A preliminary version of this paper appears in the proceedings of *the 42nd International Cryptology Conference* (CRYPTO 2022). This is the full version.

A New Approach to Efficient Non-Malleable Zero-Knowledge*

Allen Kim, Xiao Liang, Omkant Pandey

Stony Brook University, Stony Brook, USA
{allekim,liang1,omkant}@cs.stonybrook.edu

Abstract. Non-malleable zero-knowledge, originally introduced in the context of man-in-the-middle attacks, serves as an important building block to protect against concurrent attacks where different protocols may coexist and interleave. While this primitive admits almost optimal constructions in the plain model, they are *several* orders of magnitude slower in practice than standalone zero-knowledge. This is in sharp contrast to non-malleable *commitments* where practical constructions (under the DDH assumption) have been known for a while.

We present a new approach for constructing efficient non-malleable zero-knowledge for all languages in $\mathcal{NP}$, based on a new primitive called *instance-based non-malleable commitment* (IB-NMC). We show how to construct practical IB-NMC by leveraging the fact that *simulators* of *sub-linear* zero-knowledge protocols can be much faster than the honest prover algorithm. With an efficient implementation of IB-NMC, our approach yields the first general-purpose non-malleable zero-knowledge protocol that achieves practical efficiency *in the plain model*.

All of our protocols can be instantiated from symmetric primitives such as block-ciphers and collision-resistant hash functions, have reasonable efficiency in practice, and are general-purpose. Our techniques also yield the first efficient non-malleable commitment scheme *without public-key assumptions*.

Keywords: Non-malleability · Efficiency · Symmetric Assumptions.

* This material is based upon work supported in part by DARPA SIEVE Award HR00112020026, NSF CAREER Award 2144303, NSF grants 1907908, 2028920, 2106263, and 2128187. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, DARPA, or NSF.

Table of Contents

1 Introduction

Non-malleable Zero-Knowledge. Dolev, Dwork, and Naor [DDN91] introduced the notion on non-malleable cryptography. They also provided constructions of non-malleable zero-knowledge and non-malleable commitments in the *plain* model assuming only the existence of *one-way functions* (OWFs). While these primitives were originally introduced in the context of "man-in-the-middle" attacks, they were soon used as a *building block* for constructing secure computation protocols. For example, non-malleable commitments were used extensively to improve their round-efficiency [KOS03, PPV08, Wee10, Goy11, GMPP16, BGJ+18, CCG+20], and non-malleable zero-knowledge played a central role in protecting them against concurrent attacks [Can00, Can01, CLOS02, Pas03b, PS04, MPR06, BDH+17].

A long line of research has since focused on several aspects of these primitives, including their round-complexity [DDN91, Bar02, PR05b, LPV08, LP09, Wee10, LP11b, Goy11, COSV17, Khu17], black-box usage of underlying primitives [GLOV12, GRRV14], and even concrete efficiency [BGR+15] without assuming any trusted setup. Notably, constant-round non-malleable commitments assuming only OWFs were first constructed in independent and concurrent works of Goyal [Goy11] and Lin and Pass [LP11b]. Finally, four-round non-malleable *zero-knowledge* assuming only OWFs was first achieved by Goyal et al. [GRRV14] for all of $\mathcal{NP}$; and three-round non-malleable commitments assuming injective OWFs were constructed by Goyal, Pandey, and Richelson [GPR16, GR19]. Under falsifiable assumptions [Nao03, GW11], these rounds are optimal for commitments [Pas11], and likely to be optimal for zero-knowledge as well [GO94, FGJ18]. Stronger forms of this notion such as *concurrent* non-malleability, eventually achieved optimally in a series of works [PR05a, LPV08, COSV17], are not considered in this work. We note that non-malleability has been explored in several other contexts as well [DDN91, BCFW09, DW09, DPW10].

Efficient Constructions. While the aforementioned results are almost optimal for non-malleable zero-knowledge, their focus is primarily on *feasibility* as opposed to actual efficiency. To the best of our knowledge, the actual efficiency of non-malleable zero-knowledge has never been explicitly addressed before. This is in sharp contrast to non-malleable *commitments*, for which efficient plain-model constructions are known (under the DDH assumption) [BGR+15].

We therefore consider the efficiency of some of the main approaches for non-malleable zero-knowledge. Unless stated otherwise, we are concerned with *general-purpose* protocols (that work for all languages in $\mathcal{NP}$) *in the plain model*.

- The most common approach for non-malleable zero-knowledge is "commit-and-prove." At a high level, the prover first sends a *non-malleable* commitment to the witness, and then uses (ordinary) zero-knowledge to prove that the committed value is a valid witness [DDN91, BPS06, GRRV14, COSV17]. If the commitment supports k-bit identities and has λ-bit security, the circuit corresponding to the state-of-the-art non-malleable commitment [BGR+15] is at least $16k^2\lambda^2$, or over 100 million gates for $k = 32, \lambda = 80$. Zero-knowledge proofs for such circuits would take more than one minute using state-of-the-art (plain-model) protocols such as Ligero [AHIV17] (even taking advantage of the amortization admitted by Ligero). This is true even if the actual statement, say proving $y = \mathrm{SHA256}(x)$, requires less than a second [AHIV17] in the standalone case.[1]

[1] Although details may vary, known protocols in this paradigm generally require some form of non-algebraic consistency proof over a non-malleable commitment supporting large identities and message spaces.

It is worth noting that using state-of-the-art commitments [BGR+15] additionally requires assuming DDH, whereas "symmetric assumptions" such as OWFs are sufficient in theory. Efficient non-malleable commitments *without* relying on public-key assumptions such as DDH are therefore also not known. One option here is to implement the consistency proofs in [BGR+15] with Ligero to avoid DDH. However, this also results in large circuits.[2] Jumping ahead, our techniques offer new results for efficient non-malleable commitments, too.

– Non-malleable zero-knowledge *without* relying on non-malleable commitments was first constructed by Barak [Bar02], and by Pass and Rosen [PR05b] under improved assumptions. Both of these constructions were based on Barak's non-black-box simulation [Bar01]. A critical component of these protocols is a universal argument [BG08], which consists of a Merkle tree commitment to a *Probabilistically Checkable Proof* (PCP), parts of which are opened later in the protocol. Unfortunately, as shown by Ben-Sasson et al. [BCGT13], the underlying PCP proof in the universal argument can be astronomically large even for moderate parameters. To the best of our knowledge, the true efficiency of non-black-box simulation based constructions is currently not well understood.

– A third approach, due to Ostrovsky, Pandey, and Visconti [OPV10], relies on the DDH assumption, and efficiently converts any public-coin honest-verifier statistical zero-knowledge argument into a (concurrent) non-malleable one [BPS06]. While this approach uses non-malleable commitments, it avoids general-purpose proofs over them using ideas from the "simulatable commitment" of Micciancio and Petrank [MP03]. Though efficient, this transformation quickly becomes pretty slow. For example, for the standalone setting, it requires roughly $20k\lambda \log \lambda$ group exponentiations to support k bit identities at $2^{-\lambda}$ security level;[3] this is roughly 0.32 million exponentiations for $k = 32, \lambda = 80$. In addition, it requires efficient non-malleable commitments as well as efficient (and compatible) simulatable commitments, both of which are only known from DDH. Ideally, we would like to use only symmetric assumptions.

Constructions in the Random Oracle Model (ROM). The protocols we seek are straightforward to construct in the ROM [BR93] (see, e.g., [Pas03a, FKMV12]). Briefly, a random oracle (RO) is non-malleable by design, which completely sidesteps this issue. Furthermore, zero-knowledge is also trivial since the simulator and the reduction are allowed to see adversary's queries to the oracle and control the responses. In the real world, a cryptographic hash function is used to replace the RO, thus providing a concrete construction. This is an attractive methodology that often leads to practical constructions. That being so, there are several reasons to pursue constructions in the plain model, *even if efficient constructions are already known in the ROM*. We highlight some of them here.

– A protocol such as a zero-knowledge proof in the ROM can be particularly troublesome when it is used as a sub-protocol in a larger protocol. If the RO is shared by other parts of the larger protocol, the security is jeopardized since the security reduction for the sub-protocol

[2] We remark that for non-malleable commitments based on non-malleable codes such as [GPR16], it is hard to estimate the overall complexity; the asymptotic analysis of underlying codes such as [ADL14] has astronomically large constants, making them unsuitable in practice.

[3] The analysis in [OPV10] does not separate identity lengths from security levels; it further provides only asymptotic analysis which hides multiplicative constants and does not specify the exact negligible and super-logarithmic functions. This makes it difficult to assess the security level supported by their protocol. If the analysis is performed to support λ-bit security and k-bit identities, the overhead is at least $20k\lambda \log \lambda$ group exponentiations.

does not hold when a particular RO has already been selected by the larger protocol (see, e.g., [Pas03a, CDPW07, Unr07]). In addition, security proofs in this model often *program* the oracle, resulting in loss of properties such as *deniability*, which are otherwise implied by zero-knowledge (see [Pas03a, Wee09]). Deniability is a natural and useful property that has been explored in other contexts as well [CDNO97, DS98, OPW11, SW14].

– Using random oracles often sidesteps the main difficulty in achieving a particular task, such as CCA secure encryption or non-malleable commitments from standard assumption. Therefore, a construction or security proof in the ROM, while valuable, is usually not as *insightful* as its plain-model counterparts.

– Finally, while security proofs in the ROM are valuable, it requires a leap of faith to believe that instantiating the random oracle with a real world hash function maintains the claimed security. Indeed, this is not always the case [CGH98, DNRS99, Nie02, GK03]. It stands to reason that whenever possible the ROM should be avoided.

Improved constructions can be achieved in other trusted setup models as well. Di Crescenzo, Ishai, and Ostrovsky [DIO98] construct non-interactive non-malleable commitments in the CRS model, and Di Crescenzo et al. [DKOS01] do so efficiently under DDH. Lower rounds can also be achieved in the plain model under non-falsifiable assumptions [PPV08, Pas11, LPS17, KS17].

1.1 Our Results

We present a new approach for constructing *efficient* and *general-purpose* non-malleable zero-knowledge *in the plain model*. Our protocols can be viewed as a transformation which takes as input an efficient general-purpose zero-knowledge protocol, such as Ligero [AHIV17], and yields a *non-malleable* zero-knowledge protocol of (less but still) comparable efficiency. To the best of our knowledge, this is the first construction of general-purpose non-malleable zero-knowledge that achieves practical efficiency in the plain model. Our approach has the additional benefit of requiring only *symmetric* assumptions (in addition to the assumptions of the given proof system). Specifically, it suffices to assume collision-resistant hash functions.

While our primary focus is on non-malleable zero-knowledge, we also get new results for non-malleable *commitments*. Specifically, we get the first efficient construction of non-malleable commitments with large identities and message space *under symmetric assumptions*. Though this improves upon the DDH assumption required by the state-of-the-art construction [BGR+15], our construction is somewhat slower in comparison.

Even though our focus is on efficiency, our results are theoretical in nature. Our transformation makes use of non-malleable commitments in a fundamentally new way. We define and construct a new primitive called *instance-based non-malleable commitments* (IB-NMC), which admit more efficient modes than a traditional non-malleable commitment. We show how IB-NMC can be used in conjunction with the *OR-Composition* technique from [CDS94, Dam02] to obtain efficient *simulation-sound* protocols, which in turn yields efficient non-malleable protocols for both zero-knowledge and commitments. This primitive may be useful in other contexts as well.

