples of such homomorphic languages and associated sharing and reconstruction operations.

5.1 Combiner description

We are now ready to describe how our combiner works. Let $\{\Pi_k = (\texttt{setup}_k, \mathcal{P}_k, \mathcal{V}_k)\}_{k \in [n]}$ be non-interactive protocols for the homomorphic language $\mathcal{L}_{\mathsf{HL}}$ defined above. These protocols represent the input primitives of the combiner. Let x be the theorem the verifier wants to verify. At a high level in our scheme the prover secret shares the witness w using our secret sharing scheme, and computes n *sub-statements* by evaluating f on each obtained share. Then the prover proves that each of these sub-statements belongs to the homomorphic language, using the input candidates (the k-th input candidate Π_k is used to prove the k-th sub-statement). The verifier picks the first t sub-statements and homomorphically reconstructs the polynomial in H by running $\overline{\texttt{reconstruct}}_{\mathsf{H}}$. The verifier then accepts if all the proofs are accepting, and the homomorphic reconstruction matches with the theorem x and with all of the sub-statements. We formally describe our combiner $\mathsf{C}_{\mathsf{HL}} = (\mathsf{C}_{\mathsf{setup}}, \mathsf{C}_{\mathsf{prove}}, \mathsf{C}_{\mathsf{vrfy}})$ in Figure 5.1:

Figure 5.1: Combiner C_{HL}

- $\mathsf{C}_{\mathsf{setup}}(1^l, 1^\lambda)$: Compute $pp_k \xleftarrow{\$} \texttt{setup}_k(1^l, 1^\lambda)$, for $k \in [n]$. Output $pp \leftarrow \{pp_k\}_{k \in [n]}$.
- $\mathsf{C}_{\mathsf{prove}}(pp, x, w)$:
  - Compute $(w_1, \ldots, w_n) \xleftarrow{\$} \texttt{share}_{\mathsf{G}}(w)$.
  - Compute $x_k \leftarrow f(w_k)$, for $k \in [n]$.
  - Compute $\pi_k \leftarrow \mathcal{P}_k(pp_k, x_k, w_k)$, for $k \in [n]$.
  - Output $\pi \leftarrow (\{\pi_k\}_{k \in [n]}, \{x_k\}_{k \in [n]})$.
- $\mathsf{C}_{\mathsf{vrfy}}(pp, x, \pi)$:
  - Parse (pp, π) as $(\{pp_k\}_{k \in [n]}, \{\pi_k\}_{k \in [n]}, \{x_k\}_{k \in [n]})$. Set $x_0 \leftarrow x$.
  - If $\mathcal{V}_k(pp_k, x_k, \pi_k)$ outputs 0 for some $k \in [n]$, output 0. Otherwise, for each $k \in [n] \setminus [t] \cup \{0\}$, behave as follows:
    - Compute $x'_k \leftarrow \overline{\texttt{reconstruct}}_{\mathsf{H}}(\{x_j\}_{j \in [t]}, k)$.
    - Check whether $x'_k = x_k$. If the check fails then return 0.
  - Return 1.

5.2 Proof sketch

Before formally stating and proving our theorems, we provide a high-level overview of how our proofs work.

Soundness. To show that C_{HL} is sound, we argue that the only way the adversary can produce an accepting proof for a statement $x \notin \mathcal{L}_{\mathsf{HL}}$ is by having sufficiently-many reductions (*i.e.*, $n - t + 1$) to soundness of the underlying candidates. Suppose by contradiction that a prover can make a verifier accept a proof for a false statement (*i.e.*, x is s.t. $f(w) \neq x$ for all w) and that only $n - t$ reductions to the soundness are possible. This means that there are at least t sub-statements that are valid (*i.e.*, they belong to $\mathcal{L}_{\mathsf{HL}}$). Moreover, if the verifier of our combiner has accepted the proof, this means that the witnesses for all the sub-statements represent points that belong to the same polynomial of degree $t - 1$. In particular, this polynomial is univocally determined by the witnesses of the t valid sub-statements (that exist by contradiction). Given that this polynomial evaluated on 0 is equal to x (this holds because the verifier has accepted the proof) then the correctness of our secret sharing scheme and the homomorphic property of the language guarantees that x belongs to the language.

Zero-knowledge. Proving the zero-knowledge of this (and of the other schemes we propose) requires some particular care. The traditional way to prove the zero-knowledge of our protocol would be to design a zero-knowledge simulator that executes the simulators of t of the candidates. To prove that the output of this simulator is indistinguishable from the output of an adversarial verifier when interacting with an honest prover, we design a chain of indistinguishable hybrid experiments, the first corresponding to the experiment where the proof is generated via an honest prover, and the last corresponding to the simulated experiment. If by contradiction our simulator is not good (*i.e.*, zero-knowledge does not hold) this would imply that a pair of adjacent hybrids is not indistinguishable, hence, a reduction to a cryptographic primitive can be performed.

In the proof we have just sketched, we have that if by contradiction zero-knowledge does not hold, then we have *one* reduction. However, our security definition states that if zero-knowledge does not hold, then it must be possible to create $n-t+1$ reductions. Clearly, the proof approach we have discussed does not achieve what we want. Indeed, the above approach does not really make sense as our simulator does not know for which of the candidates the simulator should be run.

We recall that our definition of combiner states that if the combiner is not zero-knowledge then it should be possible to perform $n - t + 1$ reductions to zero-knowledge of the candidates. In particular, this means that if the combiner is not zero-knowledge, then every zero-knowledge simulator we design can be attacked. The intuition is that to prove the security of our scheme, we need to design multiple simulators, such that any time an adversary wins against one simulator, we can create a new reduction to different input candidates. We construct these multiple simulators as follows.

Each simulator $\mathcal{S}^T$ is parameterized by a set T containing t distinct indices in $[n]$, and runs the simulators for the candidates whose indices are in T. We design a sequence of hybrid arguments in which we gradually replace all the sub-proofs generated by all the candidates in T, using real shares of the witness, with simulated sub-proofs. After these sub-proofs are simulated, we replace the $n - t$ remaining shares of the witness with random group elements. Because there cannot exist a good simulator, the output of $\mathcal{S}^T$ must be distinguishable, for all $T \subseteq [n]$ with $|T| = t$. But this means that there exists an adversary $\mathcal{A}$ that is able to distinguish two adjacent hybrids of the hybrid chain (say, $\mathcal{H}_{i^*-1}$ and $\mathcal{H}_{i^*}$), and we can therefore state that

there exists one reduction to zero-knowledge of candidate Π_{i^*}.

Intuitively, whenever some $\mathcal{A}$ breaks zero-knowledge of candidate Π_{i^*} in $\mathcal{S}^T$ by distinguishing the output of a specific protocol Π_{i^*}, it also breaks all the other $\mathcal{S}^{T'}$ for $T \neq T'$ that contain candidate Π_{i^*} within their chain of hybrids. However, there still exist simulators that do not include Π_{i^*} as part of their hybrid chain. Therefore, (a possibly different) $\mathcal{A}$ has to break zero-knowledge of another candidate. By a combinatorics argument, the only way *all* $\binom{n}{t}$ simulators can be broken is by breaking zero-knowledge for at least $n - t + 1$ candidates.

Argument of Knowledge. The proof for the AoK property is similar in spirit to the zero-knowledge one: we construct an extractor $\mathcal{E}^T$ which is parameterized by a set T containing t distinct indices in $[n]$, and runs the extractors for the candidates in T. Then, we show that either $\mathcal{E}^T$ successfully extracts a witness, or there exists at least one reduction to AoK of one of the candidates whose indices are in T.

The first portion of the proof consists of an hybrid argument in which we gradually replace all the public parameters generated by setup algorithms of the candidates with ones generated by trapdoor setup algorithms. As for zero-knowledge, if some adversary detects any variation within the hybrid chain, we have a reduction to argument-of-knowledge of a specific candidate.

After that, the extractor attempts to run all the extractors for the primitives indexed by T to recompute a witness from an accepting proof (x, π) supplied by the adversary. For that, we argue that the only way the extractor may output the witness is by successfully running the extractor for all the candidates whose indices are in T. Hence, it suffices for one extractor of the candidates to fail to cause $\mathcal{E}^T$ to fail. For this scheme this is indeed the case, as each extractor extracts a share of the witness w_i, and the only way w can be extracted is by running the reconstruction procedure of Shamir's secret sharing scheme with these t shares.

The proof concludes with the same combinatorics argument used for the proof of zero-knowledge, with the only difference being that reductions may exist either because some adversary distinguishes the setup generation algorithm of one of the candidates, or because some adversary produces an accepting proof (x, π) for which a candidate fails to extract a share of the witness.

5.3 Formal analysis

Below we provide the formal theorems that we have just sketched above.

Theorem 3. *The construction in Figure 5.1 is a correct n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner for the language $\mathcal{L}_{\mathsf{HL}}$, as per Definition 2, for $n, t \in \mathsf{poly}(\lambda)$.*

Theorem 4. *The construction in Figure 5.1 is a t-out-of-n B sound combiner for the language $\mathcal{L}_{\mathsf{HL}}$, as per Definition 4, for $t > \frac{n}{2}$ and $n, t \in \mathsf{poly}(\lambda)$.*

Theorem 5. *The construction in Figure 5.1 is a t-out-of-n BB ZK combiner for the language $\mathcal{L}_{\mathsf{HL}}$, as per Definition 6, for $t > \frac{n}{2}$ and $n, t \in \mathsf{poly}(\lambda)$.*

Theorem 6. *The construction in Figure 5.1 is a t-out-of-n BB AoK combiner for the language $\mathcal{L}_{\mathsf{HL}}$, as per Definition 8, for $t > \frac{n}{2}$ and $n, t \in \mathsf{poly}(\lambda)$.*

As already remarked in the technical overview, if the input candidates are statistically sound (resp. proof-of-knowledge), Theorem 4 (resp. Theorem 6) yield a statistically-sound (resp. proof-of-knowledge) combiner. The formal proofs of these theorems are deferred, respectively, to Appendix C.1, Appendix C.2, Appendix C.3, and Appendix C.4.

6 Recursive proof combiner

In this section we define a t-out-of-n NB NIZKAoK/PoK combiner, with $t > \frac{n}{2}$ and $n, t \in O(1)$, that generates proofs by "nesting" proofs of the candidates.

6.1 Nesting proof systems

Nesting proofs fundamentally entails two operations:

- Computing the "innermost" proof. The innermost proof is generated by using a proof system to prove knowledge of a witness w for a statement x with respect to language $\mathcal{L}$.
- Computing an "intermediate" proof. An intermediate proof consists of using a proof system to prove knowledge of an accepting proof π generated by another (possibly different) proof system. More formally, the NP-language for a scheme Π_j that proves knowledge of an accepting proof generated with scheme Π_{j-1} is defined as $\mathcal{L}_j = \{(\mathcal{V}_{j-1}, pp_{j-1}, x_{j-1}) \mid \exists \pi_{j-1} \ s.t. \ \mathcal{V}_{j-1}(pp_{j-1}, x_{j-1}, \pi_{j-1}) = 1\}$.

The second operation can be applied recursively, and the last intermediate proof will be output in the clear. Intuitively, if such a proof is accepting, it proves that the entire chain of proofs was generated correctly. Moreover, the presence of intermediate proofs is what makes the combiner NB, as the construction is quantified after the specific instantiations of the candidates $\Pi_1, \ldots, \Pi_n$ are quantified.

6.2 Combiner description

The combiner takes as input n non-interactive protocols $\Pi_k = (\mathtt{setup}_k, \mathcal{P}_k, \mathcal{V}_k) \in \mathcal{F}_{\mathsf{NI}}$ as per Definition 1, for $k \in [n]$, which work for any NP language $\mathcal{L}$. From a high-level perspective, the prover generates $\binom{n}{t}$ proofs, each with a different subset of t out of n candidates. For each proof, the candidate with the lowest index generates the innermost proof by using statement x and witness w. After that, intermediate proofs are generated recursively by using all the other candidates in the specific subset. We formally describe our combiner $\mathsf{C}_{\mathsf{PoP}} = (\mathsf{C}_{\mathsf{setup}}, \mathsf{C}_{\mathsf{prove}}, \mathsf{C}_{\mathsf{vrfy}})$ in Figure 6.1.

Figure 6.1: Combiner $\mathsf{C}_{\mathsf{PoP}}$

- $\mathsf{C}_{\mathsf{setup}}(1^l, 1^\lambda)$: Compute $pp_k \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$, for $k \in [n]$. Output $pp \leftarrow \{pp_k\}_{k \in [n]}$.
- $\mathsf{C}_{\mathsf{prove}}(pp, x, w)$:
  - Index all possible subsets containing t of the n candidates with set $S = \{S^i\}_{i \in \binom{n}{t}}$.
  - For $S^i \in S$, denote $\{s_{i,1}, \ldots, s_{i,t}\}$ as the indices of the candidates for this specific subset sorted in lexicographic order. Then, compute proof π_i^{PoP} as follows:
    - Compute the innermost proof $\pi_{s_{i,1}} \leftarrow \mathcal{P}_{s_{i,1}}(pp_{s_{i,1}}, x, w)$. The language associated to this proof is $\mathcal{L}_{s_{i,1}} = \{x \mid \exists w \ s.t. \ (x, w) \in \mathscr{R}_{\mathcal{L}}\}$.
    - For $j \in [2, \ldots, t]$, let $x_{s_{i,j}} \leftarrow (\mathcal{V}_{s_{i,j-1}}, pp_{s_{i,j-1}}, x_{s_{i,j-1}})$. Compute intermediate proof $\pi_{s_{i,j}} \leftarrow \mathcal{P}_{s_{i,j}}(pp_{s_{i,j}}, x_{s_{i,j}}, \pi_{s_{i,j-1}})$. The language associated to the j-th proof is $\mathcal{L}_j = \{x_{s_{i,j}} \mid \exists \pi_{s_{i,j-1}} \ s.t. \ \mathcal{V}_{s_{i,j-1}}(pp_{s_{i,j-1}}, x_{s_{i,j-1}}, \pi_{s_{i,j-1}}) = 1\}$.
    - Set $\pi_i^{\mathsf{PoP}} \leftarrow \pi_{s_{i,t}}$.
  - Set $\pi \leftarrow \{\pi_i^{\mathsf{PoP}}\}_{i \in \binom{n}{t}}$. Output π.
- $\mathsf{C}_{\mathsf{vrfy}}(pp, x, \pi)$:

- Parse π as $\{\pi_i^{\mathsf{PoP}}\}_{i\in\binom{n}{t}}$.
- Index all possible subsets containing t of the n candidates with set $S = \{S^i\}_{i\in\binom{n}{t}}$.
- For $S^i \in S$, denote $\{s_{i,1}, \ldots, s_{i,t}\}$ as the indices of the candidates for this specific subset sorted in lexicographic order. Then, run $\mathcal{V}_{s_{i,t}}(pp_{s_{i,t}}, x_{s_{i,t}}, \pi_i)$.
- If any of the verifier runs above outputs 0, output 0. Otherwise, output 1.

6.3 Formal analysis

In what follows, we provide formal theorems for correctness, zero-knowledge, and argument-of-knowledge of this construction.[9] Similarly to the previous section, as long as the input candidates are proof-of-knowledge, Theorem 9 yields a proof-of-knowledge combiner.

Theorem 7. *The construction in Figure 6.1 is a correct n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner for any* NP *language $\mathcal{L}$, as per Definition 2, for $n, t \in O(1)$.*

The formal proof of correctness is deferred to Appendix C.5.

Theorem 8. *The construction in Figure 6.1 is a t-out-of-n NB ZK combiner for any* NP *language $\mathcal{L}$, as per Definition 6, for $t > \frac{n}{2}$ and $n, t \in O(1)$.*

Proof (Theorem 8). The structure of this proof is the same as the one sketched in Section 5.2. First, we construct a simulator $\mathcal{S}^T$, and argue that either its output is indistinguishable from the output of the real prover, or there exists at least one reduction to zero-knowledge of one of the candidates whose indices are in T. We describe such a $\mathcal{S}^T$ in Lemma 1. Given that this lemma follows traditional proof techniques, we defer its analysis to Appendix C.6. Then, we argue in Lemma 2 that the existence of adversaries who break zero-knowledge of the combiner implies the existence of sufficiently-many reductions to zero-knowledge of the underlying candidates, concluding the proof.

□

Lemma 1. *There exists a simulator $\mathcal{S}^T$ for which either $\{\texttt{REAL}\} \approx_c \{\texttt{IDEAL}_{\mathcal{S}^T}\}$, or there exists a reduction to zero-knowledge of at least one of the candidates $\{\Pi_k\}_{k\in T}$.*

Lemma 2. *The existence of adversaries breaking zero-knowledge of all possible simulators $\mathcal{S}^T$, as described in Lemma 1, yields reductions to zero-knowledge of at least $n - t + 1$ distinct candidates.*

Proof. Let $\mathtt{S}_0$ be a set of cardinality $\binom{n}{t}$ containing all the possible instantiations of $\mathcal{S}^T$ as described in Lemma 1 (*i.e.*, by considering all the possible choices of T out of the n candidates). The fact that the combiner is not zero-knowledge (as per Definition 6) implies that no simulator in $\mathtt{S}_0$ can be such that $\{\texttt{REAL}\} \approx_c \{\texttt{IDEAL}_{\mathcal{S}^T}\}$. Moreover, Lemma 1 prescribes that if $\{\texttt{REAL}\} \not\approx_c \{\texttt{IDEAL}_{\mathcal{S}^T}\}$, there exists a reduction to at least one of the candidate schemes identified by an index in T. Let $i_1 \in [n]$ be such an index. This means that there exists an adversary $\mathcal{A}$ is able to break zero-knowledge for scheme Π_{i_1}, *i.e.*, $\mathcal{A}$ breaks all possible simulators of Π_{i_1}. Observe that *any* simulator that is parameterized by a T containing i_1 is such that $\{\texttt{REAL}\} \not\approx_c \{\texttt{IDEAL}_{\mathcal{S}^T}\}$. However, there still exists a set $\mathtt{S}_1$ of cardinality $\binom{n-1}{t}$ containing all the simulators

[9] For this scheme, we do not prove soundness in isolation, as the only way we know how to prove such property would be to additionally assume sufficiently-many candidates to be AoK. Nevertheless, the proof we show for AoK suffices, given that AoK implies soundness.

parameterized by a T that do *not* contain i_1. Since there should exist (a possibly different) $\mathcal{A}$ that breaks all the simulators in $\mathtt{S}_1$, an index $i_2 \in [n]$ distinct from i_1 exists. This means that $\mathcal{A}$ is able to break zero-knowledge for scheme Π_{i_2}. Generalizing the above, we denote I_j as the set of indices of cardinality j for which some $\mathcal{A}$ is able to break zero-knowledge of the respective candidate Π_{i_k}. We further denote $\mathtt{S}_j$ as the set of simulators that are parameterized by a T that does not contain any $i_k \in I_j$. The cardinality of $\mathtt{S}_j$ is $\binom{n-j}{t}$. Since $\mathcal{A}$ must also break all the simulators in $\mathtt{S}_j$, there exists another index $i^* \in [n] \setminus I_j$ related to a scheme Π_{i^*} for which $\mathcal{A}$ breaks its ZK property.

In order to have $|\mathtt{S}_j| = 0$, it suffices to show that $t > n - j$. The minimum j for which this condition holds is $j = n - t + 1$, as $\binom{t-1}{t} = 0$. Hence, the existence of adversaries breaking zero-knowledge of the combiner implies the existence of at least $n-t+1$ reductions to zero-knowledge of distinct candidates, concluding the proof. □

Theorem 9. *The construction in Figure 6.1 is a t-out-of-n NB $\mathcal{P}_{\mathsf{AoK}}$-combiner for any* NP *language $\mathcal{L}$, as per Definition 8, for $t > \frac{n}{2}$ and $n, t \in O(1)$.*

Proof (Theorem 9). Following Section 5.2, we construct an extractor $\mathcal{E}^T$ and prove in Lemma 3 that either it successfully extracts a witness, or there exists at least one reduction to AoK of one of the candidates whose indices are in T. Given that this lemma follows traditional proof techniques, we defer its analysis to Appendix C.7. From there, one can exhibit an argument similar to Lemma 2 to argue that the existence of adversaries who break AoK of the combiner implies the existence of sufficiently-many reductions to AoK of the underlying candidates, concluding the proof. This step is the same for all our combiners, and is formally proved in Lemma 12. □

Lemma 3. *There exists an extractor $\mathcal{E}^T$ for which either a witness w is successfully extracted from an accepting proof, or there exists a reduction to argument-of-knowledge of at least one of the candidates $\{\Pi_k\}_{k \in T}$.*

7 MPC-in-the-head approach

In this section we define a t-out-of-n NB NIZKPoK combiner, with $t > \lfloor \frac{2}{3} \rfloor n$, that relies on MPC-in-the-head techniques and may be used for all of NP.

7.1 Notation and building blocks

In what follows, we introduce notation and building blocks we use specifically within this combiner.

7.1.1 Secure Multiparty Computation (MPC) Definitions

We follow the same definition from [IKOS07]. An n-party protocol $\Pi = (P_1, \ldots, P_n)$ is an n-tuple of probabilistic polynomial-time (PPT) interactive Turing machines (ITMs). Each party P_i is initialized with a local input $w_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. All the parties share a public input $x \in \{0,1\}^*$. In this paper, we mainly focus on R-round MPC protocols that securely realize an n-party functionality f, where f maps the input $(x, w_1, \ldots, w_n)$ to an n-tuple of outputs (when only a single output is specified, this output is assumed to be given to all parties).

In this paper, instead of letting every party share the input x, we hardcode x in the function f. All the MPC protocols we use will have this structure; hence, this will be implicit from here onwards.

7.1.2 Notation for MPC

Rather than viewing a protocol Π as an n-tuple of interactive Turing machines, it is convenient to view each Turing machine as a sequence of multiple algorithms: $\texttt{frst-msg}_i$, to compute P_i's first messages to its peers; $\texttt{nxt-msg}_i^k$, to compute P_i's k-th round messages for $2 \leq k \leq R$; and $\texttt{output}_i$, to compute P_i's output. Thus, a protocol Π can be defined as $\{(\texttt{frst-msg}_i, \texttt{nxt-msg}_i^k, \texttt{output}_i)\}_{i\in[n],k\in[R]\setminus\{1\}}$. The syntax of the algorithms is as follows:

- $\texttt{frst-msg}_i(w_i; r_i) \to (\mathsf{msg}_{i\to 1}^1, \ldots, \mathsf{msg}_{i\to n}^1)$ produces the first-round messages of party P_i to all parties. If the message is sent over a broadcast channel, it holds that $\mathsf{msg}_{i\to 1}^1 = \mathsf{msg}_{i\to 2}^1 = \cdots = \mathsf{msg}_{i\to n}^1$. Note that a party's message to itself can be considered to be its state.
- $\texttt{nxt-msg}_i^k(w_i, \{\mathsf{msg}_{j\to i}^l\}_{j\in[n],l\in[k-1]}; r_i) \to (\mathsf{msg}_{i\to 1}^k, \ldots, \mathsf{msg}_{i\to n}^k)$ produces the k-th round messages of party P_i to all parties.
- $\texttt{output}_i(w_i, \{\mathsf{msg}_{j\to i}^k\}_{j\in[n],k\in[R]}; r_i) \to y_i$ produces the output returned to party P_i.

We note that, unless needed, to not overburden the notation, we do not pass the random coin r as an explicit input of the cryptographic algorithms. In addition, we define the view of party P_i and the consistency of views as follows:

Definition 9 (View)**.** *The view* view_i *of party* P_i *is defined as the input* w_i*, the randomness* r_i*, and the all the messages* P_i *sends (denoted as* $\mathsf{msg}_{i,\mathsf{out}}$*, with* $\mathsf{msg}_{i,\mathsf{out}} \leftarrow \{\mathsf{msg}_{i\to j}^k\}_{j\in[n]\setminus\{i\},k\in[R]}$*) and receives (denoted as* $\mathsf{msg}_{i,\mathsf{in}}$*, with* $\mathsf{msg}_{i,\mathsf{in}} \leftarrow \{\mathsf{msg}_{j\to i}^k\}_{j\in[n]\setminus\{i\},k\in[R]}$*) during the execution of the MPC protocol.*

Definition 10 (View Consistency)**.** *A pair of views* view_i *and* view_j *are consistent (with respect to the protocol* Π *realizing* f_x*) if* $\{\mathsf{msg}_{i\to j}^k\}_{k\in[R]}$ *is identical to* $\{\mathsf{msg}_{j\to i}^k\}_{k\in[R]}$ *and vice versa.*

7.1.3 Building blocks

This combiner requires the following primitives:

- n non-interactive protocols $\Pi_k = (\texttt{setup}_k, \mathcal{P}_k, \mathcal{V}_k) \in \mathcal{F}_{\mathsf{NI}}$ as per Definition 1, for $k \in [n]$, which work for any NP language $\mathcal{L}$.
- A t-out-of-n secret sharing scheme $\mathtt{SS} = (\texttt{share}, \texttt{reconstruct})$, as per Definition 13.
- A statistically binding and computationally hiding non-interactive string commitment $\mathtt{NIC} = (\texttt{setup}, \texttt{commit}, \texttt{open})$, as per Definition 12.
- An MPC protocol $\Pi_{\mathsf{MPC}} = \{(\texttt{frst-msg}_i, \texttt{nxt-msg}_i^k, \texttt{output}_i)\}_{i\in[n],k\in[R]\setminus\{1\}}$ that realizes the function f_x (Figure 7.1) with perfect $\mathcal{K}$-robustness (as per Definition 23) in the malicious model and perfect $\mathcal{K}$-privacy (as per Definition 22) in the semi-honest model, where $\mathcal{K} = (n - t)$.

Figure 7.1: Function f_x

Inputs: $w_1, \ldots, w_n$.
Hardcoded: The statement x.
Output: If $(x, \texttt{reconstruct}(\{w_j\}_{j\in[t]})) \in \mathscr{R}_{\mathcal{L}}$, output 1. Otherwise output 0.

7.2 From NIZK candidates to non-interactive commitments

Among the primitives for this construction, we require a commitment scheme that is computationally hiding. This is incompatible with our formalization of combiners, given that an adversary breaking the security of the combiner can now yield a reduction to the newly-introduced (computational) assumption, rather than $n - t + 1$ reductions to the security of the underlying candidates. In what follows, we show that it is possible to instantiate statistically binding non-interactive string commitments by exclusively relying on the security of the candidates. Formally, we prove the following lemma:

Lemma 4. *Let* $\{\Pi_k = (\mathtt{setup}_k, \mathcal{P}_k, \mathcal{V}_k)\}_{k\in[n]} \in \mathcal{F}_{\mathsf{NI}}$ *be non-interactive protocols, as per Definition 1, for any* NP *language* $\mathcal{L}$*. Let* t *of the candidates be sound and* t *of the candidates be zero-knowledge, with* $t > n/2$*. Assuming that* $\mathsf{NP} \subseteq \mathsf{ZKA}$ *and that* $\mathsf{ZKA} \not\subseteq \mathsf{ioP/poly}$*, there exists an efficient construction of statistically binding non-interactive string commitments in the CRS model.*

Proof. The high-level idea of the proof consists in expanding the following chain of implications: t-out-of-n NIZK candidates $\implies$ 1-out-of-n OWFs $\implies$ OWF $\implies$ PRG $\implies$ non-interactive commitments.

- **t-out-of-n NIZK candidates $\implies$ 1-out-of-n OWF candidates.** Let $\mathtt{S}_{\mathsf{snd}}$ be the set of sound candidates and $\mathtt{S}_{\mathsf{zk}}$ be the set of ZK candidates. Since $t > n/2$, $\mathtt{S}_{\mathsf{snd}} \cap \mathtt{S}_{\mathsf{zk}} \neq \emptyset$. Hence, at least one candidate is both sound and zero-knowledge. Given that we assume $\mathsf{NP} \subseteq \mathsf{ZKA}$ and that $\mathsf{ZKA} \not\subseteq \mathsf{ioP/poly}$, we can invoke [LMP24, Theorem 1.1] on each of the candidates, obtaining a set of n candidate functions $\{f_i\}_{i\in[n]}$. Since at least one of the candidates is both sound and zero-knowledge, at least one of the functions will be a OWF.
- **1-out-of-n OWF candidates $\implies$ OWF.** [Her05, Lemma 4.1] shows a combiner that splits the input among the function candidates (*e.g.*, by XOR), and then simply concatenates the output of all the candidates. More explicitly, as long as there exists an index i for which f_i is a OWF, $f(x) = f_1(x_1)||\dots||f_n(x_n)$ is also a OWF, with $x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$.
- **OWF $\implies$ PRG.** By [HILL99], there exists a PRG that may be instantiated by a OWF.
- **PRG $\implies$ NI-Com.** Following [MP12], Naor's bit commitment scheme [Nao91] can be made non-interactive by moving the randomness of the receiver to the CRS that is statistically binding and computationally hiding. Once we have a non-interactive bit commitment scheme, we also have a non-interactive string commitment scheme.

□

Further remarks. Proving this lemma requires that $\mathsf{NP} \subseteq \mathsf{ZKA}$ and that $\mathsf{ZKA} \not\subseteq \mathsf{ioP/poly}$. As hinted throughout the technical overview, if the first assumption does not hold, it means that there exists a language in NP that does not admit zero-knowledge arguments. Hence, no combiner for all of NP can exist. From here, if the second assumption is falsified, it would imply that $\mathsf{NP} \subseteq \mathsf{ioP/poly}$. At the same time, This transformation is also the reason we restrict our combiner to only proofs instead of arguments. Looking ahead, we will use statistical binding in soundness and computational hiding in zero-knowledge. While the former is unconditional, the latter implies reductions to both soundness and zero-knowledge of the underlying candidates. This is the case, given that the first transformation of Lemma 4 simultaneously requires both properties. By restricting our attention to proofs, soundness also becomes unconditional. Hence, we do not have any reduction to soundness within the proof for zero-knowledge.

7.3 Combiner description

We are now ready to define our combiner. The construction is very similar to [GJS19]. Intuitively, the prover secret shares the witness w, runs the MPC protocol Π_{MPC} to generate views of all parties, uses the non-interactive commitment scheme to commit to each of these views (which, again, include the input, randomness, and sent/received messages of the party they relate to) the sent and received messages of all the parties. Then, for each party, it uses a NIZK candidate to prove that the commitments related to that party are computed correctly and the committed values are from an execution of Π_{MPC} for which the party's output is 1. The verifier checks whether all the sub-proofs are accepting, and checks whether the commitment corresponding to outgoing messages are consistent with the commitment corresponding to inbound messages. If all the checks succeed, the verifier outputs 1. Formally, we define our combiner $\mathsf{C}_{\mathsf{MPC}} = (\mathsf{C}_{\mathsf{setup}}, \mathsf{C}_{\mathsf{prove}}, \mathsf{C}_{\mathsf{vrfy}})$ in Figure 7.2.

Figure 7.2: Combiner $\mathsf{C}_{\mathsf{MPC}}$

- $\mathsf{C}_{\mathsf{setup}}(1^l, 1^\lambda)$: Compute $pp_k \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$, for $k \in [n]$. Compute $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. Output $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$.
- $\mathsf{C}_{\mathsf{prove}}(pp, x, w)$:
  - Compute $(w_1, \ldots, w_n) \xleftarrow{\$} \mathtt{share}(w)$.
  - Run MPC protocol Π_{MPC}, where every party P_i ($i \in [n]$) takes w_i as input, and uses uniformly randomly sampled value r_i as the randomness. Obtain $\mathsf{view}_1, \ldots, \mathsf{view}_n$ from the execution of Π_{MPC}.
  - For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]})$.
  - For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$ and $w'_i \leftarrow (\mathsf{view}_i, \{o_{i,j}\}_{j\in[n]}, \{o_{j,i}\}_{j\in[n]})$.
  - For $k \in [n]$, compute $\pi_k = \mathcal{P}_k(pp_k, (pp_{\mathtt{NIC}}, x'_k), w'_k)$. The language associated to the sub-proofs π_i is $\mathcal{L}_{i,\mathsf{MPC}} = \Big\{ (pp_{\mathtt{NIC}}, \{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]}) \;\Big|\; \forall k \in [R] \setminus \{1\}, \exists (\mathsf{view}_i, \{o_{i,j}\}_{j\in[n]}, \{o_{j,i}\}_{j\in[n]}), \{\mathsf{msg}^1_{i\to j}\}_{j\in[n]} = \mathtt{frst\text{-}msg}_i(w_i; r_i) \wedge \{\mathsf{msg}^k_{i\to j}\}_{j\in[n]} = \mathtt{nxt\text{-}msg}^k_i(w_i, \{\mathsf{msg}^l_{j\to i}\}_{j\in[n], l\in[k-1]}; r_i) \wedge 1 = \mathtt{output}_i(w_i, \mathsf{msg}_{i,\mathsf{in}}; r_i) \wedge \forall j \in [n], \mathtt{open}(com_{i,j}, o_{i,j}) = \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]} \wedge \mathtt{open}(com_{j,i}, o_{j,i}) = \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]} \Big\}$.
  - Output $\pi = (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.
- $\mathsf{C}_{\mathsf{vrfy}}(pp, x, \pi)$: Output 1 if all following conditions hold:
  - Compute $\mathcal{V}_k(pp_k, (pp_{\mathtt{NIC}}, x'_k), \pi_k)$ for $k \in [n]$, and all $\mathcal{V}_k$ outputs 1.
  - Parse $\{x'_k\}_{k\in[n]}$ as $\{(\{com_{k,j}\}_{j\in[n]}, \{com_{j,k}\}_{j\in[n]})\}_{k\in[n]}$. For $k \in [n], j \in [n]$, there exists $j \neq k$ s.t. $com_{k,j} = com_{j,k}$.

We remark that, even though all the proofs for the remainder of this section work for $t > n/2$, we rely on the instantiation of Π_{MPC} as in BGW [BGW88]. In [GMO16], the authors claim that such a protocol is $\lfloor \frac{n-1}{3} \rfloor$-private, perfect $\lfloor \frac{n-1}{3} \rfloor$-robust protocol, and use it to instantiate the protocol of [IKOS07].

Theorem 10. *The construction in Figure 7.2 is a correct n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner for any*

NP *language* $\mathcal{L}$*, as per Definition 2, for* $n, t \in \mathsf{poly}(\lambda)$.

Theorem 11. *The construction in Figure 7.2 is a t-out-of-n* N *statistically-sound combiner for any* NP *language* $\mathcal{L}$ *for* $t > \frac{n}{2}$ *and* $n, t \in \mathsf{poly}(\lambda)$.

Theorem 12. *The construction in Figure 7.2 is a t-out-of-n NB ZK combiner for any* NP *language* $\mathcal{L}$*, as per Definition 6, for* $t > \frac{n}{2}$ *and* $n, t \in \mathsf{poly}(\lambda)$.

Theorem 13. *The construction in Figure 7.2 is a t-out-of-n NB PoK combiner for any* NP *language* $\mathcal{L}$ *for* $t > \frac{n}{2}$ *and* $n, t \in \mathsf{poly}(\lambda)$.

The formal proofs of these theorems follow the same structure of the previous combiners, and are deferred, respectively, to Appendix C.8, Appendix C.9, Appendix C.10, and Appendix C.11.