The overhead of our transformation is within reach of practical computing. Table 1 shows the running times and communication for our non-malleable protocols for some sample parameters. A detailed analysis of the empirical results is presented in Sec. 6.

Table 1: Performance of our protocols for λ-bit security and k-bit identities. NMZK proves a witness for SHA256.

Param.	**NMZK**			**NMCom**		
(k, λ)	P time (s)	V time (s)	Comm. (MB)	P time (s)	V time (s)	Comm. (MB)
(32, 40)	1.68	0.74	19.68	2.52	1.12	19.74
(32, 80)	3.56	1.49	24.88	4.68	2.06	24.97
(64, 80)	5.04	2.23	28.84	6.72	3.09	28.93

1.2 Overview of Techniques

We start by recalling the central efficiency bottleneck in constructing non-malleable zero-knowledge for $\mathcal{NP}$. We assume that efficient *standalone* zero-knowledge (ZK) proofs already exist for all languages $L \in \mathcal{NP}$ in the plain model such as [GMO16, AHIV17]. For concreteness, we will use Ligero [AHIV17].

The main inefficiency of non-malleable zero-knowledge stems from the fact that almost all known constructions [BPS06, LPTV10, LP11a] make a non-black-box use of non-malleable commitments. More specifically, the prover commits to a witness or a trapdoor string using a non-malleable commitment and later relies on expensive $\mathcal{NP}$ reductions to prove that it either committed a valid witness *or* a trapdoor (i.e., an OR-statement); the latter is shown difficult to do for the man-in-the-middle adversary M by relying on the non-malleability of the commitment. The $\mathcal{NP}$ reduction corresponding to the OR-statement typically results in a circuit description of formidable size since the non-malleable commitment usually contains many calls to cryptographic functions such as block-ciphers. The resulting protocols are prohibitively inefficient even with state-of-the-art ZK constructions. Other approaches (based on non-black-box simulation or DDH outlined earlier) are irrelevant to our construction.

The starting point of our work is the observation that the use of non-malleable commitments in these protocols is merely *a means to an end*. In particular, the honest prover generally commits to a random or an all-zero string in these commitments; it is the simulator who makes real use of their non-malleable properties. Therefore, if we can create a situation in our protocols where the honest prover *does not have to execute even a single full non-malleable commitment*, we can improve the computational efficiency of these protocols. Let us briefly highlight why achieving this property is extremely important for our goals: As noted above, efficient non-malleable commitments in the plain model are based on DDH [OPV10, BGR+15]. One option to avoid public-key assumptions is to instantiate the scheme in [BGR+15] with Ligero; However, the running time of the resulting commitment scheme *alone* (under moderate parameters) will run in more than one minutes. The actual non-malleable zero-knowledge protocol which depends on these commitments in a non-black-box way will be much worse. We therefore seek to avoid even *one* full execution of a non-malleable commitment in our ZK protocol.

It is worthwhile to note that *black-box* constructions of non-malleable ZK from non-malleable commitments are (surprisingly) not known. The closest work in this regard is by Jain and Pandey [JP14], who construct simulation-sound ZK from a stronger version of non-malleable commitments (called *1-1 CCA* [CLP10, Kiy14]) in black-box. Currently, it is unclear if their approach can yield an efficient protocol that avoids even one execution of the non-malleable commitment.

Instance-Based Non-Malleable Commitments. Returning back to our goal of avoiding even one execution of full non-malleable commitment during the proof, we consider a new relaxation of such commitments which we call *instance-based* non-malleable commitments (IB-NMCs). Roughly speaking, an IB-NMC is just like an ordinary non-malleable commitment except that it takes as input a statement y (from an implicit $\mathcal{NP}$ language Y). The commitment has two modes: if $y \notin Y$, then it is an ordinary non-malleable commitment, and the committer commits to any desired value v by following the actual commitment algorithm C. Otherwise, if $y \in Y$, then the commitment is not guaranteed to have any non-malleability property. However, in this case, there exists a much faster algorithm C^* that, with the help of a witness for $y \in Y$, can *fake* (or simulate) an execution that looks indistinguishable from the real execution with C for any value v.

To construct IB-NMC, we combine the following key ideas:

- The simulator of a general-purpose zero-knowledge proof can be much faster than the real prover algorithm. This is best seen by considering the sub-linear zero-knowledge arguments based on PCPs [Kil92, Mic94]. In such protocols, a prover commits to a full Merkle tree over the PCP proof; but note that the simulator does not have to construct the whole tree. Instead, the simulator can simply prepare the nodes of the opened paths in a consistent manner, which is much faster. In particular, this is true for our chosen ZK system Ligero.
- The well-known OR-Composition technique developed for Σ-protocols [CDS94, Dam02] can be applied in our setting to give proofs for statements of the type "either $x \in L$ or $y \in Y$." Recall that under this technique, a prover with a witness for x constructs proofs correctly for the "$x \in L$" part, but uses the *simulator* of the Σ-protocol for the "$y \in Y$" part. Observe that if the simulator for the "$y \in Y$" part is fast (as discussed in the previous item), then the composed proofs can be almost as fast as a proof *only* for $x \in L$.
- Finally, we apply the aforementioned observations to a suitable non-malleable commitment scheme to get an efficient IB-NMC. In particular, we apply it to a modification of the BGRRV protocol [BGR+15], leading to a construction based solely on symmetric-key (or Minicrypt) assumptions (referred to as $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$). More specifically, $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ has a *commit phase* and a *proof phase* where the latter proves the "consistency" of the former. To get IB-NMC, we simply change the proof phase to prove that either the first phase is consistent or $y \in Y$ (where y is an additional input to the committer); this proof is done using the OR-composition of two Ligero protocols as described above.

We remark that this approach runs into several other issues that are not discussed here, e.g., OR-composition in general applies only to Σ-protocols but Ligero is not a Σ-protocol, the role of Y and how to choose it, etc. We will handle them in Sec. 5. The use of a honest-verifier simulator to protect against malicious attacks first appears in the work of Cramer, Damgård, and Schoenmakers [CDS94].

Non-Malleability via Simulation Soundness. While IB-NMC is an interesting primitive, it is not clear how to use it at all to construct non-malleable zero knowledge. Instead, we show that IB-NMC can be used successfully to construct a fast *simulation-sound* ZK protocol [Sah99, JP14]. Constructing this protocol requires repeated applications of the OR composition and the fake-proof technique discussed above. The simulation-sound protocol can be directly useful in larger protocols since this notion suffices for typical applications of non-malleability. Finally, we show how to use this protocol to get an efficient and full-fledged non-malleable ZK as well as an efficient non-malleable

commitment. In both cases, the transformation inherits the assumptions of the underlying zero-knowledge and IB-NMC, which in our case, are symmetric primitives only.

2 Preliminaries

We use $\lambda \in \mathbb{N}$ to denote the security parameter. Symbols $\overset{c}{\approx}$, $\overset{s}{\approx}$, and $\overset{id}{=}$ are used to denote computational, statistical, and perfect indistinguishability respectively. Let $\mathsf{negl}(\lambda)$ denote negligible functions. Familiarity with basic definitions including commitments, witness indistinguishability, zero-knowledge, arguments of knowledge, etc. is assumed; we refer to [Gol01, Gol04] for formal treatment of these notions. We also recall the definitions of CRHFs, extractable commitments, and statistically-hiding commitments in Appx. A.

Non-Malleable Interactive Proofs. We work with identity-based (or "tag-based") definitions of non-malleability and follow the definitions and conventions from [PR05b]. Let $\mathcal{A}$ be a (non-uniform) probabilistic Turing machine, specifying a man-in-the-middle strategy. $\mathcal{A}$ runs in time polynomial in the security parameter λ. Let $z \in \{0,1\}^*$ be an arbitrary string (denoting the non-uniform "advice" for $\mathcal{A}$). Let $\langle P, V, \rangle$ be an interactive proof system for an $\mathcal{NP}$ complete language L. Let $x \in L$ be a statement of length λ; we assume that P is PPT and receives a witness $w \in R_L(x)$ as its auxiliary input. The definition is based on the comparison between a *man-in-the-middle* execution and a *stand-alone* execution among the above parties.

The man-in-the-middle experiment begins by selecting uniform randomness for $\mathcal{A}$, and honest parties P and V. $\mathcal{A}(x,z)$ interacts with $P(x,w)$ on left acting as a verifier in the proof for $x \in L$; $\mathcal{A}$ simultaneously participates in a right proof with V, proving a related statement $\widetilde{x}$, supposedly in L.[4] Let the tag (or "identity") strings on left and right be id and $\widetilde{\mathsf{id}}$ respectively with $|\mathsf{id}| = |\widetilde{\mathsf{id}}| = \lambda$. We let $\mathsf{mim}_V^{\mathcal{A}}(\mathsf{id}, \widetilde{\mathsf{id}}, x, \widetilde{x}, w, z)$ be a random variable describing the output of V in the man-in-the-middle execution.

In the stand-alone execution, a machine S interacts with the honest verifier V. As in the man-in-the-middle execution, V receives as input an instance $\widetilde{x}$ and the identity $\widetilde{\mathsf{id}}$. S receives x, an auxiliary input z and id as input. We let $\mathsf{sta}_V^S(\mathsf{id}, \widetilde{\mathsf{id}}, x, \widetilde{x}, z)$ be a random variable describing the output of V in the stand-alone execution.

Definition 1 (Non-Malleable Interactive Proof). *An interactive proof $\langle P, V\rangle$ for language L is said to be non-malleable w.r.t. tags of length m if for every* PPT *man-in-the-middle adversary $\mathcal{A}$, there exists a* PPT *stand-alone prover S and a negligible function negl such that for every $x \in L$, every $w \in R_L(x)$, every $\widetilde{x} \in \{0,1\}^{|x|}$, every $\mathsf{id}, \widetilde{\mathsf{id}} \in \{0,1\}^m$ so that $\mathsf{id} \neq \widetilde{\mathsf{id}}$, and every $z \in \{0,1\}^*$, it holds that*

$$\Pr[\mathsf{mim}_V^{\mathcal{A}}(\mathsf{id}, \widetilde{\mathsf{id}}, x, \widetilde{x}, w, z) = 1] < \Pr[\mathsf{sta}_V^S(\mathsf{id}, \widetilde{\mathsf{id}}, x, \widetilde{x}, z) = 1] + \mathsf{negl}(|x|).$$

We will refer to *synchronizing* adversaries: they are the man-in-the-middle attackers who, upon receiving a message in one session, immediately respond with the corresponding message in the other session. An adversary is said to be *non-synchronizing* if it is not synchronizing.

Definition 2 (Non-Malleable Zero Knowledge). *An interactive proof between prover P and verifier V is said to be non-malleable zero knowledge if it is a non-malleable interactive proof that also has the zero-knowledge property.*

[4] We remark that statement $\widetilde{x}$ may be chosen either adaptively depending on the left execution, or statically by announcing it before the left execution begins.

Simulation Soundness. The notion of *simulation soundness* [Sah99] is a form of non-malleable ZK. Typically it is all one needs when building higher-level constructs using non-malleable ZK. In the non-interactive setting, it requires that a man-in-the-middle adversary cannot generate convincing proofs for false statements, even given access to a simulator who can generate false proofs.