Acknowledgements

Daniele Venturi is member of the Gruppo Nazionale Calcolo Scientifico Istituto Nazionale di Alta Matematica (GNCS-INdAM). His reaserch was supported by project SERICS (PE00000014) and by project PARTHENON (B53D23013000006), under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU, and by project BEAT, funded by Sapienza University of Rome. Lorenzo Magliocco was supported by project PARTHENON (B53D23013000006), under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU, and by project BEAT, funded by Sapienza University of Rome. Michele Ciampi was supported by the Sunday Group, Inc. and by the Input Output Research Hub (IORH) of the University of Edinburgh.

References

[ABJ+19] Prabhanjan Ananth, Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. From FE combiners to secure MPC and back. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 199–228, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 491–520, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany.

[AJS17] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Robust transforming combiners from indistinguishability obfuscation to functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 91–121, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.

[BB06] Dan Boneh and Xavier Boyen. On the impossibility of efficiently combining collision resistant hash functions. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 570–583, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Heidelberg, Germany.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BG24] Nir Bitansky and Nathan Geier. Amplification of non-interactive zero knowledge, revisited. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 361–390, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BKP+24] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron D. Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th Annual ACM Symposium on Theory of Computing*, pages 435–443, Vancouver, BC, Canada, June 24–28, 2024. ACM Press.

[CDFR17] Ignacio Cascudo, Ivan Damgård, Oriol Farràs, and Samuel Ranellucci. Resource-efficient OT combiners with active security. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 461–486, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[Cra96] Ronald Cramer. Modular design of secure yet practical cryptographic protocols. *Ph. D.-thesis, CWI and U. of Amsterdam*, 2, 1996.

[CRS+07] Ran Canetti, Ronald L. Rivest, Madhu Sudan, Luca Trevisan, Salil P. Vadhan, and Hoeteck Wee. Amplifying collision resistance: A complexity-theoretic treatment. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 264–283, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Heidelberg, Germany.

[DFG+23] Yevgeniy Dodis, Niels Ferguson, Eli Goldin, Peter Hall, and Krzysztof Pietrzak. Random oracle combiners: Breaking the concatenation barrier for collision-resistance. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 514–546, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209, Cambridge, MA, USA, February 10–12, 2005. Springer, Berlin, Heidelberg, Germany.

[FHNS16] Marc Fischlin, Amir Herzberg, Hod Bin Noon, and Haya Shulman. Obfuscation combiners. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 521–550, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany.

[FL08] Marc Fischlin and Anja Lehmann. Multi-property preserving combiners for hash functions. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 375–392, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Heidelberg, Germany.

[FLP08] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions revisited. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 655–666, Reykjavik, Iceland, July 7–11, 2008. Springer, Berlin, Heidelberg, Germany.

[FLP14] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions. *Journal of Cryptology*, 27(3):397–428, July 2014.

[FR24] Oriol Farràs and Jordi Ribes-González. One-out-of-q OT combiners. *IEEE Trans. Inf. Theory*, 70(4):2984–2998, 2024.

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Cham, Switzerland.

[GJS19] Vipul Goyal, Aayush Jain, and Amit Sahai. Simultaneous amplification: The case of non-interactive zero-knowledge. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 608–637, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 1069–1083, Austin, TX, USA, August 10–12, 2016. USENIX Association.

[Goo47] Reuben Louis Goodstein. Transfinite ordinals in recursive number theory. *The Journal of Symbolic Logic*, 12(4):123–129, 1947.

[Her05] Amir Herzberg. On tolerant cryptographic constructions. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 172–190, San Francisco, CA, USA, February 14–18, 2005. Springer, Berlin, Heidelberg, Germany.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Heidelberg, Germany.

[HN24] Shuichi Hirahara and Mikito Nanashima. One-way functions and zero knowledge. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *56th Annual ACM Symposium on Theory of Computing*, pages 1731–1738, Vancouver, BC, Canada, June 24–28, 2024. ACM Press.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press.

[JMS20] Aayush Jain, Nathan Manohar, and Amit Sahai. Combiners for functional encryption, unconditionally. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 141–168, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

[LMP24] Yanyi Liu, Noam Mazor, and Rafael Pass. A note on zero-knowledge for NP and one-way functions. *Electron. Colloquium Comput. Complex.*, TR24-095, 2024.

[Mau09] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286, Gammarth, Tunisia, June 21–25, 2009. Springer, Berlin, Heidelberg, Germany.

[Mit12] Arno Mittelbach. Hash combiners for second pre-image resistance, target collision resistance and pre-image resistance have long output. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12: 8th International Conference on Security in Communication Networks*, volume 7485 of *Lecture Notes in Computer Science*, pages 522–539, Amalfi, Italy, September 5–7, 2012. Springer, Berlin, Heidelberg, Germany.

[Mit13] Arno Mittelbach. Cryptophia's short combiner for collision-resistant hash functions. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 136–153, Banff, AB, Canada, June 25–28, 2013. Springer, Berlin, Heidelberg, Germany.

[MP06] Remo Meier and Bartosz Przydatek. On robust combiners for private information retrieval and other primitives. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 555–569, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Heidelberg, Germany.

[MP12] Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 701–718, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Heidelberg, Germany.

[MP14] Bart Mennink and Bart Preneel. Breaking and fixing cryptophia's short combiner. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *CANS*

14: 13th International Conference on Cryptology and Network Security, volume 8813 of *Lecture Notes in Computer Science*, pages 50–63, Heraklion, Crete, Greece, October 22–24, 2014. Springer, Cham, Switzerland.

[MPW07] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. Robuster combiners for oblivious transfer. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 404–418, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Berlin, Heidelberg, Germany.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991.

[Pie07] Krzysztof Pietrzak. Non-trivial black-box combiners for collision-resistant hash-functions don't exist. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 23–33, Barcelona, Spain, May 20–24, 2007. Springer, Berlin, Heidelberg, Germany.

[Pie08] Krzysztof Pietrzak. Compression from collisions, or why CRHF combiners have a long output. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 413–432, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Heidelberg, Germany.

[PR20] Bertram Poettering and Paul Rösler. Combiners for AEAD. *IACR Transactions on Symmetric Cryptology*, 2020(1):121–143, 2020.

[Ps05] Rafael Pass and Abhi shelat. Unconditional characterizations of non-interactive zero-knowledge. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 118–134, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Heidelberg, Germany.

[Som06] Christian Sommer. *Robust Combiners for Cryptographic Primitives*. Master thesis, ETH Zurich, 2006.

A Deferred definitions

In this section of the appendix, we report definitions that are already used in the literature.

Definition 11 (Pseudorandom function)**.** *A pseudorandom function (PRF) is a family of functions $F_s : \{0,1\}^k \to \{0,1\}^l$, indexed by a key $s \in \{0,1\}^\lambda$, for which the following holds:*

- *$F_s(x)$ is efficiently computable, given s and x;*
- *For all PPT adversaries $\mathcal{A}$, the following holds:*

$$\Pr\left[\mathcal{A}^{F_s(\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{R(\cdot)}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

where $s \xleftarrow{\$} \{0,1\}^\lambda$ and $R : \{0,1\}^k \to \{0,1\}^l$, with R being a random function.

Definition 12 (Non-interactive string commitment)**.** *A non-interactive string-commitment scheme $com = (\mathsf{S}, \mathsf{R})$ for message space $\mathcal{M}$ is composed of an efficient sender S and an efficient receiver R for which the following conditions hold:*

- ***Completeness.***

$$\Pr\left[\mathtt{open}(pp, com, o) = m \mid pp \xleftarrow{\$} \mathtt{setup}(1^\lambda); (com, o) \leftarrow \mathtt{commit}(pp, m)\right] \geq 1 - \mathsf{negl}(\lambda)$$

- ***Computational Hiding.*** *Let $pp \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. Then, for all $m_0 \neq m_1$, with $m_0, m_1 \in \mathcal{M}$:*

$$\mathtt{commit}(pp, m_0) \approx_c \mathtt{commit}(pp, m_1)$$

- ***Computational Binding.*** *For all PPT algorithms S' and all $m_0 \neq m_1$, with $m_0, m_1 \in \mathcal{M}$:*

$$\Pr\left[\mathtt{open}(pp, com, o_0) = m_0 \ \wedge \ \mathtt{open}(pp, com, o_1) = m_1 \;\middle|\; \begin{array}{l} pp \xleftarrow{\$} \mathtt{setup}(1^\lambda) \\ (com, o_0, o_1) \leftarrow \mathsf{S}'(pp) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

If the properties above hold against unbounded adversaries, we refer to statistical hiding and statistical binding, respectively.

Secret sharing. We provide a generic definition of a t-out-of-n secret sharing scheme (that we use in Section 7).

Definition 13 (t-out-of-n secret sharing scheme)**.** *A t-out-of-n secret sharing scheme is a pair of algorithms (*`share`*,* `reconstruct`*) that behaves as follows:*

- `share` *is a randomized algorithm that, on input a secret m, outputs a set of n shares $(s_1, \ldots, s_n)$.*
- `reconstruct` *is a deterministic algorithm that, on input t shares $(s_1, \ldots, s_t)$, outputs the secret m.*

The scheme satisfies the correctness *property if, $\forall m$, $\forall S = \{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ of size t, it holds that:*

$$\Pr_{\mathtt{share}(m) \to (s_1, \ldots, s_n)}[\mathtt{reconstruct}(s_{i_1}, \ldots, s_{i_t}) = m] = 1$$

In addition, the scheme satisfies perfect security *if, $\forall m, m'$, $\forall S \subseteq \{1, \ldots, n\}$, where $|S| < t$, the following holds:*

$$\{(s_i : i \in S) \mid (s_1, \ldots, s_n) \leftarrow \mathtt{share}(m)\} \approx_p \{(s'_i : i \in S) \mid (s'_1, \ldots, s'_n) \leftarrow \mathtt{share}(m')\}$$

Additionally, we specify properties we require from the secret sharing scheme in Section 5 (which are similar to some of the basic properties of the Shamir's secret sharing).

Definition 14 (Correctness)**.** *We say the scheme satisfy the correctness property if,* $\forall m$, $\forall S = \{i_1, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ *of size* t*, it holds that:*

$$\Pr_{\mathtt{share}_{\mathsf{G}}(m)\to(s_1,\ldots,s_n)}[\mathtt{reconstruct}_{\mathsf{G}}(s_{i_1}, \ldots, s_{i_t}) = m] = 1$$

Additionally, we require that for any set $S = \{i_1, \ldots, i_t\} \subseteq [n]$*, the following holds:* $\forall k \in [n] \setminus S$*,*

$$\Pr_{\mathtt{share}_{\mathsf{G}}(m)\to(s_1,\ldots,s_n)}[\overline{\mathtt{reconstruct}}_{\mathsf{G}}((s_{i_1}, \ldots, s_{i_t}), k) = s_k] = 1$$

Definition 15 (Perfect security)**.** *We say that* $(\mathtt{share}_{\mathsf{G}}, \mathtt{reconstruct}_{\mathsf{G}})$ *satisfies perfect security if,* $\forall m, m'$, $\forall S \subseteq \{1, \ldots, n\}$*, where* $|S| < t$*, the following holds:*

$$\{(s_i : i \in S) \mid (s_1, \ldots, s_n) \leftarrow \mathtt{share}_{\mathsf{G}}(m)\} \approx_p \{(s'_i : i \in S) \mid (s'_1, \ldots, s'_n) \leftarrow \mathtt{share}_{\mathsf{G}}(m')\}$$

Standard security properties of non-interactive proof systems. Next, we report standard counterparts of the security definitions we consider for our combiners (*i.e.*, soundness, zero-knowledge, and argument-of-knowledge), providing both the black-box and the non-black-box flavours when applicable.

Definition 16 (Soundness)**.** *Let* $\Pi \in \mathcal{F}_{\mathsf{NI}}$*. We say* Π *is sound if, for all PPT algorithms* $\mathcal{A}$*, the following holds:*

$$\Pr\left[\mathcal{V}(pp, x, \pi) = 1 \wedge x \notin \mathcal{L} \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^{\lambda}); (x, \pi) \xleftarrow{\$} \mathcal{A}(pp)\right] < \mathsf{negl}(\lambda).$$

Definition 17 (Black-box Non-Interactive Zero Knowledge)**.** *Let* $\Pi \in \mathcal{F}_{\mathsf{NI}}$*. Let* $\mathtt{REAL} = \{(pp, x, w, \pi) \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^{\lambda});\ (x, w) \xleftarrow{\$} \mathcal{A}(pp);\ \pi \xleftarrow{\$} \mathcal{P}(pp, x, w)\}_{\lambda\in\mathbb{N}}$ *and* $\mathtt{IDEAL}_{\mathcal{S}} = \{(pp, x, w, \pi) \mid (pp, \tau) \xleftarrow{\$} \mathcal{S}_0(1^{|x|}, 1^{\lambda});\ (x, w) \xleftarrow{\$} \mathcal{A}(pp);\ \pi \xleftarrow{\$} \mathcal{S}_1(pp, \tau, x)\}_{\lambda\in\mathbb{N}}$*. We say that* Π *satisfies black-box zero-knowledge if there exists a PPT algorithm* $\mathcal{S}$ *such that, for all PPT algorithms* $\mathcal{A}$*, the following holds:*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow \mathcal{A}(pp, x, w, \pi) \end{array}; (pp, x, w, \pi) \xleftarrow{\$} \begin{cases} \mathtt{REAL} & \text{if } b = 0 \\ \mathtt{IDEAL}_{\mathcal{S}} & \text{if } b = 1 \end{cases}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Definition 18 (Non-Black-Box Non-Interactive Zero-Knowledge)**.** *Let* $\Pi \in \mathcal{F}_{\mathsf{NI}}$*. Let* $\mathtt{REAL} = \{(pp, x, w, \pi) \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^{\lambda});\ (x, w) \xleftarrow{\$} \mathcal{A}(pp);\ \pi \xleftarrow{\$} \mathcal{P}(pp, x, w)\}_{\lambda\in\mathbb{N}}$ *and* $\mathtt{IDEAL}_{\mathcal{S}} = \{(pp, x, w, \pi) \mid (pp, \tau) \xleftarrow{\$} \mathcal{S}_0(1^{|x|}, 1^{\lambda});\ (x, w) \xleftarrow{\$} \mathcal{A}(pp);\ \pi \xleftarrow{\$} \mathcal{S}_1(pp, \tau, x)\}_{\lambda\in\mathbb{N}}$*. We say that* Π *satisfies non-black-box zero-knowledge if, for all PPT algorithms* $\mathcal{A}$*, there exists a PPT algorithm* $\mathcal{S}$ *for which the following holds:*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} b \xleftarrow{\$} \{0,1\} \\ b' \leftarrow \mathcal{A}(pp, x, w, \pi) \end{array}; (pp, x, w, \pi) \xleftarrow{\$} \begin{cases} \mathtt{REAL} & \text{if } b = 0 \\ \mathtt{IDEAL}_{\mathcal{S}} & \text{if } b = 1 \end{cases}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Definition 19 (Black-Box Argument-of-Knowledge)**.** *Let* $\Pi \in \mathcal{F}_{\mathsf{NI}}$*. We say that* Π *satisfies black-box argument-of-knowledge if there exists a PPT algorithm* $\mathcal{E}$ *such that, for all PPT algorithms* $\mathcal{A}$*, the following holds:*

$$\Pr\left[\mathcal{A}(pp) = 1 \mid pp \xleftarrow{\$} \mathtt{setup}(1^{|x|}, 1^{\lambda})\right] \approx_c \Pr\left[\mathcal{A}(pp) = 1 \mid (pp, \xi) \xleftarrow{\$} \mathcal{E}_0(1^{|x|}, 1^{\lambda})\right],$$

and

$$\Pr\left[(x,w)\in\mathscr{R}_{\mathcal{L}},\mathcal{V}(pp,x,\pi)=1 \;\middle|\; \begin{array}{l}(pp,\xi)\xleftarrow{\$}\mathcal{E}_0(1^n,1^\lambda);(x,\pi)\leftarrow\mathcal{A}(pp)\\ w\leftarrow\mathcal{E}_1(pp,\xi,x,\pi)\end{array}\right]\geq 1-\mathsf{negl}(\lambda).$$

Definition 20 (Non-Black-Box Argument-of-Knowledge)**.** *Let $\Pi\in\mathcal{F}_{\mathsf{NI}}$. We say that Π satisfies non-black-box argument-of-knowledge if, for all PPT algorithms $\mathcal{A}$, there exists a PPT algorithm $\mathcal{E}$ for which the following holds:*

$$\Pr\left[\mathcal{A}(pp)=1 \mid pp\xleftarrow{\$}\mathtt{setup}(1^{|x|},1^\lambda)\right]\approx_c \Pr\left[\mathcal{A}(pp)=1 \mid (pp,\xi)\xleftarrow{\$}\mathcal{E}_0(1^{|x|},1^\lambda)\right],$$

and

$$\Pr\left[(x,w)\in\mathscr{R}_{\mathcal{L}},\mathcal{V}(pp,x,\pi)=1 \;\middle|\; \begin{array}{l}(pp,\xi)\xleftarrow{\$}\mathcal{E}_0(1^n,1^\lambda);(x,\pi)\leftarrow\mathcal{A}(pp)\\ w\leftarrow\mathcal{E}_1(pp,\xi,x,\pi)\end{array}\right]\geq 1-\mathsf{negl}(\lambda).$$

MPC properties. In what follows, we report MPC properties following from [IKOS07].