The definition for the interactive setting appears in [JP14]. It requires a *single* machine S—the simulator—which guarantees indistinguishability of the view for true statements (to capture ZK), and the soundness for statements on the *right hand side* even in the presence of simulated false proofs *on the left hand side*. We use $\mathsf{MIM}^{\mathcal{A}}_{\langle P,V\rangle}(x,w,z,\mathsf{id})$ to denote the joint view of the adversary $\mathcal{A}$ in the same man-in-the-middle execution described above.

Definition 3 (Simulation-Sound Zero-Knowledge). *An interactive argument $\langle P,V\rangle$ for a language L is said to be a* simulation-sound zero-knowledge argument *if for every* PPT *man-in-the-middle algorithm $\mathcal{A}$, there exists a expected PPT algorithm S (the simulator) such that:*

- **(Indistinguishable Simulation)** *For every $x \in L$, every $w \in R_L(x)$, every $\mathsf{id} \in \{0,1\}^\lambda$, and every (auxiliary input) $z \in \{0,1\}^*$:*

$$S(x,z,\mathsf{id}) \quad \overset{c}{\approx} \quad \mathsf{MIM}^{\mathcal{A}}_{\langle P,V\rangle}(x,w,z,\mathsf{id})$$

- **(Simulation Soundness)** *There exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $x \in \{0,1\}^\lambda$, every $\mathsf{id} \in \{0,1\}^\lambda$, and every $z \in \{0,1\}^*$:*

$$\Pr\left[\nu \leftarrow S(x,z,\mathsf{id}) : \widetilde{x} \notin L \wedge \widetilde{\mathsf{id}} \neq \mathsf{id} \wedge \widetilde{b} = 1\right] \leq \mathsf{negl}(\lambda)$$

  where $\widetilde{x}, \widetilde{\mathsf{id}}$ and $\widetilde{b}$ denote the statement, identity, and verifier's decision in the right-side view of the simulated joint-view ν.

Non-Malleable Commitments. We use the tag-based definition from [LPV08, GPR16]. Specifically, we compare an ideal interaction with a real one. In the ideal interaction, a man-in-the-middle adversary $\mathcal{A}$ interacting with a committer C in the *left* session, and a receiver R in the *right*. We denote the relevant entities used in the right interaction as "tilde'd" version of the corresponding entities on the left. In particular, suppose that C commits to v in the left interaction, and $\mathcal{A}$ commits to $\widetilde{v}$ on the right. Let MIM_v denote the random variable that is the pair $(\mathsf{View}, \widetilde{v})$, consisting of the adversary's entire view of the man-in-the-middle execution as well as the value committed to by $\mathcal{A}$ on the right (assuming C commits to v on the left). The *ideal* interaction is similar, except that C commits to some arbitrary fixed value (say $0^{|v|}$, i.e. an all-zero string of length $|v|$) on the left. Let MIM_0 denote the pair $(\mathsf{View}, \widetilde{v})$ in the ideal interaction. We ensure that $\mathcal{A}$ uses a distinct identity (or "tag") $\widetilde{\mathsf{id}}$ on the right from the identity id it uses on the left. This is done by stipulating that MIM_v and MIM_0 both output a special value $\perp_{id}$ when $\mathcal{A}$ uses the same identity in both the left and right executions. Let $\mathsf{MIM}_v(z)$ and $\mathsf{MIM}_0(z)$ denote real and ideal interactions resp., when $\mathcal{A}$'s auxiliary input is z.

Definition 4 (Non-Malleable Commitments). *A tag-based statistically binding commitment scheme $\langle C,R\rangle$ is* non-malleable *if for every PPT man-in-the-middle adversary $\mathcal{A}$, and for all values $v \in \{0,1\}^\lambda$, it holds that*

$$\{\mathsf{MIM}_v(z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*} \quad \overset{c}{\approx} \quad \{\mathsf{MIM}_0(z)\}_{\lambda\in\mathbb{N},z\in\{0,1\}^*}.$$

3 Preparatory Work

In this section, we prepare the ingredients for use in the construction of our non-malleable zero-knowledge protocol. More specifically, we recall how Ligero's ZK simulator works (from [AHIV17]). Also, we show a slightly-modified version of the non-malleable commitment from [BGR+15]. We will recall related notation/techniques only to the extent that is adequate to understand our construction. For completeness, we include a more detailed review of Ligero in Appx. B.

On Notation. In [AHIV17], the authors first built a public-coin *zero-knowledge interactive PCP* (ZKIPCP) scheme. They then converted the ZKIPCP to a 6-round *honest-verifier* ZK protocol relying on Kilian's transformation [Kil92, Mic94]. Finally, they further converted it to a 7-round (fully) zero-knowledge protocol using the techniques from [IMS12, IW14][5]. Henceforth, we will use Ligero to denote their *honest-verifier* ZK protocol (shown in Prot. 8), and use Ligero′ to denote their *fully* ZK construction (shown in Prot. 9).

Simulator HVSim for Ligero. We will use the fact that simulating a Ligero (i.e., the honest-verifier version of [AHIV17]) proof is *much faster* than the real prover algorithm if the challenge of the verifier is known. The simulator's algorithm will be denoted by HVSim (HV for "honest-verifier"). There are two parts to be simulated: the first one is simulating the ZKIPCP interaction (a.k.a. the challenge-response slot); and the second one is simulating paths of the Merkle tree that are consistent with opened parts of the ZKIPCP proof string π (a.k.a. the oracle query-answer slot). The full description of HVSim is presented in Algo. 1.

Algorithm 1: HVSim: Honest-Verifier Zero-Knowledge Simulator for Ligero

Input: a statement x, a collision-resistant hash function h, and a ZKIPCP query b.

1. Run the honest-verifier simulator algorithm corresponding to the ZKIPCP system for statement x and verifier randomness (h, b) to obtain a (perfectly) simulated ZKIPCP transcript. By definition, the transcript contains simulated parts of the "proof string" π. Let $L = \{(i, \pi_i)\}$ denote these simulated parts where $i \in [|\pi|]$ denotes position in the proof. Thus, L is simply the list of opened leaves in a Merkle tree (constructed below). Note that $n = |\pi|$ is the total number of leaves and known in advance. The simulated transcript also contains the honest verfier's challenge, which is simuilated as a random string b, and the corresponding (simulated) response c.
2. Generate the paths of the Merkle tree that are consistent with $L = \{(i, \pi_i)\}$. This is straightforward, we provide the steps below for completeness:
   (a) For every element $z = (i, \pi_i)$ in L, let i' represent the index corresponding to the sibling of z in the Merkle tree (note that i' exists for every i by definition). We first check if the sibling of z exists in L by checking if any element in L contains index i'. If no sibling for z exists in L, we add a new element $z' = (i', r_i)$ into L, where r_i is a random string with length equal to the output length of h.
   (b) Let L^* be the empty set. For every z, along with its sibling z', in L, we let $z^* = h(z||z')$. We add z^* to L^*. In the end, the cardinality of L^* is equal to $|L|/2$.

[5] We remark that [AHIV17] also presented another approach—applying Fiat-Shamir transformation to their ZKIPCP will give a (fully) ZK protocol directly; moreover, the resulting protocol will be non-interactive. But this approach is irrelevant in the current paper as we are interested in constructions in the plain model (without random oracles).

(c) Set $L = L^*$ and $L^* = \emptyset$. Repeat Steps 2a to 2c while $|L| > 1$.

(d) The remaining element in L is the root of the Merkle tree.

Instantiating BGRRV with Symmetric Primitives. We will need an extractable non-malleable commitment (ENMC) that is fast and, preferably, based only on symmetric-key primitives. We work with a modified version of Brenner et al.'s protocol [BGR+15] (which is in turn based on [GRRV14]). This modified version uses Ligero' (the malicious-verifier version of Ligero) as the ZK proof system in the consistency-proof stage of the protocol. For concreteness, this instantiation is completely specified in Prot. 1. We refer to it as $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$.

Protocol 1: $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$: Extractable Non-Malleable Commitment in Minicrypt

Public Input: an identity $\mathsf{id} \in \{0,1\}^k$, a large prime q, an integer ℓ, and vector spaces $V_1, \ldots, V_n \subset \mathbb{Z}_q^\ell$ which are derived from id. These parameters satisfy the following relation: $\ell = 2(k+1)$ and $n = k+1$. (For the meanings of these parameters, we refer the readers to [BGR+15].)

Private Input: commiter C takes $\mathbf{m} \in \mathbb{Z}_q^{\ell-1}$ as its private input (i.e., the value to commit to).

Committing Stage. The committing stage consists of the following steps.

1. $R \to C$: Send the first message ρ of the Naor's commitment scheme [Nao90].
2. $C \to R$: C chooses random values $r_1, \ldots, r_n \in \mathbb{Z}_q$. This defines vectors $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbb{Z}_q^\ell$ where $\mathbf{z}_i = (r_i, \mathbf{m})$. C sends commitments $(\widehat{\mathbf{m}}, \widehat{\mathbf{r}})$ where:

$$\widehat{\mathbf{m}} = \big(\mathsf{Com}_\rho(m_1; s_1), \ldots, \mathsf{Com}_\rho(m_{\ell-1}; s_{\ell-1})\big), \quad \widehat{\mathbf{r}} = \big(\mathsf{Com}_\rho(r_1; s'_1), \ldots, \mathsf{Com}_\rho(r_n; s'_n)\big),$$

   where Com_ρ denotes the second round of Naor's commitment w.r.t. first message ρ. Note that this commits C to every coordinate of $\mathbf{z}_i$. For future reference, define the following language which contains valid commitment and message pairs:

$$L_{\mathsf{Com}_\rho} := \big\{(c, a) : \exists b \text{ s.t. } c = \mathsf{Com}_\rho(a; b)\big\}.$$

3. $R \to C$: Send random challenge vectors $\{\mathbf{v}_i\}_{i=1,\ldots,n}$ where each $\mathbf{v}_i \in V_i \subset \mathbb{Z}_q^\ell$.
4. $C \to R$: C sends evaluations $\{w_i\}$, where each $w_i = \langle \mathbf{v}_i, \mathbf{z}_i \rangle \in \mathbb{Z}_q$.

Consistency Proof. Using Ligero', C proves that the preamble was executed correctly. That is, C proves the following statement: $\exists\big((m_1, s_1), \ldots, (m_{\ell-1}, s_{\ell-1}), (r_1, s'_1), \ldots, (r_n, s'_n)\big)$ such that

- $\widehat{\mathbf{m}} = \big(\mathsf{Com}_\rho(m_1; s_1), \ldots, \mathsf{Com}_\rho(m_{\ell-1}; s_{\ell-1})\big)$, **and**
- $\widehat{\mathbf{r}} = \big(\mathsf{Com}_\rho(r_1; s'_1), \ldots, \mathsf{Com}_\rho(r_n; s'_n)\big)$, **and**
- $w_i = \langle \mathbf{z}_i, \mathbf{v}_i \rangle \ \forall i \in [n]$ where $\mathbf{z}_i = (r_i, m_1, \ldots, m_{\ell-1})$.

Notation: Henceforth, we denote the above language as L^ρ_{consis}. We say that the above **Consistency Proof** stage is proving that the statement $(\widehat{\mathbf{m}}, \widehat{\mathbf{r}}, \{w_i\}_{i\in[n]})$ is in language L^ρ_{consis}.

Observe that each message from the $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ receiver, informally speaking, is *efficiently simulatable* during "rewindings" given all prior information. That is, each message must be of one of the following three types:

1. It is a public random string;
2. It can be sampled from scratch; **or**
3. It is simply a complete opening of a previous commitment (and thus repeatable in rewind threads if needed).

This observation will play an important role later when we prove the non-malleability of our ZK protocol (more specifically, when proving Claim 1). But we also emphasize that this observation is crucial *only* in the non-synchronous setting (but not in the synchronous setting).

Extractability of BGRRV. We remark that BGRRV is an extractable commitment scheme. Extraction can be performed from the preamble stage by simply rewinding to the second message, obtaining a valid answer for a different challenge, and then solving two equations in $\mathbb{Z}_p$.

4 Our Non-Malleable Zero Knowledge Protocol

In this section, we present the generic framework of our non-malleable zero-knowledge. Later in Sec. 5, we will instantiate each component of this protocol in special ways so that the final protocol admits an efficient implementation using only symmetric-key primitives. We use the following ingredients:

1. An extractable commitment scheme ExtCom. We will use the standard 3-round scheme (see Prot. 7, Appx. A.2). Note that the first committer message of this scheme is statistically-binding.
2. A tag-based commitment scheme ENMC that is both *non-malleable* and *extractable*; for concreteness, we will use scheme $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ specified in Prot. 1. We assume for convenience that the commitments are generated using Naor's scheme [Nao90] w.r.t. an implicit first string ρ chosen by the receiver of the commitment (and dropped from the notation henceforth).
   We assume that the first *committer* message of ENMC is statistically binding. For concreteness, we say that a string c is an **honest** ENMC **commitment to a value** v **with tag** id if there exists randomness r such that c is the first committer message of ENMC produced by the honest committer algorithm on input value v, tag id, and randomness r.
3. A *statistically* witness-indistinguishable argument of knowledge sWIAoK.

Our construction is shown in Prot. 2 below. At a high level, the protocol is as follows: V starts by committing to a random string σ. P then uses an extractable non-malleable commitment ENMC to commit to an all-zero string. Then V decommits to its commitment made at the beginning of the protocol. Finally, P and V execute a sWIAoK protocol, where P proves to V that *either* it knows a witness to x, *or* that the commitment in ENMC equals σ.

Protocol 2: $\langle P, V\rangle_{\mathsf{NMZK}}$**: Non-Malleable Zero-Knowledge**
Public input: Security parameter λ, statement x (supposedly in an $\mathcal{NP}$ language L), and a tag $\mathsf{id} \in \{0,1\}^{\leq\lambda}$. **Private input:** P takes the witness w as its private input.

1. V commits to a random string $\sigma \leftarrow \{0,1\}^\lambda$, using the extractable commitment scheme ExtCom. We denote the first committer message by com_1.
2. P commits to $\sigma' = 0^\lambda$ using the extractable non-malleable commitment ENMC with tag id. We denote the first committer message of this stage by com_2.
3. V sends σ along with decommitment information for com_1.
4. If Step 3 decommitment is valid, P proves the following compound statement to V using a *statistical* witness-indistinguishable argument of knowledge sWIAoK:
   - there exists a w such that $R(x,w) = 1$; **or**
   - com_2 is an honest ENMC commitment to σ with tag id.

For future reference (σ', r) is called the trapdoor witness for statement $(\mathsf{com}_2, \mathsf{id})$ if r is s.t. com_2 is the 1st committer message of ENMC on input σ', tag id, and randomness r.

Theorem 1. *The protocol $\langle P, V\rangle_{\mathsf{NMZK}}$ (shown in Prot. 2) is a non-malleable zero-knowledge argument of knowledge for $\mathcal{NP}$.*

Thm. 1 can be established by combining Lem. 1 and Lem. 2, which we prove in the following.

Lemma 1. *$\langle P, V\rangle_{\mathsf{NMZK}}$ is a zero-knowledge argument of knowledge.*

Proof. The completeness of our protocols follows from the completeness of the sWIAoK in Step 4. The description of the ZK simulator and the knowledge extractor (for AoK property) will be implicit in our proof of non-malleability that follows. Therefore, here we only provide a sketch of them. Also, note that soundness follows from the AoK property.

Zero-Knowledge Property. The simulator Sim for the ZK property is constructed as follows: Sim first extracts the value σ from $\mathsf{ExtCom}(\sigma)$, in expected polynomial time, by rewinding the (cheating) verifier V^*. It then sets $\sigma' = \sigma$ in the Step 2 ENMC commitment, and uses this condition as a trapdoor witness to succeed in Step 4 sWIAoK. The indistinguishability of the simulated and real views relies on the hiding property of ENMC and the witness indistinguishability (WI) of sWIAoK. The proof of non-malleability given later contains this argument implicitly in full detail.

Argument-of-Knowledge Property. Let us first prove that no PPT machine P^* can succeed in setting $\sigma' = \sigma$, except with negligible probability. If not, we can build an adversary $\mathcal{A}_{\mathsf{h}}$ to break the hiding property of ExtCom as follows. $\mathcal{A}_{\mathsf{h}}$ incorporates P^* internally; it gets $\mathsf{com}_1 = \mathsf{ExtCom}(\sigma)$ from an external committer and forwards it to $P^*(x)$ as the Step 1 message of $\langle P^*, V\rangle_{\mathsf{NMZK}}$. Let S^* denote the machine $P^*(x)$ at the end of Step 1; S^* proceeds exactly as P^* but halts at the end of ENMC in Step 2. Using the standard averaging argument, if η is the success probability of P^* then with probability at least $\eta/2$, machine S^* can successfully complete the remaining execution with probability at least $\eta/2$. We call such machines S^* to be good. Now, we apply the extractor guaranteed by the extractability property of ENMC, to produce a tuple (τ, σ^*) in expected polynomial time so that τ is distributed identically to a real execution with honest receiver. It follows that τ is convincing with probability at least $\eta/2$, and whenever that happens, we have that $\sigma' = \sigma$ (due to identical distributions) and $\sigma^* = \sigma'$ (due to correctness of extraction). Note that over-extraction is not an issue here since the condition $\sigma = \sigma'$ is well defined only when $\sigma' \neq \perp$. It

follows that $\mathcal{A}_h$ contradicts the hiding property of ExtCom with probability at least $\eta^2/4 - \mathsf{negl}(\lambda)$. We now observe that the AoK property follows from the AoK property of the Step 4 sWIAoK. □

Lemma 2. *$\langle P, V\rangle_{\mathsf{NMZK}}$ is non-malleable.*

We prove Lem. 2 in subsequent subsections. We first present in Sec. 4.1 the proof regarding synchronous adversaries (who send their right messages as soon as they receive the corresponding left message). Then, we deal with the general case of non-synchronous adversaries in Sec. 4.2.

When reading the proofs in the synchronous setting, it would be helpful to keep in mind also the non-synchronous case. We add remarks at the end of each hybrid to address this. We hope it can improve the readability when we talk about the non-synchronous setting later.

4.1 Non-Malleability against Synchronous Adversaries

To prove non-malleability, we need to build a simulator which can convince V with roughly the same probability as a man-in-the-middle adversary $\mathcal{A}_{\mathsf{mim}}$ (up to some negligible difference), but without the help of the left interaction. We first define the following invariant condition.

Definition 5 (Invariant Condition). *The probability that the value $\widetilde{\sigma}'$ committed in com_2 by $\mathcal{A}_{\mathsf{mim}}$ is equal to $\widetilde{\sigma}$ committed in com_1 by the honest verifier is negligible.*

Note that if the invariant condition holds and $\mathcal{A}_{\mathsf{mim}}$ gives a convincing proof, we can extract the witness $\widetilde{w}$ for $\widetilde{x}$ by running the sWIAoK extractor.

At a high level, our proof goes in the following way. We start with the man-in-the-middle setting, where an honest prover $P(x, w)$ interacts with $\mathcal{A}_{\mathsf{mim}}$ in the left interaction, and $\mathcal{A}_{\mathsf{mim}}$ proves to an honest verifier V for a statement $\widetilde{x} \neq x$ in the right. We will build a sequence of hybrids, where we gradually substitute $P(x, w)$ and $V(\widetilde{x})$ with our simulator. Between each pair of adjacent hybrids, we show that the view of $\mathcal{A}_{\mathsf{mim}}$ does not change *and* that the invariant condition holds. In the last hybrid, we do not need the real witness w in the left interaction, and we can extract $\mathcal{A}_{\mathsf{mim}}$'s witness $\widetilde{w}$ via the sWIAoK extractor (we are guaranteed to extract $\widetilde{w}$ because of the invariant condition). With the extracted $\widetilde{w}$, our simulator can give a "straight-line" proof for the statement $\widetilde{x}$ to V, which completes the proof of non-malleability. Next, we describe the hybrids.

Hybrid H_0. This is the real execution of the MIM game. Specifically, H_0 sets up the left and right executions for $\mathcal{A}_{\mathsf{mim}}$ with $P(x, w)$ and V, respectively. H_0 outputs the joint view of $\mathcal{A}_{\mathsf{mim}}$ containing both left and right executions.

Invariant condition. If the invariant condition does not hold, then consider the prover machine P^* which behaves identically to H_0 except that it forwards the right ExtCom to an external committer. Using this P^* we can violate the hiding of ExtCom by extracting the value committed in the right ENMC.

Hybrid H_1. This hybrid is identical to H_0, except that whenever the left ExtCom is accepting, H_1 extracts the committed value σ in the left ExtCom. If the extractor fails ($\sigma = \perp$), H_1 outputs $\perp$ and halts; otherwise it continues as H_0.

$H_0 \overset{s}{\approx} H_1$. The outputs of H_0 and H_1 differ only when $\sigma = \perp$; and due to the extractability of ExtCom (Def. 7), that happens with only negligible probability.

Invariant condition. The invariant condition holds in H_1 since it holds in H_0 and the two hybrids are statistically close.

Remark 1. Note that the above proofs for both indistinguishability and invariant condition are independent of $\mathcal{A}_{\mathsf{mim}}$'s scheduling of the messages. Thus, they also hold in the non-synchronous scenario.

Hybrid H_2. This hybrid is identical to H_1, except that H_2 sets $\sigma' = \sigma$ in **Stage-2** ENMC on left.

$H_1 \overset{\text{c}}{\approx} H_2$ follows immediately from the computational-hiding property of ENMC.

Invariant condition. The fact that the invariant condition holds can be reduced to the *non-malleability* of ENMC. Specifically, we consider a man-in-the-middle adversary $\mathcal{A}_{\mathsf{ENMC}}$ for ENMC that acts as follows: $\mathcal{A}_{\mathsf{ENMC}}$ internally runs H_2 except that it obtains the left ENMC execution from an outsider committer on the left and forwards the right ENMC interaction to an external receiver. Furthermore, the external committer commits as follows: recall that H_2 already has the extracted value σ before the left ENMC begins; $\mathcal{A}_{\mathsf{ENMC}}$ forwards $\sigma'_0 = 0^\lambda$ and $\sigma'_1 = \sigma$ to the external committer who then commits to one of them at random. $\mathcal{A}_{\mathsf{ENMC}}$ halts when H_2 halts. Now consider a distinguisher D (that incorporates the above adversary $\mathcal{A}_{\mathsf{ENMC}}$), and by definition of non-malleability, receives the value $\mathcal{A}_{\mathsf{ENMC}}$ commits to in the right interaction, say $\widetilde{\sigma}$. Clearly, if the invariant condition does not hold in H_2 then the distribution of $\widetilde{\sigma}$ is different depending on whether $\mathcal{A}_{\mathsf{ENMC}}$ receives commitment to σ'_0 or σ'_1. This condition can be tested by D (which incorporates $\mathcal{A}_{\mathsf{ENMC}}$), thus violating the non-malleability of ENMC.