Definition 21 (Correctness)**.** *Let $f_x(w_1,\ldots,w_n):(\{0,1\}^*)^n\to(\{0,1\}^*)^n$ with a hardcoded value $x\in\{0,1\}^*$ be an n-party function. A protocol Π realizes the function f_x with* perfect correctness *if for all inputs $w_1,\ldots,w_n$, the probability that the output of some player is different from the output of f_x is 0, where the probability is over the independent choices of the randomness $r_1,\ldots,r_n$.*

Definition 22 ($\mathcal{K}$-Privacy)**.** *Let $f_x(w_1,\ldots,w_n):(\{0,1\}^*)^n\to(\{0,1\}^*)^n$ with a hardcoded value $x\in\{0,1\}^*$ be an n-party function. We denote the joint view $\mathsf{view}_{\mathcal{I}}(w_1,\ldots,w_n)$ as the views of parties P_i ($i\in\mathcal{I}$), corresponding to the execution where every P_i's input is w_i. We denote $f_{x,\mathcal{I}}(w_1,\ldots,w_n)$ the output of function $f_x(w_1,\ldots,w_n)$ received by the parties whose indices are in the set $\mathcal{I}$. A protocol Π realizes the function f_x with* perfect $\mathcal{K}$-Privacy *if there is a PPT simulator $\mathcal{S}$ s.t. for all inputs $(w_1,\ldots,w_n)$, and every set of corrupted players $\mathcal{I}\subseteq[n]$ with at most size $\mathcal{K}$, the joint view $\mathsf{view}_{\mathcal{I}}(w_1,\ldots,w_n)$ of parties in $\mathcal{I}$ is identically distributed to $\mathcal{S}(\mathcal{I},x,\{w_k\}_{k\in[\mathcal{I}]},f_{x,\mathcal{I}}(w_1,\ldots,w_n))$. We note that when we say "corrupted", we mean semi-honest,* i.e.*, parties will follow the protocol specification.*

Definition 23 ($\mathcal{K}$-Robustness)**.** *Let $f_x(w_1,\ldots,w_n):(\{0,1\}^*)^n\to(\{0,1\}^*)^n$ with a hardcoded value $x\in\{0,1\}^*$ be an n-party function. A protocol Π realizes the function f_x with* perfect $\mathcal{K}$-Robustness *if it is perfectly correct (Definition 21) in the presence of a semi-honest adversary, and moreover for any computational unbounded malicious adversary corrupting a set $\mathcal{I}$ with at most size $\mathcal{K}$, and for any inputs $(w_1,\ldots,w_n)$, if there is no $(w'_1,\ldots,w'_n)$ s.t. $f_x(w'_1,\ldots,w'_n)=1$, then the probability that some honest parties output 1 in an execution of Π realizing f_x is 0.*

B Concrete instantiations of homomorphic languages and $\overline{\mathtt{reconstruct}}_{\mathsf{H}}$

In this section, we report concrete instantiations of $\overline{\mathtt{reconstruct}}_{\mathsf{H}}$ for some of the homomorphic languages presented in [Mau09].

We also note that, in the following examples, the witness is either in $\mathbb{Z}_q$ or $\mathbb{Z}_q\times\mathbb{Z}_q$. When the witness is in $\mathbb{Z}_q$, we know $\mathbb{Z}_q$ is actually a finite field, and the secret sharing scheme $(\mathtt{share}_{\mathsf{G}},\mathtt{reconstruct}_{\mathsf{G}})$ in Section 5 satisfies correctness (as per Definition 14) and perfect security (as per Definition 15). Regarding $\mathbb{Z}_q\times\mathbb{Z}_q$, if we consider the addition operator as component-wise addition, and the multiplication operator as component-wise multiplication, then $\mathbb{Z}_q\times\mathbb{Z}_q$ is a finite field, and the correctness and perfect security of $(\mathtt{share}_{\mathsf{G}},\mathtt{reconstruct}_{\mathsf{G}})$ also holds. Intuitively, for this case, we secret share each component of the witness, and perform the reconstruction component-wise.

DLOG. In DLOG, $(\mathsf{G}, \star) = (\mathbb{Z}_q, +)$, and $(\mathsf{H}, \otimes) = (\mathsf{H}, \cdot)$, with H being a group of prime order q. The homomorphic group operation is defined as:

$$\mathsf{G} \to \mathsf{H} : w \to f(w) = g^w$$

Therefore, $\{2\star\} = \cdot$, and $\{2\otimes\} = \exp$. More specifically:

- $f(w_1) \cdot f(w_2) = g^{w_1} \cdot g^{w_2} = g^{w_1+w_2} = f(w_1 + w_2)$
- $f(w) \exp v = g^w \exp v = \underbrace{g^w \cdot g^w \cdot \cdots \cdot g^w}_{v \text{ copies of } g^w} = g^{w \cdot v} = f(w \cdot v)$

For completeness, we also instantiate the generic $\mathtt{share}_{\mathsf{G}}$ of Section 5 with the operators defined above:

- $\mathtt{share}_{\mathsf{G}}(w)$: $a_1, \ldots, a_{t-1} \xleftarrow{\$} \mathtt{Gen}_{\mathsf{G}}(1^\lambda)$. Output $\{w_i \leftarrow w + \sum_{j=1}^{t-1}(a_j \cdot i^j)\}_{i \in [n]}$
- $\overline{\mathtt{reconstruct}}_{\mathsf{G}}(\{s_i\}_{i \in I}, l)$: Output $w_l \leftarrow \sum_{j \in I}(s_i \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k}))$

We expand $\overline{\mathtt{reconstruct}}_{\mathsf{H}}$ in terms of $\overline{\mathtt{reconstruct}}_{\mathsf{G}}$, as in Section 5.

$$\begin{aligned}
x'_l &= \bigotimes_{j \in I}(x_j\{2\otimes\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) \\
&= \prod_{j \in I}(g^{w_j} \exp(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) \\
&= \prod_{j \in I}(g^{w_j \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})}) \\
&= g^{\sum_{j \in I} w_j \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})} \\
&= g^{w_l} = f(\overline{\mathtt{reconstruct}}_{\mathsf{G}}(\{w_j\}_{j \in I}, l)) = f(w_l) = x_l
\end{aligned}$$

DH tuples. In DH, $f(w) = (g^w, h^w)$, $(\mathsf{G}, \star) = (\mathbb{Z}_q, +)$, and $(\mathsf{H}, \otimes) = (\mathsf{H} \times \mathsf{H}, \odot)$, with H being a group of prime order q. The homomorphic group operation is defined as:

$$\mathsf{G} \to \mathsf{H} \times \mathsf{H} : w \to f(w) = (g^w, h^w)$$

The group operation of H $\odot$ is the Hadamard product (*i.e.*, the component-wise product), with its matching hyperoperation $\{2\odot\}$ being the component-wise exponentiation. More specifically:

- $f(w_1) \odot f(w_2) = (g^{w_1} \cdot g^{w_2}, h^{w_1} \cdot h^{w_2}) = (g^{w_1+w_2}, h^{w_1+w_2}) = f(w_1 + w_2)$
- $f(w)\{2\odot\}v = \underbrace{f(w) \odot f(w) \odot \ldots \odot f(w)}_{v \text{ copies of } f(w_i)} = (g^{w \cdot v}, h^{w \cdot v}) = f(w \cdot v)$

The algorithms $\mathtt{share}_{\mathsf{G}}$, $\mathtt{reconstruct}_{\mathsf{G}}$ are exactly the same as in the previous example. We expand $\overline{\mathtt{reconstruct}}_{\mathsf{H}}$ in terms of $\overline{\mathtt{reconstruct}}_{\mathsf{G}}$, as in Section 5.

$$\begin{aligned}
x'_l &= \bigotimes_{j \in I}(x_j\{2\otimes\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) \\
&= \bigodot_{j \in I}((g^{w_j}, h^{w_j})\{2\odot\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) \\
&= \bigodot_{j \in I}(g^{w_j \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})}, h^{w_j \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})}) \\
&= (g^{\sum_{j \in I} w_j \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})}, h^{\sum_{j \in I} w_j \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})}) \\
&= (g^{w_l}, h^{w_l}) = f(\overline{\mathtt{reconstruct}}_{\mathsf{G}}(\{w_j\}_{j \in I}, l)) = f(w_l) = x_l
\end{aligned}$$

Pedersen commitments. In Pedersen commitments, $f(w_1, w_2) = h_1^{w_1} h_2^{w_2}$, $(\mathsf{G}, \star) = (\mathbb{Z}_q \times \mathbb{Z}_q, \star)$, with $\star$ being the component-wise addition, with its matching hyperoperation $\{2\star\}$ being the component-wise multiplication, and $(\mathsf{H}, \otimes) = (\mathsf{H}, \cdot)$, with H being a group of prime order q. The homomorphic group operation is defined as:

$$\mathbb{Z}_q \times \mathbb{Z}_q \to \mathsf{H} : (w_1, w_2) \to f(w_1, w_2) = h_1^{w_1} h_2^{w_2}$$

We expand the homomorphic operations as follows:

- $f(a) \otimes f(b) = (h_1^{a_1} h_2^{a_2}) \cdot (h_1^{b_1} h_2^{b_2}) = h_1^{a_1+b_1} h_2^{a_2+b_2} = f(a \star b)$
- $f(a)\{2\otimes\}v = \underbrace{(h_1^{a_1} h_2^{a_2}) \cdot (h_1^{a_1} h_2^{a_2}) \ldots (h_1^{a_1} h_2^{a_2})}_{v \text{ copies of } a = (a_1, a_2)} = f(h_1^{va_1} h_2^{va_2}) = f(a\{2\star\}v)$

For completeness, we also report $\mathtt{share}_{\mathsf{G}}$ instantiated with the operators defined above. Intuitively, we secret share each component of the witness on a different polynomial, and perform the reconstruction component-wise:

- $\mathtt{share}_{\mathsf{G}}(w = (w_1, w_2))$: $(a_{1,1}, a_{2,1}) \ldots, (a_{1,t-1}, a_{2,t-1}) \xleftarrow{\$} \mathtt{Gen}_{\mathsf{G}}(1^\lambda)$. Output $\{(w_{1,i}, w_{2,i}) \leftarrow (w_1 + \sum_{j=1}^{t-1}(a_{1,j} \cdot i^j), w_2 + \sum_{j=1}^{t-1}(a_{2,j} \cdot i^j))\}_{i \in [n]}$
- $\overline{\mathtt{reconstruct}}_{\mathsf{G}}(\{(s_{1,i}, s_{2,i})\}_{i \in I}, l)$:
  Output $w = (w_1, w_2) \leftarrow (\sum_{j \in I}(s_{1,i} \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})), \sum_{j \in I}(s_{2,i} \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})))$

We expand $\overline{\mathtt{reconstruct}}_{\mathsf{H}}$ in terms of $\overline{\mathtt{reconstruct}}_{\mathsf{G}}$, as in Section 5.

$$\begin{aligned}
x'_l &= \bigotimes_{j \in I}(x_j\{2\otimes\}(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) \\
&= \prod_{j \in I}(h_1^{a_{1,j}} h_2^{a_{2,j}}) \exp(\prod_{k \in I, k \neq j} \frac{l-k}{j-k})) \\
&= \prod_{j \in I}(h_1^{a_{1,j} \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})} h_2^{a_{2,j} \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})} \\
&= h_1^{\sum_{j \in I} a_{1,j} \cdot (\prod_{k \in I, k \neq j} \frac{l-k}{j-k})} h_2^{\sum_{j \in I} a_{2,j} \cdot (\prod_{k \in S, k \neq j} \frac{l-k}{j-k})} \\
&= f(\overline{\mathtt{reconstruct}}_{\mathsf{G}}(\{(a_{1,j}, a_{2,j})\}_{j \in I}), l) = f(a_l) = x_l
\end{aligned}$$

C Security proofs

In this section we report the proofs of security that were deferred throughout the composition.

C.1 Combiner for homomorphic languages: correctness

Proof (Theorem 3). In order for the construction to be an n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner, the following has to hold:

$$\Pr\left[\mathsf{C}_{\mathsf{vrfy}}(pp, x, \mathsf{C}_{\mathsf{prove}}(pp, x, w)) = 1 \mid pp \xleftarrow{\$} \mathsf{C}_{\mathsf{setup}}(1^{|x|}, 1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)$$

Intuitively, this happens if both (i) all sub-proofs are accepting, and (ii) all reconstructed values are consistent with the same polynomial, which evaluates to x in 0. We explore the two events separately.

Event 1. Let $\mathsf{E}_{\mathsf{corr,k}}$ as the event that the k-th sub-proof is accepting. We denote $\mathsf{E}_{\mathsf{corr}}$ as the event that all sub-proofs are accepting, *i.e.*:

$$\Pr[\mathsf{E}_{\mathsf{corr}}] = \prod_{k=1}^{n} \Pr[\mathsf{E}_{\mathsf{corr,k}}]$$

For $t \in \mathsf{poly}(\lambda)$, the aforementioned quantity is overwhelming. By Bernoulli's inequality (*i.e.*, $(1+x)^n \geq 1+nx$), the following holds:

$$\begin{aligned}
\Pr[\mathsf{E}_{\mathsf{corr}}] &\geq (\min_k \{\Pr[\mathsf{E}_{\mathsf{corr,k}}]\})^n \\
&\geq (1 - \mathsf{negl}(\lambda))^{\mathsf{poly}(\lambda)} \\
&\geq 1 - \mathsf{negl}(\lambda)\mathsf{poly}(\lambda) \\
&\geq 1 - \mathsf{negl}(\lambda)
\end{aligned}$$

Event 2. The verifier computes $x'_l = \overline{\texttt{reconstruct}}_{\mathsf{H}}(\{x_j\}_{j\in[t]}, l)$, for $l \in [n]\backslash[t]\cup\{0\}$. Following Section 5, this is equivalent to $f(\overline{\texttt{reconstruct}}_{\mathsf{G}}(\{s_j\}_{j\in[t]}, l))$. By correctness of Shamir's secret sharing (as per Definition 14), the verifier successfully reconstructs the sub-statement $x'_l = f(w_l) = x_l$ with probability 1. □

C.2 Combiner for homomorphic languages: soundness

Proof (Theorem 4). Assume by contradiction that the construction C_{HL} is not sound. Then, there exists a PPT algorithm $\mathcal{A}$ for which $(\mathsf{C}_{\mathsf{HL}}, \mathcal{A}) \in \mathcal{R}_{\mathsf{Sound}}$ as per Definition 4. This means that $\mathcal{A}$ outputs an accepting proof π for the combiner for a statement $x \notin \mathcal{L}_{\mathsf{HL}}$ with non-negligible probability. The verifier parses (x, π) as $\{x_k, \pi_k\}_{k\in[n]}$, and behaves according to the protocol description. In particular, the verifier checks that (i) each sub-proof π_i verifies against the verifier of candidate i $\mathcal{V}_i$, and (ii) each sub-statement x_i is an evaluation of the same polynomial of degree $t-1$ in i.

In order to prove soundness for our combiner, it suffices to show that any accepting proof for an $x \notin \mathcal{L}_{\mathsf{HL}}$ must yield a set of indices $I = \{i_1, \ldots, i_{n-t+1}\}$ such that $(\Pi_{i_k}, \mathsf{Red}^{\mathcal{A},\Pi_{i_k}}) \in \mathcal{R}_{\mathsf{Sound}}$ for all $i_k \in I$. We prove that this fact holds in Lemma 5, and specify the behaviour of $\mathsf{Red}^{\mathcal{A},\Pi_{i_k}}$ next.

Figure C.1: Reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i_k}}$

- Receive public parameters pp'_{i_k} from the challenger $\mathcal{C}$.
- Compute $pp \xleftarrow{\$} \mathsf{C}_{\mathsf{setup}}(1^l, 1^\lambda)$. Parse pp as $\{pp_j\}_{j\in[n]}$. Replace pp_{i_k} with pp'_{i_k}.
- Forward pp to $\mathcal{A}$.
- Upon receiving (x, π) from $\mathcal{A}$, Parse π as $\{\pi_k, x_k\}_{k\in[n]}$.
- Forward (x_{i_k}, π_{i_k}) to $\mathcal{C}$.

□

Lemma 5. *For $t > \frac{n}{2}$, any accepting proof π for the combiner related to a statement $x \notin \mathcal{L}_{\mathsf{HL}}$ must include at least $n-t+1$ distinct indices $I = \{i_1, \ldots, i_{n-t+1}\}$ for which $(\Pi_{i_k}, \mathsf{Red}^{\mathcal{A},\Pi_{i_k}}) \in \mathcal{R}_{\mathsf{Sound}}$ for all $i_k \in I$.*

Proof. Assume by contradiction that the lemma statement does not hold. Then, there exists an accepting proof π for the combiner related to a statement $x \notin \mathcal{L}_{\mathsf{HL}}$ that includes at most $n-t$

distinct indices $I = \{i_1, \ldots, i_{n-t}\}$ for which π_{i_k} is an accepting proof for a statement $x_{i_k} \notin \mathcal{L}_{\mathsf{HL}}$ for candidate Π_{i_k}, with $i_k \in I$.

Since the verifier of the combiner accepts proof π, all sub-proofs are accepting. In particular, this means that there exists a subset I' containing sub-proofs $\{\pi_{i_k}\}_{i_k \notin I}$ that do not yield reductions to soundness. Given that these proofs are accepting, they relate to sub-statements that are in the language. We observe that, by assumption, the cardinality of set I' is $n-(n-t) = t$. In what follows, we denote the set of shares identified by I' as $S = \{x_i\}_{i \in I'}$.