Remark 2. Observe that in the non-synchronous case, the proof of indistinguishability will go through, but the proof of invariant condition will not. This is because the extraction of α on left from ExtCom may rewind some parts of ENMC on right, and this is not allowed by the non-malleability definition. We will deal with this issue in Sec. 4.2.2.

Hybrid H_3. Identical to H_2 except that it switches from real witness w to the trapdoor witness (i.e., values and randomness corresponding to $\sigma' = \sigma$) in the **Stage-4** sWIAoK on left.

$H_3 \overset{\text{s}}{\approx} H_2$ follows directly from the statistical WI property of sWIAoK.

Invariant condition. Since we are in the synchronous setting, the invariant condition holds since the executions in the two hybrids are identical up to the end of **Stage-2**, at which point the invariant condition is already determined; any changes after that stage have no effect on the invariant condition.

Remark 3. As in Rmk. 2, in the non-synchronous case, the argument for indistinguishability still holds, but the argument for the invariant condition will require extra caution. This is because the left sWIAoK may get aligned with the right ENMC so that the switch of witness may affect the invariant condition. We will deal with this issue in Sec. 4.2.1.

Simulator for Non-Malleability. The indistinguishability among the above hybrids implies that: if $\mathcal{A}_{\mathsf{mim}}$ gives a convincing proof in the right interaction of H_0, it should also give a convincing proof in the right interaction of H_3. We construct a simulator Sim in the following way. Given a man-in-the-middle adversary $\mathcal{A}_{\mathsf{mim}}$, Sim first invokes H_3 with $\mathcal{A}_{\mathsf{mim}}$. If $\mathcal{A}_{\mathsf{mim}}$ indeed gives a convincing proof in the right interaction, Sim extracts $\mathcal{A}_{\mathsf{mim}}$'s witness $\widetilde{w}$ from sWIAoK on the right execution; otherwise, Sim aborts. The invariant condition in H_3 guarantees that Sim can extract such a $\widetilde{w}$. With

$\widetilde{w}$, Sim then executes protocol $\langle P, V\rangle_{\mathsf{NMZK}}$ (in "straight-line") with an honest verifier. It convinces the honest verifier with roughly the same probability as $\mathcal{A}_{\mathsf{mim}}$ (except for negligible difference due to Sim's failure in extracting $\widetilde{w}$). This finishes the proof of non-malleability *against synchronous adversaries*.

4.2 Non-Malleability against Non-Synchronous Adversaries

As mentioned in Rmk. 1 to 3, the proofs for indistinguishability among *all* hybrids, as well as the invariant condition for H_0 and H_1, remain unchanged in the non-synchronous setting. Therefore, we only need to prove the invariant conditions for H_2 and H_3, which will be done in Sec. 4.2.1 and 4.2.2. (We first show the proof for H_3 since it is simpler.)

4.2.1 The Invariant Condition for H_3

Recall that the witness indistinguishability of the sWIAoK is *statistical*. It follows that the invariant condition must hold in H_3 for non-synchronous adversaries as well. If not, an exponential time distinguisher can recover the value committed by $\mathcal{A}_{\mathsf{mim}}$, thus breaks the statistical WI by testing whether the invariant condition.

4.2.2 The Invariant Condition for H_2

Before giving the formal lemma and proof, we provide the high-level idea. As mentioned in Rmk. 2, the problem happens if the $\mathcal{A}_{\mathsf{mim}}$ interleaves the left ExtCom messages with the right ENMC messages. In such a schedule, we cannot reduce the invariant condition to the non-malleability of ENMC without rewinding the outside challenger in ENMC's man-in-the-middle game. Recall that both H_1 and H_2 rewind the left ExtCom to extract the committed value σ.

We first note that if the reduction can simulate the receiver-to-committer messages in ENMC, then there is no issue during rewinding since in the right interaction, the reduction can forward messages between $\mathcal{A}_{\mathsf{mim}}$ and the outside challenger to the "main thread" and simply simulate them in "rewinding" threads. This (informally-explained) property is indeed satisfied by our $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ commitment (Prot. 1).

In the following, we show the formal claim and its proof.

Claim 1. *The invariant condition holds in Hybrid H_2 described in Sec. 4.1 for non-synchronous adversaries.*

Proof. This proof relies on the special structure of ENMC (when instantiated as the $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ protocol shown in Prot. 1). We will refer to different rounds of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$, which are recalled below for convenience (see and compare with Prot. 1):

- **(1):** R sends the first message for Naor's commitment, which consists of public coins only.
- **(2):** C sends the second message of Naor's commitment.
- **(3):** R sends some (public) random vectors as his challenge.
- **(4):** C responds to R's challenges. C also sends the first message (which consists of some public coins that specifies a CRHF) of a Ligero' instance (depicted in Prot. 9), which is used for consistency proof.

– **(5)-(10):** These are Rounds 2 to 7 of Ligero' between C and R. Note that **(5)** is the (statistically-hiding) commitments to verifier's random challenge Γ_1 and Γ_2; **(7)** is R's decommitment to Γ_1 and **(9)** is R's decommitment to Γ_2.

With the structure of ENMC in mind, we now start to prove Claim 1. First, observe that ExtCom has only one "slot" that needs to be rewound to extract σ. Therefore, we only need to worry about the schedule where some messages of the right ENMC are "nested" in this slot. In the following, we show that the invariant condition hold for all schedules.

In the following, we use (i) ($i \in [10]$) to denote the i-th step of the right ENMC (as recalled above). We denote the first message of the rewindable slot in the left ExtCom as top, and the last message as bottom. See Fig. 1 for an illustration of these notations. Note that in Fig. 1, no messages can appear between "adjacent messages" of the right ENMC, for example, message **(2)**-**(3)**, **(6)**-**(7)** etc. This is because honest parties send their next message as soon as they receive the previous message.

Easy Cases. First, note that if bottom happens before **(1)**, we can rewind the slot without rewinding the right ENMC. Therefore, the same proof for the invariant condition in H_2 in the synchronous setting also applies here. Also, it is an easy case when **(10)** happens before bottom. In this case, $\mathcal{A}_{\mathsf{mim}}$ cannot generate the right ENMC messages based on the left ENMC interactions, since the left ENMC has not started yet. Therefore, the invariant condition holds automatically. Another easy case is when **(1)** gets nested in the slot. In such a case, rewinding the slot will cause a fresh execution of the right ENMC, so it will not cause any problem when we try to reduce the invariant condition to the non-malleability of ENMC. At a high level, this is because we can always forward the messages when we do the last rewinding to the outside non-malleability challenger in the reduction. But we suppress the details here since we will provide a formal argument of such type when we handle the hard cases next.

Hard Cases. We now focus on the remaining schedules (beyond those discussed in **Easy cases**). These schedules consist of the situations where

– **(1)** happens before top, *and*

– **(10)** happens after bottom.

There are 10 such cases in total: one of them is shown in Fig. 1; the other 9 are shown in Fig. 2. Since these 10 schedules can be handled via similar arguments, in the following, we will first use the one in Fig. 1 as a representative to present a full proof, and then discuss how to extend the same proof to the remaining 9 cases.

The Hard Case in Fig. 1. For the schedule shown in Fig. 1, we build a man-in-the-middle adversary $\mathcal{A}_{\mathsf{ENMC}}$ attacking the non-malleability of ENMC. Recall that in the non-malleability game, the man-in-the-middle adversary $\mathcal{A}_{\mathsf{ENMC}}$ talks to an honest committer in the left, and to an honest receiver in the right. We will refer to them as the left challenger and right challenger respectively. Our $\mathcal{A}_{\mathsf{ENMC}}$ acts in the following way:

1. $\mathcal{A}_{\mathsf{ENMC}}$ starts by running the hybrid experiment H_2 *internally* with $\mathcal{A}_{\mathsf{mim}}$ up to the step right before **(1)**. It then invokes the right challenger for the non-malleability game of ENMC, and forwards the messages between the challenger and $\mathcal{A}_{\mathsf{mim}}$ as the right interaction. It plays the left interaction in the same way as the simulator in H_2, until the execution reaches top for the first time.

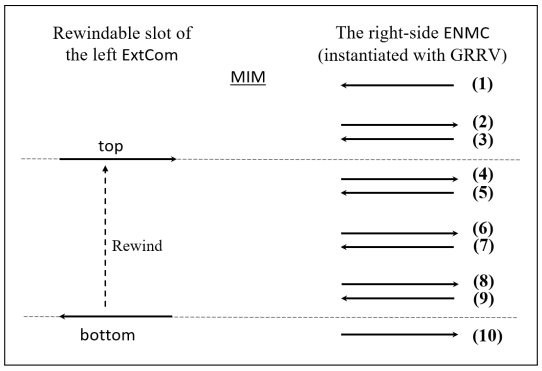

Fig. 1: Special Schedules in the Non-Synchronous Scenario

2. $\mathcal{A}_{\mathsf{ENMC}}$ now needs to execute the slot $(\mathsf{top}, \mathsf{bottom})$ in the "main-thread", and then rewind this slot for (w.l.o.g.) $k = \mathsf{poly}(\lambda)$ times to extract the σ value in the left interaction. To do that, $\mathcal{A}_{\mathsf{ENMC}}$ proceeds as follows:

   (a) For the main-thread execution, $\mathcal{A}_{\mathsf{ENMC}}$ plays the right interaction by forwarding messages between $\mathcal{A}_{\mathsf{mim}}$ and the outside right challenger.

   (b) From the 1st to the k-th rewinding, $\mathcal{A}_{\mathsf{ENMC}}$ will prepare the right ENMC incoming messages (i.e. **(5)**, **(7)**, and **(9)**) *by himself*, instead of forwarding them between $\mathcal{A}_{\mathsf{mim}}$ and the outside right challenger. To do that, $\mathcal{A}_{\mathsf{ENMC}}$ samples *fresh* Γ_1 and Γ_2, and commits to them as message **(5)**; it sends the honest decommitments to (the fresh) Γ_1 as message **(7)**; similarly, it sends the honest decommitments to (the fresh) Γ_2 as message **(9)**. We emphasize that $\mathcal{A}_{\mathsf{ENMC}}$ can indeed decommit to them because the commitments in **(5)** (in these rewinding threads) are generated by himself.

   Note that the simulated messages during rewinding have identical distribution as the main-thread **(5)**, **(7)**, and **(9)**, which guarantees that $\mathcal{A}_{\mathsf{mim}}$'s view does not change. Thus, after the above rewindings, σ can be extracted except for negligible probability, for which $\mathcal{A}_{\mathsf{ENMC}}$ just halts outputting $\perp$.

3. $\mathcal{A}_{\mathsf{ENMC}}$ continues the internal (main-thread) interaction until the left ENMC starts. He then invokes the outside left challenger by sending the values $\sigma'_0 = 0^\lambda$ and $\sigma'_1 = \sigma$. Then, ENMC forwards the messages between $\mathcal{A}_{\mathsf{mim}}$ and the outside left challenger and $\mathcal{A}_{\mathsf{mim}}$ as the left ENMC interaction. In the right interaction, ENMC acts as the simulator in H_2 except that when $\mathcal{A}_{\mathsf{mim}}$ sends the message **(10)**, it forwards the message to the outside right challenger.