In order for the verifier of the combiner to accept proof π, it should be the case that $x = \overline{\texttt{reconstruct}}_{\mathsf{H}}(\{x_k\}_{k \in I'}, 0)$, *i.e.*:

$$x = \bigotimes_{j \in I'} (x_j \{2\otimes\} (\prod_{k \in I', k \neq j} \frac{k}{k-j}))$$

Since all the sub-statements in S are in the language (*i.e.*, $\forall j \in I', x_j = f(w_j)$), the following holds:

$$x = \bigotimes_{j \in I'} (f(w_j) \{2\otimes\} (\prod_{k \in I', k \neq j} \frac{k}{k-j}))$$

Then, we apply the homomorphism of f by invoking Fact 1 and Fact 2:

$$\begin{aligned} x &= \bigotimes_{j \in I'} (f(w_j) \{2\otimes\} (\prod_{k \in I', k \neq j} \frac{k}{k-j})) \\ &= f(\bigstar_{j \in I'} (w_j \{2\star\} (\prod_{k \in I', k \neq j} \frac{k}{k-j}))) \\ &= f(\overline{\texttt{reconstruct}}_{\mathsf{G}}(\{w_j\}_{j \in I'}, 0)) \\ &= f(w) \end{aligned}$$

Note that, by the assumption of $t > \frac{n}{2}$, the reconstruction of the secret out of the n shares is uniquely determined by any set of size t. Hence, set S uniquely determines the secret $f(w)$. This is a contradiction to the assumption that $x \notin \mathcal{L}_{\mathsf{HL}}$, as $x = f(w)$. Therefore, any accepting proof π for the combiner for a statement $x \notin \mathcal{L}_{\mathsf{HL}}$ implies the existence of at least $n - t + 1$ reductions to soundness of the underlying candidates.

□

C.3 Combiner for homomorphic languages: zero-knowledge

Proof (Theorem 5). Following Section 5.2, we construct a simulator $\mathcal{S}^T$ and prove in Lemma 6 that either its output is indistinguishable from the output of the real prover, or there exists at least one reduction to zero-knowledge of one of the candidates whose indices are in T. With that, we can use exactly the same the combinatorics argument used to prove in Lemma 2. □

Lemma 6. *There exists a simulator $\mathcal{S}^T$ for which either $\{\texttt{REAL}\} \approx_c \{\texttt{IDEAL}_{\mathcal{S}^T}\}$, or there exists a reduction to zero-knowledge of at least one of the candidates $\{\Pi_k\}_{k \in T}$.*

Proof. Let $T = \{i_1, \ldots, i_t\}$ be a set of indices sorted in lexicographic order identifying candidates $\Pi_{i_1}, \ldots, \Pi_{i_t}$. Moreover, we recall that we denote T_j as the set containing the first j indices of T, *i.e.*, $T_j = \{i_1, \ldots, i_j\}$. $\mathcal{S}^T = (\mathcal{S}^{T,0}, \mathcal{S}^{T,1})$ is constructed as follows:

Figure C.2: Simulator $\mathcal{S}^T$

$\mathcal{S}^{T,0}(1^l, 1^\lambda)$:

- For $l \in T$, compute $(pp_l, \tau_l) \xleftarrow{\$} \mathcal{S}_l^0(1^l, 1^\lambda)$.
- For $i \in [n] \setminus T$, compute $pp_i \leftarrow \mathtt{setup}_i(1^l, 1^\lambda)$.
- Compute $pp \leftarrow \{pp_k\}_{k\in[n]}$, $\tau \leftarrow \{\tau_k\}_{k\in T}$.
- Return (pp, τ).

$\mathcal{S}^{T,1}(pp, \tau, x)$:

- Sample $\{r_i\}_{i\in[n]\setminus T\cup T_{2t-n-1}} \xleftarrow{\$} \mathtt{Gen}_\mathsf{G}(1^\lambda)$.
- For $i \in [n] \setminus T \cup T_{2t-n-1}$, compute $x'_i \leftarrow f(r_i)$. Let $x'_0 \leftarrow x$.
- For $l \in T \setminus T_{2t-n-1}$, compute $x'_l \leftarrow \overline{\mathtt{reconstruct}}_\mathsf{H}(\{x'_j\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)$.
- For $i \in [n] \setminus T$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, x'_i, r_i)$. For $l \in T$, compute $\pi_l \leftarrow \mathcal{S}_l^1(pp_l, \tau_l, x'_l)$.
- Return $(\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.

The proof of this lemma goes through $t+3$ hybrid experiments, where hybrid $\mathcal{H}_0$ is the same as the real world, and hybrid $\mathcal{H}_{t+2}$ is the same as the ideal world. We denote the output of the adversary in hybrid $\mathcal{H}_i$ as $\mathsf{out}^{\mathcal{H}_i}$, for $i \in \{0, 1, \ldots, t+2\}$. Formally we prove that, for $j \in \{1, \ldots, t+2\}$ and for any PPT algorithm $\mathcal{A}$, the following holds:

$$\left|\Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_{j-1}}) = 1] - \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_j}) = 1]\right| \leq \mathsf{negl}(\lambda).$$

We define the hybrids as follows:

Figure C.3: Hybrid experiments $\mathcal{H}_0$, $\mathcal{H}_j$, $\mathcal{H}_{t+1}$, $\mathcal{H}_{t+2}$, for $1 \leq j \leq t$

$\mathcal{H}_0$, $\boxed{\mathcal{H}_j}$, $\boxed{\mathcal{H}_{t+1}}$, $\boxed{\mathcal{H}_{t+2}}$

1. $pp_k \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$ for $k \in [n]$. $pp \leftarrow \{pp_k\}_{k\in[n]}$.

1. $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T_j$. $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$. $pp \leftarrow \{pp_k\}_{k\in[n]}$.

2. Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$

3. Compute $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}_\mathsf{G}(w)$.
   For $i \in [n]$, compute $x_i \leftarrow f(w_i)$.

3. Sample $\{r_i\}_{i\in[n]\setminus T\cup T_{2t-n-1}} \xleftarrow{\$} \mathtt{Gen}_\mathsf{G}(1^\lambda)$.
   For $i \in [n] \setminus T \cup T_{2t-n+1}$, compute $x'_i \leftarrow f(r_i)$. Let $r_0 \leftarrow w$.
   For $l \in T \setminus T_{2t-n-1}$, compute $w_l \leftarrow \overline{\mathtt{reconstruct}}_\mathsf{G}(\{r_i\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)$. Compute $x_l \leftarrow f(w_l)$.

3. Sample $\{r_i\}_{i\in[n]\setminus T\cup T_{2t-n-1}} \xleftarrow{\$} \mathtt{Gen}_\mathsf{G}(1^\lambda)$.
   For $i \in [n] \setminus T \cup T_{2t-n-1}$, compute $x'_i \leftarrow f(r_i)$. Let $x'_0 \leftarrow x$.
   For $l \in T \setminus T_{2t-n-1}$, compute $x'_l \leftarrow \overline{\mathtt{reconstruct}}_\mathsf{H}(\{x'_j\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)$.

4. $\pi_k \leftarrow \mathcal{P}_k(pp_k, x_k, w_k)$ for $k \in [n]$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x_k\}_{k\in[n]})$.

4. $\pi_k \leftarrow \mathcal{S}_k^1(pp_k, \tau_k, x_k)$ for $k \in T_j$. $\pi_i \leftarrow \mathcal{P}_i(pp_i, x_i, w_i)$ for $i \in [n] \setminus T_j$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x_k\}_{k\in[n]})$.

4. $\pi_k \leftarrow \mathcal{S}_k^1(pp_k, \tau_k, x_k)$ for $k \in T$. $\pi_i \leftarrow \mathcal{P}_i(pp_i, x'_i, r_i)$ for $i \in [n] \setminus T$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x_k\}_{k\in T\setminus T_{2t-n-1}} || \{x'_k\}_{k\in[n]\setminus T\cup T_{2t-n-1}})$.

4. $\pi_k \leftarrow \mathcal{S}_k^1(pp_k, \tau_k, x'_k)$ for $k \in T$. $\pi_i \leftarrow \mathcal{P}_i(pp_i, x'_i, r_i)$ for $i \in [n] \setminus T$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.

5. Run $\mathcal{A}(pp, x, \pi)$.

Intuitively, in the first t hybrids, the real proofs generated by the candidates in T are gradually replaced with simulated proofs generated by their respective simulator. Note that, in principle, it could be the case that some of these candidates are not zero-knowledge. Hence, the adversary may be able to distinguish the output of two adjacent hybrids. This is acceptable as per the lemma statement, since in this case we would have a reduction to ZK of that specific candidate.

In $\mathcal{H}_{t+1}$, we replace $t-1$ shares of the real witness w with $t-1$ random group elements in G. This is possible as shares of the real witness are used in at most $t-1$ of the prover algorithms of the candidates[10]. Hence, any adversary can leak at most $t-1$ of the shares, and it is therefore unable to determine whether these are shares of w or random group elements. Since this argument is information theoretical, the two hybrids are identically distributed. Finally, the last hybrid is just a syntactic change that derives from homomorphic operations. Formally, this lemma is proved by having Lemma 7, Lemma 8, and Lemma 9. □

Lemma 7 (Transition from $\mathcal{H}_{j-1}$ to $\mathcal{H}_j$, for $1 \le j \le t$)**.** *If there exists a PPT algorithm $\mathcal{A}$ for which $\mathsf{out}^{\mathcal{H}_{j-1}} \not\approx_c \mathsf{out}^{\mathcal{H}_j}$, there exists a PPT algorithm $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ that breaks zero-knowledge of the i_j-th scheme.*

Proof. Given $\mathcal{A}$ as in the theorem statement, we describe $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ in what follows:

Figure C.4: Reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$

- Receive public parameters pp_{i_j} from the challenger $\mathcal{C}$.
- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T_{j-1}$. Compute $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$. Set $pp \leftarrow \{pp_k\}_{k\in[n]}$.
- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$.
- Compute $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}_{\mathsf{G}}(w)$. Compute $\{x_i \leftarrow f(w_i)\}_{i\in[n]}$.
- Send $\{x_{i_j}, w_{i_j}\}$ to $\mathcal{C}$, and receive proof π_{i_j}. This proof is generated either as $\mathcal{P}_{i_j}(pp_{i_j}, x_{i_j}, w_{i_j})$ or $\mathcal{S}_{i_j}^1(pp_{i_j}, \tau_{i_j}, x_{i_j})$.
- Compute $\pi_k \leftarrow \mathcal{S}_k^1(pp_k, \tau_k, x_k)$ for $k \in T_{j-1}$. Compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, x_i, w_i)$ for $i \in [n] \setminus T_j$.
- Set $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x_k\}_{k\in[n]})$.

[10] More precisely, the real prover is used exactly in $n-t$ of the candidates. Therefore, $\mathcal{H}_1$ removes the dependency on the witness from all the proofs generated by real provers, as well as the first $t-1-(n-t) = 2t-n-1$ elements of the set T (*i.e.*, T_{2t-n-1}). From the honest majority assumption, the cardinality of T_{2t-n-1} is always greater or equal to 0. Looking ahead, this enables that the reconstruction procedure in $\mathcal{H}_2$ to be carried out with sufficiently-many shares.

- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

If $\mathcal{C}$ provides a real proof, we are in hybrid $\mathcal{H}_{j-1}$. Otherwise, we are in hybrid $\mathcal{H}_j$. Therefore, $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking ZK of candidate Π_{i_j}. □

Lemma 8 (Transition from $\mathcal{H}_t$ to $\mathcal{H}_{t+1}$)**.** $\mathsf{out}^{\mathcal{H}_t}$ *and* $\mathsf{out}^{\mathcal{H}_{t+1}}$ *are identically distributed.*

Proof. Assume by contradiction there exists an unbounded algorithm $\mathcal{A}$ that can distinguish $\mathsf{out}^{\mathcal{H}_t}$ from $\mathsf{out}^{\mathcal{H}_{t+1}}$. Then, we can construct the following adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ breaking security (as per Definition 15) of the t-out-of-n Shamir's secret sharing scheme:

Figure C.5: Adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$

- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T$. $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T$. $pp \leftarrow \{pp_k\}_{k\in[n]}$.
- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$, and forward w to the challenger $\mathcal{C}$.
- Receive $t-1$ shares $\{s_i\}_{i\in[n]\setminus T\cup T_{2t-n+1}}$ from $\mathcal{C}$ which are either shares of the real witness w or random group elements $\{r_i\}_{i\in[n]\setminus T\cup T_{2t-n-1}} \xleftarrow{\$} \mathtt{Gen}_{\mathsf{G}}(1^\lambda)$.
- Set $s_0 \leftarrow w$.
- For $l \in T \setminus T_{2t-n-1}$, compute $w_l \leftarrow \overline{\mathtt{reconstruct}}_{\mathsf{G}}(\{s_i\}_{i\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)$.
- For $l \in T \setminus T_{2t-n-1}$, compute $x_l \leftarrow f(w_l)$. For $i \in [n] \setminus T \cup T_{2t-n-1} \cup \{0\}$, compute $x_i \leftarrow f(s_i)$.
- For $l \in T$, compute $\pi_l \leftarrow \mathcal{S}_l^1(pp_l, \tau_l, x_l)$. For $i \in [n] \setminus T$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, x_i, s_i)$.
- Set $\pi \leftarrow \{(\pi_k, x_k)\}_{k\in[n]}$.
- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

First, observe that $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ uses w alongside the shares s_i output by the challenger $\mathcal{C}$ in order to reconstruct shares of the real w for $i \in T \setminus T_{2t-n-1}$. This is necessary, as both $\mathcal{H}_t$ and $\mathcal{H}_{t+1}$ simulate proofs for $i \in T \setminus T_{2t-n-1}$ by supplying $x_i = f(w_i)$ as the input to the simulators of the candidates whose indices are in $T \setminus T_{2t-n-1}$.

We now argue that $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ correctly reproduces the behavior of hybrids $\mathcal{H}_t$ and $\mathcal{H}_{t+1}$ according to the challenger's behaviour. If the challenger shares the real witness w, all the proofs in $[n] \setminus T \cup T_{2t-n-1}$ are generated exactly as in $\mathcal{H}_t$ (*i.e.*, using shares of w). If the challenger generates random group elements in G, all the proofs in $[n] \setminus T \cup T_{2t-n-1}$ are generated exactly as in $\mathcal{H}_{t+1}$ (*i.e.*, using random group elements in G). Therefore, $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking security of Shamir's secret sharing scheme. This is a contradiction; therefore, $\mathsf{out}^{\mathcal{H}_t} = \mathsf{out}^{\mathcal{H}_{t+1}}$. □

Lemma 9 (Transition from $\mathcal{H}_{t+1}$ to $\mathcal{H}_{t+2}$)**.** $\mathsf{out}^{\mathcal{H}_{t+1}}$ *and* $\mathsf{out}^{\mathcal{H}_{t+2}}$ *are identically distributed.*

Proof. The proof follows by comparing how the sub-statements are generated and by applying the property of homomorphic language. More precisely:

- **Comparing** $\{x_l\}_{l\in[n]\setminus T\cup T_{2t-n-1}}$**.** In both experiments, these sub-statements are generated as $f(r_l)$ for a random $r_l \in \mathsf{G}$.
- **Comparing** x_0**.** In $\mathcal{H}_{t+1}$, $x_0 = f(r_0)$, with $r_0 = w$. In $\mathcal{H}_{t+2}$, $x_0 = x$. Since $f(w) = x$ by assumption, the two sub-statements are identical.

- **Comparing $\{x_l\}_{l\in T\setminus T_{2t-n-1}}$.**
  In $\mathcal{H}_{t+1}$, $x_l = f(w_l)$, with $w_l = \overline{\texttt{reconstruct}}_{\mathsf{G}}(\{r_j\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)$. In $\mathcal{H}_{t+2}$, $x_l = \overline{\texttt{reconstruct}}_{\mathsf{H}}(\{x_j\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)$, with $x_j = f(r_j)$. We show that these sub-statements are identical by applying the homomorphic property of the language, similarly to the proof for Theorem 3:

$$\begin{aligned}
f(w_l) &= f(\overline{\texttt{reconstruct}}_{\mathsf{G}}(\{r_j\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)\\
&= f(\bigstar_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}} (r_j\{2\star\}(\prod_{k\in T\setminus T_{2t-n-1}, k\neq j} \frac{l-k}{j-k})))\\
&= \bigotimes_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}} (f(r_j)\{2\otimes\}(\prod_{k\in T\setminus T_{2t-n-1}, k\neq j} \frac{l-k}{j-k}))\\
&= \overline{\texttt{reconstruct}}_{\mathsf{H}}(\{f(r_j)\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)\\
&= \overline{\texttt{reconstruct}}_{\mathsf{H}}(\{x_j\}_{j\in[n]\setminus T\cup T_{2t-n-1}\cup\{0\}}, l)
\end{aligned}$$

□

C.4 Combiner for homomorphic languages: argument-of-knowledge

Proof (Theorem 6). Following Section 5.2, we construct an extractor $\mathcal{E}^T$ and prove in Lemma 10 that either it successfully extracts a witness, or there exists at least one reduction to AoK of one of the candidates whose indices are in T. Then, we argue in Lemma 12 that the existence of adversaries who break argument-of-knowledge of the combiner implies the existence of sufficiently-many reductions to argument-of-knowledge of the underlying candidates, concluding the proof. □

Lemma 10. *There exists an extractor $\mathcal{E}^T$ for which either a witness w is successfully extracted from an accepting proof, or there exists a reduction to argument-of-knowledge of at least one of the candidates $\{\Pi_k\}_{k\in T}$.*

Proof. Let $T = \{i_1, \ldots, i_t\}$ be a set of indices sorted in lexicographic order identifying candidates $\Pi_{i_1}, \ldots, \Pi_{i_t}$. The extractor $\mathcal{E}^T = (\mathcal{E}^{T,0}, \mathcal{E}^{T,1})$ is constructed as follows:

Figure C.6: Extractor $\mathcal{E}^T$

$\mathcal{E}^{T,0}(1^l, 1^\lambda)$:

- For $k \in T$, compute $(pp_k, \xi_k) \xleftarrow{\$} \mathcal{E}_k^0(1^l, 1^\lambda)$.
- Compute $pp \leftarrow \{pp_k\}_{k\in[n]}$, $\xi \leftarrow \{\xi_k\}_{k\in T}$.
- Return (pp, ξ).