4. $\mathcal{A}_{\mathsf{ENMC}}$ continues to finish the internal interaction with $\mathcal{A}_{\mathsf{mim}}$ as in H_2 for the remaining parts of the protocol.

Now consider a distinguisher D (that incorporates the above adversary $\mathcal{A}_{\mathsf{ENMC}}$), and by definition of non-malleability, receives the value $\mathcal{A}_{\mathsf{ENMC}}$ commits to in the right interaction, say $\widetilde{\sigma}$. Clearly, if the invariant condition does not hold in H_2 then the distribution of $\widetilde{\sigma}$ is different depending on whether $\mathcal{A}_{\mathsf{ENMC}}$ receives commitment to σ'_0 or σ'_1. This condition can be easily tested by D (since it incorporates $\mathcal{A}_{\mathsf{ENMC}}$), thus violating the non-malleability of ENMC.

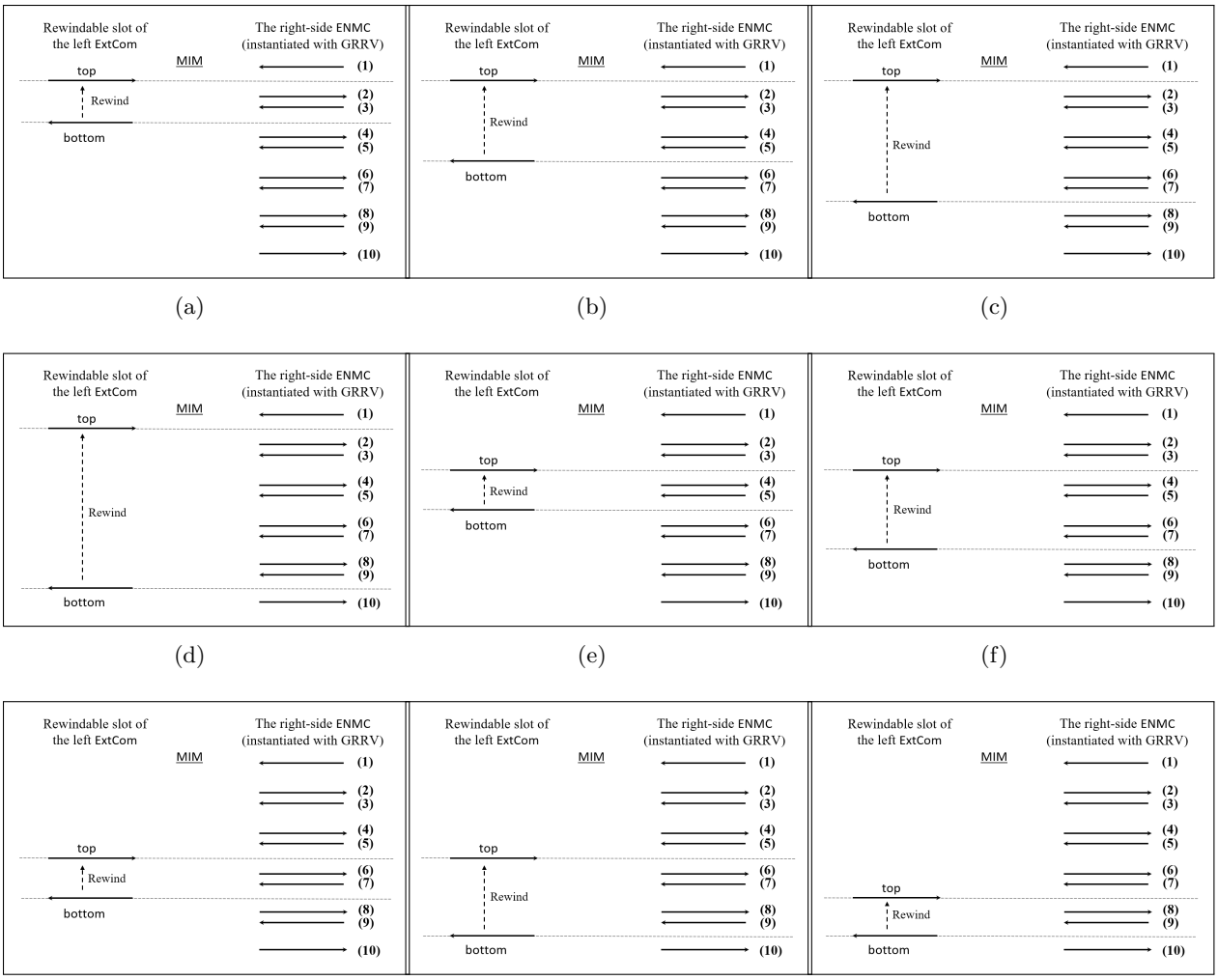

Fig. 2: The Other 9 Schedules in the Non-Synchronizing Scenario

Other Hard Cases (Fig. 2). The above argument proves Claim 1 for the special scheduling shown in Fig. 1. We now discuss the other 9 non-synchronous schedules for the **Hard cases** when proving the invariant condition in **Hybrid 2**. These schedules are shown in Fig. 2.

The analysis for Fig. 1 extends to all the schedules depicted in Fig. 2. To do that, we only need to modify Step 2b in the description of $\mathcal{A}_{\mathsf{ENMC}}$, according to the target schedule. Namely, $\mathcal{A}_{\mathsf{ENMC}}$ in Step 2b needs to prepare (or "simulate") the incoming messages nested in slot $(\mathsf{top}, \mathsf{bottom})$, for the k rewinding executions, without the help of the outside right challenger. In the following, we list the simulation strategies for all the cases shown in Fig. 2.

– Fig. 2a: **(3)** are some random vectors. In each rewinding, $\mathcal{A}_{\mathsf{ENMC}}$ can sample these values by himself, and send them to $\mathcal{A}_{\mathsf{mim}}$ as the simulated **(3)**.

– Fig. 2b: The simulation for **(3)** can be done in the same way as for Fig. 2a. The simulation for **(5)** can be done in the same way as for Fig. 1 discussed in Sec. 4.2.2. I.e., $\mathcal{A}_{\mathsf{ENMC}}$ "fakes" message **(5)** by sampling *fresh* Γ_1 and Γ_2, and committing to them by himself.

– Fig. 2c: The simulation for **(3)** can be done in the same way as for Fig. 2a. The simulation for **(5)** and **(7)** can be done in the same way as for Fig. 1 discussed in Sec. 4.2.2. I.e., $\mathcal{A}_{\mathsf{ENMC}}$ "fakes" message **(5)** by sampling *fresh* Γ_1 and Γ_2, and committing to them by himself; it "fakes" message **(7)** by decommitting to (the fresh) Γ_1 honestly.

– Fig. 2d: The simulation for **(3)** can be done in the same way as for Fig. 2a. The simulation for **(5)**, **(7)**, and **(9)** can be done in the same way as for Fig. 1 discussed in Sec. 4.2.2. I.e., $\mathcal{A}_{\mathsf{ENMC}}$ "fakes" message **(5)** by sampling *fresh* Γ_1 and Γ_2, and committing to them by himself; it "fakes" message **(7)** by decommitting to (the fresh) Γ_1 honestly; it "fakes" message **(9)** by decommitting to (the fresh) Γ_2 honestly.

– Fig. 2e: $\mathcal{A}_{\mathsf{ENMC}}$ "fakes" message **(5)** by sampling *fresh* Γ_1 and Γ_2, and committing to them by himself.

– Fig. 2f: $\mathcal{A}_{\mathsf{ENMC}}$ "fakes" message **(5)** by sampling *fresh* Γ_1 and Γ_2, and committing to them by himself; it "fakes" message **(7)** by decommitting to (the fresh) Γ_1 honestly.

– Fig. 2g: Note that message **(7)** is the decommitments to the Γ_1 committed in message **(5)**, which happens (and is fixed) before top. Thus, $\mathcal{A}_{\mathsf{ENMC}}$ can simply reuse the decommitments messages obtained from the main-thread execution in all the k rewindings.

– Fig. 2h: Note that message **(7)** (resp. **(9)**) is the decommitments to the Γ_1 (resp. Γ_2) committed in message **(5)**, which happens (and is fixed) before top. Thus, $\mathcal{A}_{\mathsf{ENMC}}$ can simply reuse the decommitments messages obtained from the main-thread execution in all the k rewindings.

– Fig. 2i: Note that message **(9)** is the decommitments to the Γ_2 committed in message **(5)**, which happens (and is fixed) before top. Thus, $\mathcal{A}_{\mathsf{ENMC}}$ can simply reuse the decommitments messages obtained from the main-thread execution in all the k rewindings.

Observe that, in all of the above cases, the simulated messages are always identical distributed as in the corresponding main-thread execution. Thus, we are guaranteed that the correct σ value can be extracted from the left ExtCom. Therefore, the same argument for for Fig. 1 (shown in Sec. 4.2.2) can be applied to reduce the invariant condition to the non-malleability of ENMC. This eventually finishes our proof for Claim 1. □

4.2.3 Generalization to "Almost Public-Coin" Statistically ZK

In this part, we take another look at the proof in Sec. 4.2.2 with the following purpose: in Sec. 4.2.2, we proved the invariant condition in H_2, relying on the special structure of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$. In particular, we assumed that the **Consistency Proof** stage of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ is conducted by Ligero'. However, we argue that Ligero' can be replaced by any "almost public-coin" (explained below) statistically zero-knowledge argument.

Motivation. Before delving into the details, let us first explain why we want to generalize the proof to almost public-coin ZK protocols: While Ligero' is efficient, using it directly in the **Consistency Proof** stage of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ results in unacceptable running time. This is because the language $L^{\rho}_{\mathsf{consis}}$ (defined toward the end of Prot. 1) has a huge circuit size. As mentioned in Sec. 1.2, we

will (in Sec. 5.3) introduce the new idea of converting $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ to an *instance-based* non-malleable commitment to achieve better efficiency. Looking ahead, the instance-based $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ shares the same structure of the original $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$, with the only difference being that the **Consistency Proof** stage is not conducted by Ligero' anymore. Instead, it will be done using a customized statistical ZK protocol called Π'_{OR}, which we construct by applying (a modified version of) the OR-composition technique [CDS94] on Ligero (i.e., the honest-verifier version of Ligero'). We need to show that the same proof in Sec. 4.2.2 will still go through[6] when we replace (the original) $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ with this instance-based $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ (i.e., when we replace Ligero' in the **Consistency Proof** stage with Π'_{OR}). Fortunately, this is possible because Π'_{OR} shares the same structure as Ligero', in terms of the application in Sec. 4.2.2. In particular, Π'_{OR} also enjoys the same "almost public-coin" property of Ligero', and this is exactly why the same proof in Sec. 4.2.2 can be applied when we replace Ligero' with Π'_{OR}. The purpose of this subsection is to distill this "almost public-coin" property and explain how it helps in the proof in Sec. 4.2.2.

Almost Public-Coin Protocols. Let us summarize how the proof in Sec. 4.2.2 makes use of the structure of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$. As we mentioned in the beginning of Sec. 4.2, $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ has 10 rounds that can be understood as two stages:

1. **Commit Stage:** This includes rounds **(1)** to **(4)**; and
2. **Consistency Proof:** This includes rounds **(4)** to **(10)**, which are exactly the statistically ZK protocol Ligero'.