$\mathcal{E}^{T,1}(pp, \xi, x, \pi)$:

- For $k \in T$, parse the k-th component of π as (x_k, π_k). Compute $w_k \leftarrow \mathcal{E}_k^1(pp_k, \xi_k, x_k, \pi_k)$.
- Output $w \leftarrow \texttt{reconstruct}_{\mathsf{G}}(\{w_j\}_{j\in T})$.

As per Definition 7, an adversary may break argument-of-knowledge either by detecting the introduction of trapdoor setups, or by outputting an accepting proof that causes any extractor to fail. If either occurs, we exhibit a reduction to AoK of that specific candidate.

Introducing trapdoor setups. We define t hybrid experiments: in hybrid $\mathcal{H}_0$ all public parameters for the candidates in T are generated by setup algorithms; in hybrid $\mathcal{H}_t$, all public parameters for the candidates in T are generated by trapdoor setup algorithms. We denote the output of the adversary in hybrid $\mathcal{H}_i$ as $\mathsf{out}^{\mathcal{H}_i}$, for $i \in \{0, 1, \ldots, t\}$. Formally, for $j \in \{1, \ldots, t\}$ and for any PPT algorithm $\mathcal{A}$, the following holds:

$$\left| \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_{j-1}}) = 1] - \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_j}) = 1] \right| \leq \mathsf{negl}(\lambda).$$

We define the hybrids as follows, and prove that the above holds in Lemma 11:

Figure C.7: Hybrid experiments $\mathcal{H}_0$, $\mathcal{H}_j$, for $1 \leq j \leq t$

$\mathcal{H}_0$, $\boxed{\mathcal{H}_j}$

1. $pp_k \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$ for $k \in [n]$. $pp \leftarrow \{pp_k\}_{k\in[n]}$.

1. $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T_j$. $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$. $pp \leftarrow \{pp_k\}_{k\in[n]}$.

2. Run $\mathcal{A}(pp)$.

Lemma 11 (Transition from $\mathcal{H}_{j-1}$ to $\mathcal{H}_j$, for $1 \leq j \leq t$)**.** *If there exists a PPT algorithm $\mathcal{A}$ for which $\mathsf{out}^{\mathcal{H}_{j-1}} \not\approx_c \mathsf{out}^{\mathcal{H}_j}$, there exists a PPT algorithm $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ that breaks argument-of-knowledge of the i_j-th scheme.*

Proof. Given $\mathcal{A}$ as in the theorem statement, we describe $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ in what follows:

Figure C.8: Reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$

- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{E}_k^0(1^l, 1^\lambda)$ for $k \in T_{j-1}$. Compute $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$.
- Receive public parameters pp_{i_j} from the challenger $\mathcal{C}$. These public parameters are generated either by using $\mathtt{setup}_{i_j}(1^l, 1^\lambda)$ or $\mathcal{E}_{i_j}^0(1^l, 1^\lambda)$.
- Set $pp \leftarrow \{pp_k\}_{k\in[n]}$.
- Run $\mathcal{A}(pp)$ and output whatever $\mathcal{A}$ outputs.

If $\mathcal{C}$ generates the public parameters using the setup algorithm, we are in hybrid $\mathcal{H}_{j-1}$. Otherwise, we are in hybrid $\mathcal{H}_j$. Therefore, $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking AoK of candidate Π_{i_j}. □

Extractor $\mathcal{E}^T$ fails to extract a witness. If any reduction occurred throughout the chain of hybrids, the existence of the extractor in Figure C.6 already implies a reduction to AoK for one of the candidates, as prescribed by the lemma statement. Indeed, if the adversary is able to distinguish whether any of the public parameters of candidates in T is generated by a trapdoor setup algorithm (*i.e.*, $\mathcal{E}_k^0$ for some $k \in T$), its behaviour may change arbitrarily when interacting with $\mathcal{E}^T$.

Conversely, if no reduction occurred so far, the adversary has at most an advantage of $t \cdot \mathsf{negl}(\lambda)$ in discerning the generation algorithm used for $pp = \{pp_k\}_{k\in[n]}$. Hence, it will always

output an accepting proof upon input pp except for negligible probability. Given that, the proof continues only if no reduction was possible in the chain of hybrids, as the adversary interacts with an extractor that controls the trapdoors used to generate pp.

We observe that the only way $\mathcal{E}^T$ can reconstruct w is by successfully extracting all the sub-witnesses in T and using the $\texttt{reconstruct}$ algorithm of Shamir's secret sharing scheme. Since the verifier $\mathsf{C}_{\mathsf{vrfy}}$ runs $\overline{\texttt{reconstruct}}_{\mathsf{H}}$ as a consistency check, when all the checks pass, it means that all sub-statements are shares of the statement x. By the property of $\overline{\texttt{reconstruct}}_{\mathsf{H}}$ we have in Section 5, all the t extracted shares are shares of w. Hence, by correctness of Shamir's secret sharing, we reconstruct w successfully.

Therefore, in order for $\mathcal{E}^T$ to fail, it suffices for one of the sub-extraction procedures to fail, as otherwise the reconstruction of Shamir's secret sharing would have been successful and $\mathcal{E}^T$ would not have failed. Let i^* be an index for which $\mathcal{E}^1_{i^*}$ fails to extract from (x_{i^*}, π_{i^*}). Any adversary that outputs an accepting proof (x, π) for the combiner that contains (x_{i^*}, π_{i^*}) in position i^* immediately causes $\mathcal{E}^T$ to fail, yielding a reduction to AoK of scheme Π_{i^*}. We describe $\mathsf{Red}^{\mathcal{A},\Pi_{i^*}}$ in what follows:

Figure C.9: Reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i^*}}$

- Receive public parameters pp'_{i^*} from the challenger $\mathcal{C}$.
- Compute $pp \xleftarrow{\$} \mathsf{C}_{\mathsf{setup}}(1^l, 1^\lambda)$. Parse pp as $\{pp_j\}_{j\in[n]}$. Replace pp_{i^*} with pp'_{i^*}.
- Forward pp to $\mathcal{A}$.
- Upon receiving (x, π) from $\mathcal{A}$, Parse π as $\{\pi_k, x_k\}_{k\in[n]}$.
- Forward (x_{i^*}, π_{i^*}) to $\mathcal{C}$.

□

Lemma 12. *The existence of adversaries breaking argument-of-knowledge of $\mathcal{E}^T$ as described in Lemma 10 yields reductions to argument-of-knowledge for at least $n - t + 1$ distinct candidates.*

Proof. The proof uses the same combinatorics argument described for the proof of Lemma 2. In Lemma 10, we defined $\binom{n}{t}$ extractors, and the existence of adversaries breaking AoK for the combiner can be used to also break the security of all the extractors. In particular, there exists an index i^* associated with candidate Π_{i^*} for which an adversary either detects the introduction of the trapdoor generation algorithm, or makes sub-extractor $\mathcal{E}^1_{i^*}$ fail upon input an accepting sub-proof. In order to break AoK for all the possible extractors, there should exist $n - t + 1$ such indices that are distinct, *i.e.*, there should exist adversaries breaking AoK for $n - t + 1$ of the underlying schemes. The existence of these adversaries implies the existence of at least $n - t + 1$ reductions to distinct candidates, concluding the proof. □

C.5 Recursive proof combiner: correctness

Proof (Theorem 7). In order for the construction to be an n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner, the following has to hold:

$$\Pr\left[\mathsf{C}_{\mathsf{vrfy}}(pp, x, \mathsf{C}_{\mathsf{prove}}(pp, x, w)) = 1 \mid pp \xleftarrow{\$} \mathsf{C}_{\mathsf{setup}}(1^{|x|}, 1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)$$

The verifier of the combiner accepts a proof if all the sub-proofs are accepting. Denoting $\mathsf{E}_{\mathsf{corr,i}}$ as the event that the i-th sub-proof is accepting, we first argue that $\Pr[\mathsf{E}_{\mathsf{corr,i}}] \geq 1 - \mathsf{negl}(\lambda)$ for all $i \in \binom{n}{t}$. Given that π_i is generated by nesting t candidates, the success probability of

generating an accepting π_i depends on the event that all the sub-proofs generated in the chain of proofs are accepting. Let $\mathsf{E}_{\mathsf{corr,i_k}}$ be the event that the i_k-th proof internally generated as part of π_i is accepting. Since $t = O(1)$, we apply Bernoulli's inequality (*i.e.*, $(1+x)^n \geq 1+nx$) to argue that $\Pr[\mathsf{E}_{\mathsf{corr,i}}]$ is overwhelming:

$$\begin{aligned}
\Pr[\mathsf{E}_{\mathsf{corr,i}}] &= \prod_{k=1}^{t} \Pr[\mathsf{E}_{\mathsf{corr,i_k}}] \\
&\geq (\min_k \{\Pr[\mathsf{E}_{\mathsf{corr,i_k}}]\})^t \\
&\geq (1 - \mathsf{negl}(\lambda))^{O(1)} \\
&\geq 1 - \mathsf{negl}(\lambda) \cdot O(1) \\
&\geq 1 - \mathsf{negl}(\lambda)
\end{aligned}$$

Finally, we observe that, for $t, n \in O(1)$, $\binom{n}{t} \in O(1)$. Denoting $\mathsf{E}_{\mathsf{corr}}$ as the event that all sub-proofs are accepting, we argue that $\Pr[\mathsf{E}_{\mathsf{corr}}]$ is overwhelming:

$$\begin{aligned}
\Pr[\mathsf{E}_{\mathsf{corr}}] &= \prod_{i=1}^{\binom{n}{t}} \Pr[\mathsf{E}_{\mathsf{corr,i}}] \\
&\geq (\min_i \{\Pr[\mathsf{E}_{\mathsf{corr,i}}]\})^{\binom{n}{t}} \\
&\geq (1 - \mathsf{negl}(\lambda))^{O(1)} \\
&\geq 1 - \mathsf{negl}(\lambda)
\end{aligned}$$

□

C.6 Recursive proof combiner: zero-knowledge lemmas

Proof (Lemma 1). Let $T = \{i_1, \ldots, i_t\}$ be a set of indices sorted in lexicographic order identifying candidates $\Pi_{i_1}, \ldots, \Pi_{i_t}$. $\mathcal{S}^T = (\mathcal{S}^{T,0}, \mathcal{S}^{T,1})$ is constructed as follows:

Figure C.10: Simulator $\mathcal{S}^T$

$\mathcal{S}^{T,0}(1^l, 1^\lambda)$:

- For $l \in T$, compute $(pp_l, \tau_l) \xleftarrow{\$} \mathcal{S}_l^0(1^l, 1^\lambda)$.
- For $i \in [n] \setminus T$, compute $pp_i \leftarrow \mathtt{setup}_i(1^l, 1^\lambda)$.
- Compute $pp \leftarrow \{pp_k\}_{k\in[n]}$, $\tau \leftarrow \{\tau_k\}_{k\in[t]}$.
- Return (pp, τ).

$\mathcal{S}^{T,1}(pp, \tau, x)$:

- Index all possible subsets containing t of the n candidates with set $S = \{S^i\}_{i\in\binom{n}{t}}$.
- Denote S^{T_k} as the set of all proofs containing candidate i_k but not candidates in T_{k-1} (*i.e.*, $S^{T_k} = \{S^i \mid \forall i \in \binom{n}{t}\ s.t.\ i_k \in S^i \wedge S^i \cap T_{k-1} = \emptyset\}$). Denote every $S^i \in S^{T_k}$ as $S^{T_k,i}$.
- For $k \in [t]$, for $S^{T_k,i} \in S^{T_k}$, simulate as follows:
  - Compute $\pi_{T_k,i} \leftarrow \mathcal{S}_{i_k}^1(pp_{i_k}, \tau_{i_k}, x_{T_k,i})$, where $x_{T_k,i} \leftarrow (\{\mathcal{V}_l, pp_l\}_{l\in S^{T_k,i} \wedge l < i_k}, x)$, and proceed as follows:
    - If $\pi_{T_k,i}$ is the outermost proof, set $\pi_{T_k,i}^{\mathsf{PoP}} \leftarrow \pi_{T_k,i}$.

* Otherwise, obtain $\pi^{\mathsf{PoP}}_{T_k,i}$ by recursively running the prover algorithm $\{\mathcal{P}_l\}_{l \in S^{T_k,i} \wedge l > i_k}$, using $\pi_{T_k,i}$ as a valid witness (*i.e.*, an accepting proof), as per the combiner description.

The proof of this lemma goes through $t+1$ hybrid experiments, where hybrid $\mathcal{H}_0$ is the same as the real world, and hybrid $\mathcal{H}_t$ is the same as the ideal world. We denote the output of the adversary in hybrid $\mathcal{H}_i$ as $\mathsf{out}^{\mathcal{H}_i}$, for $i \in \{0, 1, \ldots, t\}$. Formally we prove that, for $j \in \{1, \ldots, t\}$ and for any PPT algorithm $\mathcal{A}$, the following holds:

$$\left| \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_{j-1}}) = 1] - \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_j}) = 1] \right| \leq \mathsf{negl}(\lambda).$$

We define the hybrids as follows:

Figure C.11: Hybrid experiments $\mathcal{H}_0$, $\mathcal{H}_j$, for $1 \leq j \leq t$

$\mathcal{H}_0$, $\boxed{\mathcal{H}_j}$

1. $pp \xleftarrow{\$} \mathsf{C}_{\mathsf{setup}}(1^l, 1^\lambda)$.

1. $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}^0_k(1^l, 1^\lambda)$ for $i \in [j]$. $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus [j]$. $pp \leftarrow \{pp_i\}_{i \in [n]}$.

2. Index all possible subsets containing t of the n candidates with set $S = \{S^i\}_{i \in \binom{n}{t}}$.

3. For $S^i \in S$, compute π_i as the honest prover.

3. Denote $S^{T_k} = \{S^i \mid \forall i \in \binom{n}{t}\ s.t.\ i_k \in S^i \wedge S^i \cap T_{k-1} = \emptyset\}$. We denote every $S^i \in S^{T_k}$ as $S^{T_k,i}$.

   For $k \in [j]$, for $S^{T_k,i} \in S^{T_k}$, simulate as follows:

   - Compute $\pi_{T_k,i} \leftarrow \mathcal{S}^1_{i_k}(pp_{i_k}, \tau_{i_k}, x_{T_k,i})$, where $x_{T_k,i} \leftarrow (\{\mathcal{V}_l, pp_l\}_{l \in S^{T_k,i} \wedge l < i_k}, x)$, and proceed as follows:
     - If $\pi_{T_k,i}$ is the outermost proof, set $\pi^{\mathsf{PoP}}_{T_k,i} \leftarrow \pi_{T_k,i}$.
     - Otherwise, obtain $\pi^{\mathsf{PoP}}_{T_k,i}$ by recursively running the prover algorithm $\{\mathcal{P}_l\}_{l \in S^{T_k,i} \wedge l > i_k}$, using $\pi_{T_k,i}$ as a valid witness (*i.e.*, an accepting proof), as per the combiner description.

   For $S^i \in S \setminus \{S^{T_k}\}_{k \in [j]}$, compute π_i as the honest prover of the combiner.

4. Set $\pi \leftarrow \{\pi^{\mathsf{PoP}}_i\}_{i \in \binom{n}{t}}$.

5. Run $\mathcal{A}(pp, x, \pi)$.

Intuitively, in $\mathcal{H}_j$ we simulate all the proofs containing candidate i_j that were not already simulated in any previous hybrid. If the proof to be simulated is the outermost one, the dependency on the witness is removed by running the simulator for the outermost proof system. In any other case, the simulation generates an accepting intermediate proof without knowing its respective witness, which is used as a valid witness by recursively running the prover algorithm as per the combiner description.

Finally, we argue that the simulator is indeed independent from the witness (*i.e.*, that all proofs were simulated) through a simple combinatorics argument. Hence, if no reduction

occurred throughout the hybrid chain, the simulator successfully removes the dependency of the witness from all the $\binom{n}{t}$ proofs. We formalize the above in Lemma 13 and Lemma 14.

On the indistinguishability of the hybrid chain. If no reduction occurred, each simulated proof yields an advantage to the adversary of $\mathsf{negl}(\lambda)$. Hence, the advantage of the adversary from $\mathcal{H}_0$ to $\mathcal{H}_t$ is upper bounded by $\binom{n}{t}\mathsf{negl}(\lambda)$. In order for this quantity to be negligible, it should be the case that $\binom{n}{t} \in \mathsf{poly}(\lambda)$. Because we have $n, t \in O(1)$, $\binom{n}{t} \in O(1)$, satisfying our requirement. □

Lemma 13 (Transition from $\mathcal{H}_{j-1}$ to $\mathcal{H}_j$, for $1 \leq j \leq t$)**.** *If there exists a PPT algorithm $\mathcal{A}$ for which $\mathsf{out}^{\mathcal{H}_{j-1}} \not\approx_c \mathsf{out}^{\mathcal{H}_j}$, there exists a PPT algorithm $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ that breaks zero-knowledge of the i_j-th scheme.*

Proof (Lemma 13). Similarly to the proof for Lemma 7, we describe a PPT reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ given $\mathcal{A}$ as in the theorem statement:

Figure C.12: Reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$

- Receive public parameters pp_{i_j} from the challenger $\mathcal{C}$.
- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T_{j-1}$. Compute $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$. Set $pp \leftarrow \{pp_k\}_{k\in[n]}$.
- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$.
- Index all possible subsets containing t of the n candidates with set $S = \{S^i\}_{i\in\binom{n}{t}}$.
- Denote $S^{T_k} = \{S^i \mid \forall i \in \binom{n}{t}\ s.t.\ i_k \in S^i \wedge S^i \cap T_{k-1} = \emptyset\}$. We denote every $S^i \in S^{T_k}$ as $S^{T_k,i}$.
- For $k \in [j-1]$, for $S^{T_k,i} \in S^{T_k}$, simulate proofs as per the description of $\mathcal{H}_{j-1}$:
- Send (x, w) to $\mathcal{C}$, and receive proofs $\{\pi_l^{\mathsf{PoP}}\}_{l\in S^{T_j}}$. These proofs are either generated by running the honest prover of the combiner, or by running $\mathcal{S}_{i_j}^1$, as described in $\mathcal{H}_j$.
- For $S^i \in S \setminus \{S^{T_k}\}_{k\in[j]}$, compute π_i as the honest prover of the combiner.
- Set $\pi \leftarrow \{\pi_i^{\mathsf{PoP}}\}_{i\in\binom{n}{t}}$.
- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

If $\mathcal{C}$ provides real proofs, we are in hybrid $\mathcal{H}_{j-1}$. Otherwise, we are in hybrid $\mathcal{H}_j$. Therefore, $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking multi-proof ZK of candidate Π_{i_j} [11].

□

Lemma 14. *The output of $\mathcal{S}^T$ is independent from the witness.*

Proof (Lemma 14). First, we observe that $\mathcal{S}^T$ exclusively simulates all the proofs that include any candidate $\{\Pi_k\}_{k\in T}$ as part of their generation procedure. In order to show that the output of $\mathcal{S}^T$ is independent from the witness, we define a set of proofs $\bar{S}$ that do not include any of the candidates in T as part of their generation procedure. The cardinality of $\bar{S}$ is $\binom{n-t}{t}$. Since

[11] For convenience, we use multi-proof zero-knowledge. We remark that this is for compactness of notation in the setup generation algorithm of the combiner, and comes without loss of generality. Indeed, the same can be achieved without multi-proof ZK by generating a different trapdoor for each individual proof.

$t > n/2$ by assumption, it trivially holds that $t > n - t$. This implies that $|\bar{S}| = 0$. Therefore, all proofs are simulated. □

C.7 Recursive proof combiner: argument-of-knowledge lemmas

Proof (Lemma 3). Let $T = \{i_1, \ldots, i_t\}$ be a set of indices sorted in lexicographic order identifying candidates $\Pi_{i_1}, \ldots, \Pi_{i_t}$. The extractor $\mathcal{E}^T = (\mathcal{E}^{T,0}, \mathcal{E}^{T,1})$ is constructed as follows:

Figure C.13: Extractor $\mathcal{E}^T$

$\mathcal{E}^{T,0}(1^l, 1^\lambda)$:

- For $k \in T$, compute $(pp_k, \xi_k) \xleftarrow{\$} \mathcal{E}^0_k(1^l, 1^\lambda)$.
- Compute $pp \leftarrow \{pp_k\}_{k\in[n]}$, $\xi \leftarrow \{\xi_k\}_{k\in T}$.
- Return (pp, ξ).

$\mathcal{E}^{T,1}(pp, \xi, x, \pi)$:

- Parse π as $\{\pi_i^{\mathsf{PoP}}\}_{i\in\binom{n}{t}}$.
- Index all possible subsets containing t of the n candidates with set $S = \{S_i\}_{i\in\binom{n}{t}}$.
- Select the proof π for which $S_i = T$, and set $\pi_{i_t} \leftarrow \pi$.
- For $k \in \{0, \ldots, t-2\}$, compute $\pi_{i_{t-k-1}} = \mathcal{E}^1_{i_{t-k}}(pp_{i_{t-k}}, \xi_{i_{t-k}}, x_{i_{t-k}}, \pi_{i_{t-k}})$.
- Output $w = \mathcal{E}^1_{i_1}(pp_{i_1}, \xi_{i_1}, x, \pi_{i_1})$

Intuitively, the extractor picks the specific proof that was generated using all the candidates in T, and runs the extractors in T recursively until a witness is extracted. Since π is parsed as a set of $\binom{n}{t}$ proofs, and these proofs are enumerated by considering all the subsets of size t out of n, such a proof always exists.

As per Definition 7, an adversary may break argument-of-knowledge either by detecting the introduction of trapdoor setups, or by outputting an accepting proof that causes any extractor to fail. If either occurs, we exhibit a reduction to AoK of that specific candidate. The former step is the same for all our combiners, and is formally proved in Lemma 11. Intuitively, we replace one setup at a time, and if the adversary distinguishes two adjacent hybrids we have a reduction.

Given that no reduction occurred after introducing the trapdoor setups, we proceed with analyzing $\mathcal{E}^T$. The only way $\mathcal{E}^T$ can output w is by recursively extracting from the proof π that was generated by running all the schemes in T. Because all candidates work for any NP language $\mathcal{L}$, they also work with the specific languages we have in Figure 6.1. Therefore, in order for $\mathcal{E}^T$ to fail, it suffices for one of the sub-extraction procedures to fail. Let i^* be an index for which $\mathcal{E}^1_{i^*}$ fails to extract from (x_{i^*}, π_{i^*}). Any adversary that outputs an accepting proof (x, π) for the combiner for which the extraction procedure, upon reaching scheme Π_{i^*}, fails to extract from (x_{i^*}, π_{i^*}), immediately causes $\mathcal{E}^T$ to fail. This yields a reduction to AoK of scheme Π_{i^*}. □

C.8 MPC-in-the-head approach: correctness

Proof (Theorem 10). In order for the construction to be an n-candidate $\mathcal{F}_{\mathsf{NI}}$-combiner, the following has to hold:

$$\Pr\left[\mathsf{C}_{\mathsf{vrfy}}(pp, x, \mathsf{C}_{\mathsf{prove}}(pp, x, w)) = 1 \mid pp \xleftarrow{\$} \mathsf{C}_{\mathsf{setup}}(1^{|x|}, 1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)$$

Intuitively, this happens if (i) all sub-proofs are accepting, and (ii) the commitment corresponding to the outgoing messages are consistent with the commitment corresponding to the inbound messages. Because Π_{MPC} is honestly executed and the commitment are computed honestly, then (ii) holds. Hence, it only remains to show that all the sub-proofs are accepting:

1. By perfect correctness of Π_{MPC}, f_x of Figure 7.1 obtains $w' \leftarrow \mathtt{reconstruct}(\{w_j\}_{j\in[t]})$, where $(w_1, \ldots, w_n) \xleftarrow{\$} \mathtt{share}(w)$. By correctness of secret sharing scheme $\mathtt{SS}$, we have $w' = w$. When $(x, w) \in \mathscr{R}_{\mathcal{L}}$, f_x outputs 1.

2. By Definition 1, for $k \in [n]$, $\mathcal{V}_k$ will output 1 with probability at least $1-\mathsf{negl}(\lambda)$ if and only if $(x'_k, w'_k) \in \mathscr{R}_{k,\mathcal{L}_{\mathsf{MPC}}}$ and Π_k is for $\mathcal{L}_{k,\mathsf{MPC}}$. Since $\Pi_k \in \mathcal{F}_{\mathsf{NI}}$ works for any NP language, it will also work for $\mathcal{L}_{k,\mathsf{MPC}}$. By correctness of Π_{MPC}, $(x'_k, w'_k) \in \mathscr{R}_{\mathcal{L}_{k,\mathsf{MPC}}}$. Therefore, all $\mathcal{V}_k$ will output 1 with probability at least $1 - \mathsf{negl}(\lambda)$.

Hence, when $(x, w) \in \mathscr{R}_{\mathcal{L}}$, the probability $\mathsf{C}_{\mathsf{vrfy}}$ accepts is the same as the probability when all sub-proofs are accepting, which is exactly the same as Event 1 in the proof of Theorem 3. □

C.9 MPC-in-the-head approach: statistical soundness

Proof (Theorem 11). The proof is very similar to the proof of Theorem 4. The only difference is that we need to argue that there must be $n - t + 1$ reductions to the statistical soundness of the candidates by relying on the statistical binding of the non-interactive commitment scheme and on the $\mathcal{K}$-robustness of the MPC protocol.

Formally, assume by contradiction that there are at most $n-t$ reductions. This means there will be $n - (n - t) = t$ sub-statements x'_k that are in the language $\mathcal{L}_{k,\mathsf{MPC}}$, which means that there are t parties that output 1. By assumption, we know $x \notin \mathcal{L}$. This means that $(x, w') \notin \mathscr{R}_{\mathcal{L}}$ holds for any w'. In this case, $f_x(w_1, \ldots, w_n)$ must output 0.

Because the verifier accepts, the commitments corresponding to the outgoing messages are consistent with the commitments corresponding to the inbound messages. In this case, we claim that all pairs of views are consistent. Let $\mathsf{E}_{\mathsf{bad}}$ be the event for which the unbounded (malicious) prover $\tilde{\mathcal{P}}$ produces two consistent commitments that open to inconsistent views. This means that there exist two openings o_0, o_1 for the same commitment that open to the same value. Given that $\mathtt{NIC}$ is statistically binding, this event happens only with negligible probability.

Hence, conditioning on $\mathsf{E}_{\mathsf{bad}}$ not happening, all views come from the same execution of Π_{MPC} realizing f_x. In such an execution, by perfect $\mathcal{K}$-robustness, when there are at most $\mathcal{K}$ corrupted parties (controlled by a malicious adversary) that output 1, the other t parties must output 0. Since perfect $\mathcal{K}$-robustness holds unconditionally, we have a contradiction, as in this case the verification procedure of the combiner must have been rejecting, because some parties are outputting 0. This is the case as, in order to break $\mathcal{K}$-robustness, at least one more party should output 1. But the existence such a party would imply the existence of more reductions to the underlying primitives, contradicting the assumption of having at most $n - t$ reductions.

Therefore, any accepting proof π for the combiner for a statement $x \notin \mathcal{L}$ implies the existence of at least $n - t + 1$ reductions to statistical soundness of the underlying candidates. Once we have at least $n - t + 1$ reductions, the proof is the same as the proof of Theorem 4. □

C.10 MPC-in-the-head approach: zero-knowledge

Proof (Theorem 12). The structure of this proof is the same as the one of Theorem 5 and uses exactly the same the combinatorics argument used to prove Lemma 2. It therefore suffices to construct a simulator $\mathcal{S}^T$, and argue that either its output is indistinguishable from the

output of the real prover, or there exists at least one reduction to zero-knowledge of one of the candidates whose indices are in T. We describe such a $\mathcal{S}^T$ in Lemma 15. □

Lemma 15. *There exists a simulator $\mathcal{S}^T$ of $\mathsf{C}_{\mathsf{MPC}}$ for which either $\{\mathtt{REAL}\} \approx_c \{\mathtt{IDEAL}_{\mathcal{S}^T}\}$, or there exists a reduction to zero-knowledge of at least one of the candidates $\{\Pi_k\}_{k\in[T]}$.*

Proof. Let $T = \{i_1, \ldots, i_t\}$ be a set of indices sorted in lexicographic order identifying candidates $\Pi_{i_1}, \ldots, \Pi_{i_t}$. Assume $\mathtt{Gen}$ is an algorithm that can sample uniformly random elements in the space of shares of w. $\mathcal{S}^T = (\mathcal{S}^{T,0}, \mathcal{S}^{T,1})$ is constructed as follows:

Figure C.14: Simulator $\mathcal{S}^T$

$\mathcal{S}^{T,0}(1^l, 1^\lambda)$:

- For $l \in T$, compute $(pp_l, \tau_l) \xleftarrow{\$} \mathcal{S}_l^0(1^l, 1^\lambda)$.
- For $i \in [n] \setminus T$, compute $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$.
- Compute $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$.
- Compute $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$, $\tau \leftarrow \{\tau_k\}_{k\in T}$.
- Return (pp, τ).

$\mathcal{S}^{T,1}(pp, \tau, x)$:

- Sample $\{w_j\}_{j\in[n]\setminus T} \xleftarrow{\$} \mathtt{Gen}(1^\lambda)$. Run the $\mathcal{K}$-privacy simulator $\mathcal{S}_{\mathsf{MPC}}([n] \setminus T, x, \{w_j\}_{j\in[n]\setminus T}, 1)$ of Π_{MPC} to obtain the simulated views $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
- For $i \in T$, for $j \in [n]$, set $\{\mathsf{msg}^k_{i\to j}\}_{k\in[R]} = 0$ in view_i and view_j. Complete view_i to be consistent with $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
- For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]})$.
- For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$. For $l \in [n] \setminus T$, $w'_l \leftarrow (\mathsf{view}_l, \{o_{l,j}\}_{j\in[n]}, \{o_{j,l}\}_{j\in[n]})$.
- For $l \in T$, compute $\pi_l \leftarrow \mathcal{S}_l^1(pp_l, \tau_l, (pp_{\mathtt{NIC}}, x'_l))$. For $i \in [n] \setminus T$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$.
- Return $(\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.

The proof of this lemma goes through $t + 4$ hybrid experiments, where hybrid $\mathcal{H}_0$ is the same as the real world, and hybrid $\mathcal{H}_{t+3}$ is the same as the ideal world. We denote the output of the adversary in hybrid $\mathcal{H}_i$ as $\mathsf{out}^{\mathcal{H}_i}$, for $i \in \{0, 1, \ldots, t + 3\}$. Formally we prove that, for $j \in \{1, \ldots, t + 3\}$ and for any PPT algorithm $\mathcal{A}$, the following holds:

$$\left| \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_{j-1}}) = 1] - \Pr[\mathcal{A}(\mathsf{out}^{\mathcal{H}_j}) = 1] \right| \leq \mathsf{negl}(\lambda).$$

We define the hybrids as follows:

Figure C.15: Hybrid experiments $\mathcal{H}_0$, $\mathcal{H}_j$, $\mathcal{H}_{t+1}$, $\mathcal{H}_{t+2}$, $\mathcal{H}_{t+3}$, for $1 \leq j \leq t$

$\mathcal{H}_0$, $\boxed{\mathcal{H}_j}$, $\boxed{\mathcal{H}_{t+1}}$, $\boxed{\mathcal{H}_{t+2}}$, $\boxed{\mathcal{H}_{t+3}}$

1. $pp_k \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$ for $k \in [n]$. $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$.

1. $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T_j$. $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$. $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$.

2. Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$

3. $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}(w)$. Run MPC protocol Π_{MPC} with input $(w_1, \ldots, w_n)$ to obtain $\{\mathsf{view}_i\}_{i\in[n]}$.

3. • $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}(w)$. Run MPC protocol Π_{MPC} with input $(w_1, \ldots, w_n)$ to obtain $\{\mathsf{view}_i\}_{i\in[n]}$.
   • For $i \in T$, for $j \in [n]$, set $\{\mathsf{msg}_{i\to j}^k\}_{k\in[R]} = 0$ in view_i and view_j.

3. • $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}(w)$. Run the $\mathcal{K}$-privacy simulator $\mathcal{S}_{\mathsf{MPC}}([n] \setminus T, x, \{s_j\}_{j\in[n]\setminus T}, 1)$ of Π_{MPC} to obtain the simulated views $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
   • For $i \in T$, for $j \in [n]$, set $\{\mathsf{msg}_{i\to j}^k\}_{k\in[R]} = 0$ in view_i and view_j. Complete view_i to be consistent with $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.

3. • Sample $\{w_j\}_{j\in[n]\setminus T} \xleftarrow{\$} \mathtt{Gen}(1^\lambda)$. Run the $\mathcal{K}$-privacy simulator $\mathcal{S}_{\mathsf{MPC}}([n] \setminus T, x, \{s_j\}_{j\in[n]\setminus T}, 1)$ of Π_{MPC} to obtain the simulated views $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
   • For $i \in T$, for $j \in [n]$, set $\{\mathsf{msg}_{i\to j}^k\}_{k\in[R]} = 0$ in view_i and view_j. Complete view_i to be consistent with $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.

4. For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}_{i\to j}^k\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}_{j\to i}^k\}_{k\in[R]})$.

5. For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$ and $w'_i \leftarrow (\mathsf{view}_i, \{o_{i,j}\}_{j\in[n]}, \{o_{j,i}\}_{j\in[n]})$.

6. $\pi_k \leftarrow \mathcal{P}_k(pp_k, (pp_{\mathtt{NIC}}, x'_k), w'_k)$ for $k \in [n]$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.

6. $\pi_k \leftarrow \mathcal{S}_k^1(pp_k, \tau_k, (pp_{\mathtt{NIC}}, x'_k))$ for $k \in T_j$. $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$ for $i \in [n] \setminus T_j$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.

6. $\pi_k \leftarrow \mathcal{S}_k^1(pp_k, \tau_k, (pp_{\mathtt{NIC}}, x'_k))$ for $k \in T$. $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$ for $i \in [n] \setminus T$. $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.