We emphasize that all the receiver's messages are public random coins except for rounds **(5)** and **(7)**, which constitute the commitment and corresponding decommitment to some random coins. This public-coin property is the main reason that $\mathcal{A}_{\mathsf{ENMC}}$ works properly: in Step 2, $\mathcal{A}_{\mathsf{ENMC}}$ needs to simulate the receiver's message in rewinding threads; because all the receiver's messages (except for rounds **(5)** and **(7)**) are public-coin, $\mathcal{A}_{\mathsf{ENMC}}$ can simply sample them freshly for each rewinding; moreover, round **(5)** (resp. **(7)**) is a commitment (resp. the corresponding decommitment) to random coins, so $\mathcal{A}_{\mathsf{ENMC}}$ can also sample and commit to (resp. decommit honestly to) random coins itself. Therefore, the rewinding threads can be shown to be identically distributed as the main thread.

In light of the above, it is clear that the $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ can be replaced with any ENMC that enjoys the above public-coin property. In particular, the **Commit Stage** of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ is public-coin by design; the **Consistency Proof** stage, when implemented with Ligero', is public-coin (again, except for **(5)** and **(7)** as discussed above) because Ligero' is obtained in a special way: it is obtained by applying the Goldreich-Kahan transform on the honest-verifier version Ligero, which is a public-coin protocol.

Looking ahead, our Π'_{OR} enjoys the above public-coin property. As we will show in Sec. 5.3, Π'_{OR} is obtained by applying Goldreich-Kahan transform on a protocol Π_{OR} (which will appear in Sec. 5.2), which is also a public-coin honest-verifier ZK argument. Therefore, the above argument applies.

In summary, when we replace $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ with its instance-based version, the same proof in Sec. 4.2.2 will still go through.

[6] We do not need to revisit the proof in Sec. 4.2.1, as it is independent of the structure of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$.

5 Improving Efficiency through Fake Executions

5.1 Road Map

In this section, we describe how to instantiate our NMZK protocol $\langle P, V\rangle_{\mathsf{NMZK}}$ (shown in Prot. 2) to achieve concrete efficiency. The major bottlenecks are:

1. Step 4 of $\langle P, V\rangle_{\mathsf{NMZK}}$ is a statistical WIAoK on the OR-composition of the statement x and a trapdoor statement (let us denote it as $(x \vee x_{\mathsf{tr}})$). This proof is non-black-box on the Step 2 commitments and involves expensive $\mathcal{NP}$ reduction.
2. Step 2 of $\langle P, V\rangle_{\mathsf{NMZK}}$ is instantiated with $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ (Prot. 1), whose **Consistency Proof** step involves an expensive ZK proof.

To address Item 1, we want to employ the OR-composition technique in [CDS94] to construct the desired sWIAoK from Ligero. This will allow the prover to finish the proof for $(x \vee x_{\mathsf{tr}})$ by conducting a (light) proof for x, and running the fast Ligero simulator HVSim for the x_{tr} part. This will be much more efficient than running Ligero on $(x \vee x_{\mathsf{tr}})$ directly. However, this approach encounters obstacles: Ligero does not have the properties required by [CDS94]. We show how to solve related problems in Sec. 5.2.

To address Item 2, we wish to reuse the OR-composition technique described above. But it does not immediately apply because the target statement of the **Consistency Proof** does not have the $(x \vee x_{\mathsf{tr}})$ structure; instead, it is a single statement $x_{\mathsf{com}} \in L^{\rho}_{\mathsf{consis}}$, which is related to some vector of commitments[7]. Running Ligero for x_{com} is prohibitively expensive. To handle this issue, observe that this $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ protocol is executed as a part of our $\langle P, V\rangle_{\mathsf{NMZK}}$ protocol on some statement x_{zk}. Therefore, we change the statement of **Consistency Proof** to $(x_{\mathsf{zk}} \vee x_{\mathsf{com}})$, and then use the above OR-composition technique to boost the efficiency. We denote this extended non-malleable commitments as *instance-based non-malleable commitments* (IB-NMC). We elaborate on the above idea in Sec. 5.3.

Non-Malleability from Simulation-Soundness. Unfortunately, the above strategy induces an extra problem—replacing the Step 2 ENMC by the above instance-based version (i.e. the IB-NMC) jeopardizes the security of $\langle P, V\rangle_{\mathsf{NMZK}}$ (Prot. 2). Specifically, it is not clear whether the resulting protocol is still non-malleable. However, we will be able to prove that it is a *simulation-sound* ZK protocol (which is already sufficient for many applications). Finally, we show in Sec. 5.4 (resp. Sec. 5.5) how to use this simulation-sound ZK protocol to obtain non-malleable ZK protocols (resp. non-malleable commitments), with (almost) no efficiency overhead.

5.2 OR-Composition of Ligero

The OR-composition [CDS94] was originally designed for Σ protocols, i.e., 3-round public-coin HVZK protocols with *special soundness*, which requires that a witness can be extracted from two convincing transcripts with distinct challenges. To prove an OR statement $x \vee x'$, the OR-composition invokes a parallel execution of two Σ-protocol instances: (a_1, b_1, c_1) for proving x and (a_2, b_2, c_2) for proving x', which are called the left and right execution respectively. But the verifier sends only a single round-2 challenge b; the prover has the freedom to "decompose" it as $b = b_1 \oplus b_2$ to finish the two parallel executions. The prover may only have a witness for, say, the x part; since it

[7] Recall that the language $L^{\rho}_{\mathsf{consis}}$ is defined toward the end of $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$ (Prot. 1).

can always "equivocate" one share of b, it will first "finish" (in other words, fake) the left execution by running the HVZK simulator for the Σ-protocol by setting b_2 in advance; it can answer any $b_1 = b \oplus b_2$ as it has the witness for x.

The First Attempt. We want to apply the above OR-composition to Ligero. However, Ligero is not a Σ-protocol—it has six rounds (i.e., two challenge-response slots). Indeed, it is known that straightforward generalization of OR-composition to multi-slots protocols (i.e., the original OR-composition is applied on each slot separately) will yield an *unsound* protocol.

In more detail, recall that Ligero's messages are denoted as $(h, a, b, c, \widetilde{b}, \widetilde{c})$, where $(h, b, \widetilde{b})$ are nothing but public-coins (see Prot. 8). If we do the straightforward generalization of the above OR-composition (to prove an OR statement $x \vee x'$), it will work as follows: assuming P knows witness w for x, P uses $\mathsf{HVSim}(x')$ to simulate a proof $(h_2, a_2, b_2, c_2, \widetilde{b}_2, \widetilde{c}_2)$ for the x' part (because P does not have witness for it). Meanwhile, P generates the proof for x honestly, in the following manner: V sends h and P derives h_1 as $h_1 = h \oplus h_2$; P runs the honest Ligero prover's algorithm on input (x, w) to generate a_1, assuming the first Ligero verifier's message is h_1. Similarly, when V sends b (resp. $\widetilde{b}$), P will set $b_1 = b \oplus b_2$ (resp. $\widetilde{b}_1 = \widetilde{b} \oplus \widetilde{b}_2$), and compute the response c_1 (resp. $\widetilde{c}_1$) using the honest Ligero prover's algorithm (as it has witness w for x).

However, the above approach suffers from the following "cross attack": Since P^* has the opportunity to decide how to decompose h, b, and $\widetilde{b}$, it can pick a bad b_1 and a bad $\widetilde{b}_2$. That is, a cheating prover can choose malicious challenges in the first slot of the *left* execution and the second slot of the *right* execution, and there is no soundness guarantee for Ligero when a malicious prover can control (even) one challenge out of the two slots.

Solution. To resolve this problem, we ask P to commit to its decomposition in advance. More accurately, we ask P to generate $\mathsf{com} = \mathsf{SHCom}(h_2 \| b_2 \| \widetilde{b}_2; r)$ at the very beginning of the protocol, where SHCom is a statistically-hiding commitment. Then, we continue as the above. At the end of the execution, we ask P to give a statistical WI argument of knowledge sWIAoK for the following statement:

- com is committing to either $(h_1, b_1, \widetilde{b}_1)$ or $(h_2, b_2, \widetilde{b}_2)$.[8]

Intuitively, due to the (knowledge) soundness of sWIAoK, P^* cannot conduct the above "cross attack" anymore.

We denote this protocol as Π_{OR}. Due to space constraints, we put the formal description of Π_{OR} in Prot. 10 in Appx. C, where we also provide the complete security proof. Here, we want to emphasize that this approach invokes very small efficiency overhead compared with the plain OR-composition described in **The First Attempt**: what we add is simply a statistically-hiding commitment and a sWIAoK for its consistency. Using a modified version of Ligero as the underlying sWIAoK (see Appx. C.3), this only adds an extra computation cost of 32 milliseconds and an extra communication cost of 6.4MB. See Appx. C.2 for more details.

Regarding Malicious-Verifiers ZK. It is not hard to see that the above Π_{OR} is also an honest-verifier ZK argument (of knowledge). Using the Goldreich-Kahan technique [GK96] (as done in [IMS12, AHIV17]), we can convert it to a fully-secure ZK argument, i.e., against malicious verifiers. We denote the resulting protocol as Π'_{OR}, and present the full description of it in Prot. 11 of

[8] Note that $(h_1, b_1, \widetilde{b}_1)$ and $(h_2, b_2, \widetilde{b}_2)$ will be known to V when the protocol reaches the final sWIAoK stage.

Appx. C.1.3. Looking ahead, Π'_{OR} will be used in the instance-based non-malleable commitment in the next subsection (in Prot. 3).

5.3 Instance-Based Non-Malleability

Recall that we use $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ (Prot. 1) as our ENMC. The primary efficiency bottleneck in $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ is the consistency proof, which is done using Ligero'. Since an honest committer is never cheating, our goal is to provide the prover an easier way to get through this proof. Toward this goal, we first show an *instance-based* version of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$, denoted as $\langle C_L, R_L\rangle$. The instance-based version simply gives the option of using a witness for a true statement in the consistency proof phase of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$. At a high level, the parties get a statement x as input which may or may not be true. If x is true, the committer can additionally take as input a witness $w \in \mathcal{R}_L(x)$ and succeed in the proof phase by using w instead of completing the consistency proof for any message m. This allows the honest prover to fake the ENMC execution using a faster simulator thanks to the OR-composition. If x is false, the committer commits to a valid value m. It is also possible to do both: commit to m properly and execute consistency proof as well as proof for x. We present the full construction in Prot. 3,[9] and establish its security in Lem. 3 and 4.