7. Run $\mathcal{A}(pp, x, \pi)$.

Intuitively, in the first t hybrids, the real proofs generated by the candidates in T are gradually replaced with simulated proofs generated by their respective simulator. In $\mathcal{H}_{t+1}$, we use the computational hiding property of $\mathtt{NIC}$ to replace the messages sent by honest parties with the message 0, and update the views accordingly so the commitments remain consistent. In $\mathcal{H}_{t+2}$, we run $\mathcal{S}_{\mathsf{MPC}}([n] \setminus T, x, \{w_j\}_{j\in[n]\setminus T}, 1)$ in place of the MPC protocol to obtain simulated views for the semi-honest parties. By $\mathcal{K}$-privacy of Π_{MPC}, such a simulator exists. Finally, in $\mathcal{H}_{t+3}$, we replace $n - t$ shares of the real witness w with $n - t$ random elements in the same space of shares of w. This is possible as shares of the real witness are used in at most $n - t$ of the prover algorithms of the candidates. Because $t > \frac{n}{2}$, the adversary can leak at most $\lfloor \frac{n}{2} \rfloor$ of the shares, and is unable to determine whether these are shares of w or random elements. Formally, this

lemma is proved in Lemma 16, Lemma 17, Lemma 18, and Lemma 19 □

Lemma 16 (Transition from $\mathcal{H}_{j-1}$ to $\mathcal{H}_j$, for $1 \leq j \leq t$)**.** *If there exists a PPT algorithm $\mathcal{A}$ for which $\mathsf{out}^{\mathcal{H}_{j-1}} \not\approx_c \mathsf{out}^{\mathcal{H}_j}$, there exists a PPT algorithm $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ that breaks zero-knowledge of the i_j-th scheme.*

Proof. Given $\mathcal{A}$ as in the theorem statement, we describe $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ in what follows:

Figure C.16: Reduction $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$

- Receive public parameters pp_{i_j} from the challenger $\mathcal{C}$.
- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}_k^0(1^l, 1^\lambda)$ for $k \in T_{j-1}$. Compute $pp_i \xleftarrow{\$} \mathtt{setup}_i(1^l, 1^\lambda)$ for $i \in [n] \setminus T_j$. Compute $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. Set $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$.
- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$.
- Compute $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}(w)$. Run MPC protocol Π_{MPC} with input $(w_1, \ldots, w_n)$ to obtain $\{\mathsf{view}_i\}_{i\in[n]}$.
- For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]})$.
- For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$ and $w'_i \leftarrow (\mathsf{view}_i, \{o_{i,j}\}_{j\in[n]}, \{o_{j,i}\}_{j\in[n]})$.
- Send $\{x'_{i_j}, w'_{i_j}\}$ to the challenger $\mathcal{C}$, and receive proof π_{i_j}. This proof is generated by the challenger $\mathcal{C}$ either as $\mathcal{P}_{i_j}(pp_{i_j}, (pp_{\mathtt{NIC}}, x'_{i_j}), w'_{i_j})$ or $\mathcal{S}^1_{i_j}(pp_{i_j}, \tau_{i_j}, (pp_{\mathtt{NIC}}, x'_{i_j}))$.
- For $k \in T_{j-1}$, compute $\pi_k \leftarrow \mathcal{S}^1_k(pp_k, \tau_k, (pp_{\mathtt{NIC}}, x'_k))$. For $i \in [n] \setminus T_j$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$.
- Set $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.
- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

If $\mathcal{C}$ provides a real proof, we are in hybrid $\mathcal{H}_{j-1}$. Otherwise, we are in hybrid $\mathcal{H}_j$. Therefore, $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking ZK of candidate Π_{i_j}. □

Lemma 17 (Transition from $\mathcal{H}_t$ to $\mathcal{H}_{t+1}$)**.** $\mathsf{out}^{\mathcal{H}_t} \approx_c \mathsf{out}^{\mathcal{H}_{t+1}}$.

Proof. The indistinguishability goes through nt intermediate hybrids $\mathcal{H}_{s,e}$, with $s \in [t]$ and $e \in [n]$. The difference between two intermediate hybrids is that in $\mathcal{H}_{s,e-1}$, $(com_{i_s,i_e}, o_{i_s,i_e}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i_s\to i_e}\}_{k\in[R]})$, whereas in $\mathcal{H}_{s,e}$, $(com_{i_s,i_e}, o_{i_s,i_e}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, 0)$. This way, we effectively replace the set of messages from party i_s to party i_e (*i.e.*, $\{\mathsf{msg}^k_{i_s\to i_e}\}$) with 0. We note that, for $s \neq 1$, $\mathcal{H}_{s,0}$ is $\mathcal{H}_{s-1,n}$. We also note that $\mathcal{H}_{1,0}$ is $\mathcal{H}_t$, and $\mathcal{H}_{t,n}$ is $\mathcal{H}_{t+1}$.

Assume by contradiction there exists a PPT algorithm $\mathcal{A}$ that can distinguish $\mathsf{out}^{\mathcal{H}_{s,e-1}}$ from $\mathsf{out}^{\mathcal{H}_{s,e}}$. Then, we can construct the following adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathtt{NIC}}_{i,j}$ breaking computational hiding of $\mathtt{NIC}$.

Figure C.17: Adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathtt{NIC}}_{s,e}$

- Receive public parameters $\overline{pp}$ from the challenger $\mathcal{C}$.
- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}^0_k(1^l, 1^\lambda)$ for $k \in T$. $pp_i \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$ for $i \in [n] \setminus T$. Set $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{\overline{pp}\}$.

- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$.
- Compute $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}(w)$. Run MPC protocol Π_{MPC} with input $(w_1, \ldots, w_n)$ to obtain $\{\mathsf{view}_i\}_{i\in[n]}$.
- For $i \in T_s$:
  - if $i \neq i_s$, for $j \in [n]$, set $\{\mathsf{msg}^k_{i\to j}\}_{k\in[R]} = 0$ in view_i and view_j.
  - if $i = i_s$, for $j \in T_{e-1}$, set $\{\mathsf{msg}^k_{i\to j}\}_{k\in[R]} = 0$ in view_i and view_j.
- For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]})$.
- Send 0 and $\{\mathsf{msg}^k_{i_s\to i_e}\}_{k\in[R]}$ to the challenger $\mathcal{C}$, and receive the commitment com' that could either be a commitment of 0 or a commitment of the messages. Set $com_{i_s,i_e} \leftarrow com'$.
- For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$ and $w'_i \leftarrow (\mathsf{view}_i, \{o_{i,j}\}_{j\in[n]}, \{o_{j,i}\}_{j\in[n]})$.
- For $k \in T$, compute $\pi_k \leftarrow \mathcal{S}^1_k(pp_k, \tau_k, (pp_{\mathtt{NIC}}, x'_k))$. For $i \in [n] \setminus T$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$.
- Set $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.
- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

If $\mathcal{C}$ provides a real commitment, we are in intermediate hybrid $\mathcal{H}_{s,e-1}$. Otherwise, we are in intermediate hybrid $\mathcal{H}_{s,e}$. Therefore, $\overline{\mathcal{A}}^{\mathcal{A},\mathtt{NIC}}_{s,e}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking computation hiding property of $\mathtt{NIC}$. This is a contradiction; therefore, $\mathsf{out}^{\mathcal{H}_{s,e-1}} \approx_c \mathsf{out}^{\mathcal{H}_{s,e}}$. By applying nt intermediate hybrids, we have $\mathsf{out}^{\mathcal{H}_t} \approx_c \mathsf{out}^{\mathcal{H}_{t+1}}$.

Finally we observe that, by Lemma 4, there exists a construction of computationally hiding commitments that exclusively relies on the soundness and zero-knowledge of the NIZK candidates. Given that we are considering statistical soundness, which is unconditional, we proceed similarly to Theorem 11 by conditioning on the event that at least t candidates are statistically sound. Since this event occurs with overwhelming probability and is unconditional, we exclusively obtain reductions to zero-knowledge of the candidates. Hence, breaking computational hiding of $\mathtt{NIC}$ implies breaking zero-knowledge of the candidates, as desired. □

Lemma 18 (Transition from $\mathcal{H}_{t+1}$ to $\mathcal{H}_{t+2}$)**.** $\mathsf{out}^{\mathcal{H}_{t+1}}$ *and* $\mathsf{out}^{\mathcal{H}_{t+2}}$ *are identically distributed, if* Π_{MPC} *has perfect* $\mathcal{K}$*-privacy.*

Proof. Assume by contradiction there exists an unbounded algorithm $\mathcal{A}$ that can distinguish $\mathsf{out}^{\mathcal{H}_{t+1}}$ from $\mathsf{out}^{\mathcal{H}_{t+2}}$. Then, we can construct the following adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{MPC}}$ breaking perfect $\mathcal{K}$-privacy of Π_{MPC}:

Figure C.18: Adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{MPC}}$

- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}^0_k(1^l, 1^\lambda)$ for $k \in T$. $pp_i \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$ for $i \in [n] \setminus T$. Compute $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. Set $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$.
- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$.
- Compute $\{w_i\}_{i\in[n]} \leftarrow \mathtt{share}(w)$. Send $\{w_i\}_{i\in[n]}$ to the challenger $\mathcal{C}$.

- The challenger $\mathcal{C}$ for $\mathcal{K}$-privacy computes $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$ either by running a real execution of the protocol or by using the $\mathcal{K}$-privacy simulator $\mathcal{S}(\mathcal{I}, x, \{w_k\}_{k\in[\mathcal{I}]}, 1)$, with $\mathcal{I} = [n] \setminus T$.
- For $i \in T$, for $j \in [n]$, set $\{\mathsf{msg}^k_{i\to j}\}_{k\in[R]} = 0$ in view_i and view_j. Complete view_i to be consistent with $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
- For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]})$.
- For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$. For $l \in [n] \setminus T$, $w'_l \leftarrow (\mathsf{view}_l, \{o_{l,j}\}_{j\in[n]}, \{o_{j,l}\}_{j\in[n]})$.
- For $l \in T$, compute $\pi_l \leftarrow \mathcal{S}^1_l(pp_l, \tau_l, (pp_{\mathtt{NIC}}, x'_l))$. For $i \in [n] \setminus T$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$.
- Set $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.
- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

If $\mathcal{C}$ produces views coming from real parties, we are in hybrid $\mathcal{H}_{t+1}$. Otherwise, we are in hybrid $\mathcal{H}_{t+2}$. Therefore, $\mathsf{Red}^{\mathcal{A},\Pi_{i_j}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking perfect $\mathcal{K}$-privacy of Π_{MPC}. □

Lemma 19 (Transition from $\mathcal{H}_{t+2}$ to $\mathcal{H}_{t+3}$)**.** $\mathsf{out}^{\mathcal{H}_{t+2}}$ *and* $\mathsf{out}^{\mathcal{H}_{t+3}}$ *are identically distributed.*

Proof. Assume by contradiction there exists an unbounded algorithm $\mathcal{A}$ that can distinguish $\mathsf{out}^{\mathcal{H}_{t+2}}$ from $\mathsf{out}^{\mathcal{H}_{t+3}}$. Then, we can construct the following adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ breaking perfect security of the t-out-of-n secret sharing scheme:

Figure C.19: Adversary $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$

- Compute $(pp_k, \tau_k) \xleftarrow{\$} \mathcal{S}^0_k(1^l, 1^\lambda)$ for $k \in T$. $pp_i \xleftarrow{\$} \mathtt{setup}_k(1^l, 1^\lambda)$ for $i \in [n] \setminus T$. Compute $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$. Set $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$.
- Compute $(x, w) \xleftarrow{\$} \mathcal{A}(pp)$, and forward w to the challenger $\mathcal{C}$.
- Receive $n - t$ shares $\{s_j\}_{j\in[n]\setminus T}$ from the challenger $\mathcal{C}$ which are either shares of the real witness w or random elements $\{r_j\}_{j\in[n]\setminus T} \xleftarrow{\$} \mathtt{Gen}(1^\lambda)$.
- Run the $\mathcal{K}$-privacy simulator $\mathcal{S}_{\mathsf{MPC}}([n] \setminus T, x, \{s_j\}_{j\in[n]\setminus T}, 1)$ of Π_{MPC} to obtain the simulated views $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
- For $i \in T$, for $j \in [n]$, set $\{\mathsf{msg}^k_{i\to j}\}_{k\in[R]} = 0$ in view_i and view_j. Complete view_i to be consistent with $\{\mathsf{view}_j\}_{j\in[n]\setminus T}$.
- For $i \in [n]$, $j \in [n]$, compute $(com_{i,j}, o_{i,j}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{i\to j}\}_{k\in[R]})$, and $(com_{j,i}, o_{j,i}) \leftarrow \mathtt{commit}(pp_{\mathtt{NIC}}, \{\mathsf{msg}^k_{j\to i}\}_{k\in[R]})$.
- For $i \in [n]$, set $x'_i \leftarrow (\{com_{i,j}\}_{j\in[n]}, \{com_{j,i}\}_{j\in[n]})$. For $l \in [n] \setminus T$, $w'_l \leftarrow (\mathsf{view}_l, \{o_{l,j}\}_{j\in[n]}, \{o_{j,l}\}_{j\in[n]})$.
- For $l \in T$, compute $\pi_l \leftarrow \mathcal{S}^1_l(pp_l, \tau_l, (pp_{\mathtt{NIC}}, x'_l))$. For $i \in [n] \setminus T$, compute $\pi_i \leftarrow \mathcal{P}_i(pp_i, (pp_{\mathtt{NIC}}, x'_i), w'_i)$.
- Set $\pi \leftarrow (\{\pi_k\}_{k\in[n]}, \{x'_k\}_{k\in[n]})$.
- Run $\mathcal{A}(pp, x, \pi)$ and output whatever $\mathcal{A}$ outputs.

We now argue that $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ correctly reproduces the behavior of hybrids $\mathcal{H}_{t+2}$ and $\mathcal{H}_{t+3}$ according to the challenger's behaviour. If the challenger shares the real witness w, all the proofs in $[n] \setminus T$ are generated exactly as in $\mathcal{H}_{t+2}$ (*i.e.*, using shares of w). If the challenger generates random elements by using $\mathtt{Gen}$, all the proofs in $[n] \setminus T$ are generated exactly as in $\mathcal{H}_{t+3}$. Also, because we have $t > \frac{n}{2}$, $n - t \leq t - 1$. Therefore, $\overline{\mathcal{A}}^{\mathcal{A},\mathsf{SS}}$ retains the same distinguishing advantage of $\mathcal{A}$, breaking perfect security of the t-out-of-n secret sharing scheme. This is a contradiction; therefore, $\mathsf{out}^{\mathcal{H}_{t+2}}$ and $\mathsf{out}^{\mathcal{H}_{t+3}}$ and identically distributed. □

C.11 MPC-in-the-head approach: proof-of-knowledge

Proof (Theorem 13). The structure of this proof is the same as the one of Theorem 6 and uses exactly the same the combinatorics argument used to prove Lemma 12. It therefore suffices to construct an extractor $\mathcal{E}^T$, and argue that either $\mathcal{E}^T$ successfully extracts a witness, or there exists at least one reduction to PoK of one of the candidates whose indices are in T. We describe such an $\mathcal{E}^T$ in Lemma 20. □

Lemma 20. *There exists an extractor $\mathcal{E}^T$ for which either a witness w is successfully extracted from an accepting proof, or there exists a reduction to PoK of at least one of the candidates $\{\Pi_k\}_{k\in T}$.*

Proof. Let $T = \{i_1, \ldots, i_t\}$ be a set of indices sorted in lexicographic order identifying candidates $\Pi_{i_1}, \ldots, \Pi_{i_t}$. The extractor $\mathcal{E}^T = (\mathcal{E}^{T,0}, \mathcal{E}^{T,1})$ is constructed as follows:

Figure C.20: Extractor $\mathcal{E}^T$

$\mathcal{E}^{T,0}(1^l, 1^\lambda)$:

- For $k \in T$, compute $(pp_k, \xi_k) \xleftarrow{\$} \mathcal{E}_k^0(1^l, 1^\lambda)$.
- Compute $pp_{\mathtt{NIC}} \xleftarrow{\$} \mathtt{setup}(1^\lambda)$.
- Compute $pp \leftarrow \{pp_k\}_{k\in[n]} \cup \{pp_{\mathtt{NIC}}\}$, $\xi \leftarrow \{\xi_k\}_{k\in T}$.
- Return (pp, ξ).

$\mathcal{E}^{T,1}(pp, \xi, x, \pi)$:

- For $k \in T$, parse the k-th component of π as (x'_k, π_k). Compute $w'_k \leftarrow \mathcal{E}_k^1(pp_k, \xi_k, (pp_{\mathtt{NIC}}, x'_k), \pi_k)$. Parse w'_k as $(\mathsf{view}_k, \{o_{k,j}\}_{j\in[n]}, \{o_{j,k}\}_{j\in[n]})$. Take w_k from view_k.
- Output $w \leftarrow \mathtt{reconstruct}(\{w_k\}_{k\in T})$.

An adversary may break PoK either by detecting the introduction of trapdoor setups, or by outputting an accepting proof that causes any extractor to fail. If either occurs, we exhibit a reduction to PoK of that specific candidate.

Introducing trapdoor setups. This step is nearly the same as in the proof of Lemma 10 (the only difference is now the adversary is unbounded), and is therefore omitted.

Extractor $\mathcal{E}^T$ fails to extract a witness. We conclude the proof exactly as the proof of Lemma 10: given that no reduction occurred, we proceed with analyzing $\mathcal{E}^T$.

We observe that the only way $\mathcal{E}^T$ can reconstruct w is by successfully extracting all the sub-witnesses in T, and using the $\mathtt{reconstruct}$ algorithm of the t-out-of-n secret sharing scheme. Because the verifier $\mathsf{C}_{\mathsf{vrfy}}$ performs checks for the consistency of the commitments, when all the

checks pass, it means that all pair of views are consistent with the execution of Π_{MPC} because $\mathtt{NIC}$ is statistically binding. The proof is the same as what we have in the proof of Theorem 11.

When all the sub-extraction procedures succeed, we can extract sub-witnesses from t sub-proofs successfully. This implies that these sub-statements x'_k are in the language $\mathcal{L}_{k,\mathsf{MPC}}$, meaning that the output of parties $\{P_k\}_{k\in T}$ is 1. It also means there are at most $n-t$ parties that output 0, satisfying $\mathcal{K}$-robustness. Hence, the statement x is in the language, and using these t sub-witnesses as the input of the t-out-of-n secret sharing scheme allows to reconstruct w successfully.

Therefore, in order for $\mathcal{E}^T$ to fail, it suffices for one of the sub-extraction procedures to fail. Let i^* be an index for which $\mathcal{E}^1_{i^*}$ fails to extract from (x'_{i^*}, π_{i^*}). Any adversary that outputs an accepting proof (x, π) for the combiner that contains (x'_{i^*}, π_{i^*}) in position i^* immediately causes $\mathcal{E}^T$ to fail. As for the proof of Lemma 10, this yields a reduction to PoK of scheme Π_{i^*}.

□