Protocol 3: $\langle C_L, R_L\rangle(x)$: Instance-Based Non-Malleable Commitment

Instance-based $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ is the following commitment scheme, denoted as $\langle C_L, R_L\rangle$, defined for an arbitrary $\mathcal{NP}$ language L: the common input to both algorithms is a statement x; in addition, C_L takes a (private) auxiliary input that is either of the form $(m, \perp)$ or $(\perp, w)$ where w is a witness for $x \in L$. Recall that $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ is denoted by $\langle C, R\rangle$ and depicted in Prot. 1. The protocol proceeds in two phases:

- **Commit Stage:** In this stage R_L proceeds identically to algorithm R of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ and let ρ be its first message. For input $(m, \perp)$, C_L proceeds exactly as C proceeds in the commit stage on input m. For input $(\perp, w)$, C_L simply sends *random* values of appropriate size as the second and fourth messages of the commit stage (when interacting with R_L). Recall that the execution of the **Commit Stage** of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ will yield messages $\widehat{\mathbf{m}}$, $\widehat{\mathbf{r}}$, and $\{w_i\}_{i\in[n]}$ (see Prot. 1). We denote $\mathsf{st} \coloneqq (\widehat{\mathbf{m}}, \widehat{\mathbf{r}}, \{w_i\}_{i\in[n]})$.
- **Proof Stage:** In this stage, C_L proves that $(x, \mathsf{st}) \in L \vee L^{\rho}_{\mathsf{consis}}$ using Π'_{OR}, i.e., the fully ZK version of Π_{OR} (see Prot. 11). For input $(m, \perp)$, C_L uses the simulator HVSim for the left part (i.e., for x), and completes right part (i.e., for st) honestly by using the witness for st (from the first phase). For input $(\perp, w)$ it uses w to succeed in the left part of the proof and simulator HVSim to succeed in the right part.

If the common statement is fixed to x, we denote the instance-based $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ by $\langle C_L, R_L\rangle(x)$. The executions corresponding to inputs $(m, \perp)$ will be called **real or honest executions**, and those corresponding to $(\perp, w)$, **fake or simulated executions** of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ (or ENMC).

Lemma 3. *Let L be an $\mathcal{NP}$ language. For every $x \notin L$ protocol $\langle C_L, R_L\rangle(x)$ (Prot. 3) is an extractable non-malleable commitment scheme.*

[9] We warn that this version cannot be used in our NMZK protocol yet. See Sec. 5.4.

Proof. We observe that for every $x \notin L$, the **Proof Stage** of the protocol is a ZK argument for $\mathsf{st} \in L^{\rho}_{\mathsf{consis}}$ (i.e., consistent execution of the commit stage). In this case, $\langle C_L, R_L \rangle(x)$ is simply an instantiation of the original $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ protocol. The claim then follows from the security of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$. □

Lemma 4. *Let L be an $\mathcal{NP}$ language with witness relation $\mathcal{R}_L$. For every message m and every $(x, w) \in \mathcal{R}_L$, the following holds:*

$$\{\mathsf{view}_0 \leftarrow \langle C_L((m, \perp)), R_L \rangle(x) : \mathsf{view}_0\} \quad \overset{c}{\approx} \quad \{\mathsf{view}_1 \leftarrow \langle C_L((\perp, w)), R_L \rangle(x) : \mathsf{view}_1\}\,.$$

Proof. This lemma follows from the following two observations: (i) committer's messages in the commit stage are *pseudorandom* (since second message of Naor's commitment is pseudorandom), and (ii) the proof stage is WI (it is indeed ZK). Since the proof follows from a standard hybrid argument, we omit the details. □

Remark 4 (On Efficiency). It is worth noting that if x admits a fast Ligero proof, then fake executions are faster than the real executions since the simulator for Ligero for the right part (i.e., the real consistency proof for st) is much faster than the prover. As mentioned in Sec. 1.2, this is how we manage to obtain significant improvement on the efficiency.

5.4 Efficient Simulation-Sound Zero-Knowledge

The main benefit of the instance-based $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ in Prot. 3 is that if $x \in L$ admits fast proofs, it can be used in place of standard $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ in our NMZK protocol. Unfortunately, the resulting protocol is not a NMZK for true x! Nevertheless, the resulting protocol is *simulation-sound* (as per Def. 3), and equally importantly, *efficient*. We refer to this protocol by Π_{SS} and specify it in Prot. 4.

Protocol 4: Π_{SS}: Simulation-Sound ZKAoK

The common input is x and prover's input is a witness w for $x \in L$, where L is the desired $\mathcal{NP}$ language. This protocol is identical to protocol $\langle P, V \rangle_{\mathsf{NMZK}}$ (Prot. 2) except that the Step 2 ENMC is replaced with the instance-based non-malleable commitment (Prot. 3) with the following inputs: the common input is x and committer's auxiliary input in the **Proof Stage** of the commitment is $(\perp, w)$. Observe that the honest prover only performs a simulated execution of the non-malleable commitment.

Theorem 2. *Protocol Π_{SS} (Prot. 4) is a simulation-sound zero-knowledge argument of knowledge.*

Proof. Completeness is straightforward. We now show a single simulator S for the protocol that satisfies the two requirements in Def. 3. The proof follows closely from that of Thm. 1. We provide more details in the following.

Indistinguishable Simulation. Simulator S (for proving Def. 3) is identical to the hybrid machine H_3 constructed in Sec. 4.1 (where the "trapdoor" statement is set successfully), except that it performs a real execution of the instance-based $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ protocol in Step 2 (i.e., it uses $(\sigma, \perp)$ as the private input of the committer). It is straightforward to verify that S does not use the witness. Then, from the proof of Lem. 2 and the additional observation that the real and simulated executions of instance-based $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ are indistinguishable (Lem. 4), it follows that the output of S is indistinguishable from that of a real MIM experiment.

Simulation Soundness. To prove this property, we use the same simulator S as in last paragraph. As mentioned earlier, in the left execution, S uses $(\sigma, \perp)$ as the private input of the committer in the Step 2 instance-based ENMC (regardless of whether x is true or false). Assume for contradiction that $\widetilde{x}$ is false. Then the man-in-the-middle adversary $\mathcal{A}$ is forced to perform a real execution of the instance-based $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ protocol in Step 2 (because there is no witness $\widetilde{w}$ to $\widetilde{x}$ that can be used by $\mathcal{A}$ to run a fake execution of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$). Therefore, the $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ instances in both the left and right executions are (real) ENMC. Then, following the same argument as in the proof of Lem. 2 (from hybrid 3 to the end), we can extract a valid witness $\widetilde{w}$ for $\widetilde{x}$ (if V accepts in the right), contradicting the assumption that $\widetilde{x}$ is false. Therefore, whenever the honest verifier accepts (in the right execution), $\widetilde{x}$ must be a true statement, except for negligible probability. This establishes the simulation soundness.

Remark 5 (On the Non-Malleability of Π_{SS}). Due to the high similarity of the above proof to that of Lem. 2, one may wonder why we did *not* claim that Π_{SS} is non-malleable (instead of "simulation-sound"). The reason is that the above proof relies crucially on the condition that $\widetilde{x}$ is false. If $\widetilde{x}$ is true, the man-in-the-middle $\mathcal{A}$ may (use $(\widetilde{w}, \perp)$ to) perform a fake execution of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ in the right interaction. Then, the Step 2 execution in the right may not be a proper ENMC. In this case, the proof of Lem. 2 does not go through any more. More specifically, it is unclear how to argue that the invariant condition holds when we switch from H_1 to H_2 (as we did in the proof of Lem. 2).

Argument of Knowledge. To prove the AoK property, we only need to consider the stand-alone execution (in contrast to the above man-in-the-middle setting). Consider a malicious prover P^* that convinces the honest verifier on a statement x. First, note that P^* cannot make the **Commit Stage** (see Prot. 3) of the Step 2 $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ to be a valid commitment to σ; otherwise, one can extract σ from the **Commit Stage** of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ (regardless of whether it is a real execution or a fake execution of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$)[10], which breaks the computationally-hiding property of the Stage-1 ExtCom. Given that, we now observe that the AoK property of Π_{SS} follows from the AoK property of the **Stage-4** sWIAoK. □

5.5 Putting It All Together: Fast NMZK and NMCom

Now we show how to get efficient and full-fledged non-malleable zero-knowledge and commitment protocols with the help of our efficient simulation-sound ZKAoK protocol Π_{SS} and the statistically WIAoK protocol Π_{OR}.

Fast NMZK Protocol. We present our final NMZK protocol in Prot. 5. At a high level, the prover in Prot. 5 sets up a "trapdoor statement" in the form of a commitment cm, and proves using Π_{SS} that cm is a commitment to 0. Later, the prover proves using protocol Π_{OR} that either the statement is true or that cm is a commitment to 1. The honest prover always commits to 0 and thus remains fast. The simulator commits to 1 instead. We establish the security of Prot. 5 by Thm. 3.

Protocol 5: $\langle P, V\rangle_{\mathsf{final}}$: Non-Malleable ZKAoK
The common inputs are statement x, tag id, and security parameter λ. Prover's private input is a witness $w \in R_L(x)$, where L is the desired $\mathcal{NP}$ language. The protocol proceeds as follows:

[10] Recall that if the **Committing Stage** of $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ is a valid commitment, the committed value can be extracted by rewinding this stage only. This property has nothing to do with the **Consistency Proof** stage.

1. P commits to 0^λ using 2-round Naor commitment; let ρ be the first message of this commitment and $\mathsf{cm} = \mathsf{Com}_\rho(0^\lambda)$ the second message.
2. P and V execute Π_{SS} with tag id, where P proves that cm is a valid commitment to 0^λ.
3. P and V execute Π_{OR}, where P proves that:
   - $x \in L$, **or**
   - cm is a valid commitment to 1^λ, i.e., $(\mathsf{cm}, 1^\lambda) \in L_{\mathsf{Com}_\rho}$.

Theorem 3. *Protocol $\langle P, V\rangle_{\mathsf{final}}$ (Prot. 5) is a non-malleable zero-knowledge argument of knowledge for L.*

Proof. Completeness is straightforward. The argument of knowledge property (and hence soundness) follows from the soundness of Π_{SS} and the argument of knowledge property of Step 3 WIAoK. In the following we prove the zero-knowledge property and non-malleability.

Zero-Knowledge. The simulator Sim for the ZK property proceeds as follows: it sets cm to be a commitment to 1^λ with randomness r and succeeds in Step 2 using the simulator S guaranteed for Π_{SS}; It uses the fake witness $(1^\lambda, r)$ to succeed in WIAoK in Step 3. Indistinguishability properties and running time of Sim follow the standard arguments and omitted. The hybrids given below can also be used to prove these properties.

Non-Malleability. The proof of non-malleability is almost identical to the non-malleability proof for $\langle P, V\rangle_{\mathsf{NMZK}}$ (Lem. 2). We only provide a sketch. Given any man-in-the-middle adversary $\mathcal{A}_{\mathsf{mim}}$ that can convince an honest verifier with probability p, we consider the following sequence of hybrids:

- Hybrid H_0: the hybrid H_0 sets up the left and right executions for $\mathcal{A}_{\mathsf{mim}}$ with $P(x, w)$ and V respectively. It outputs the joint view of $\mathcal{A}_{\mathsf{mim}}$ containing both left and right executions.
- Hybrid H_1: this hybrid is identical to H_0, except that H_1 uses the simulator Sim_{SS} of Π_{SS} to simulate $\mathcal{A}_{\mathsf{mim}}$'s view in the right and left Step 2 execution.
- Hybrid H_2: this hybrid is identical to H_1, except that H_2 sets cm (the Step 1 commitment on the right) to a commitment to 1^λ.
- Hybrid H_3: this hybrid is identical to H_2, except that H_3 uses the witness for "cm is a valid commitment to 1^λ" in the Step 3 sWIAoK on the left.

The key point of the argument is to show that when H_2 starts to set cm to a commitment to 1^λ and invokes Sim_{SS} to go through Step 2, $\mathcal{A}_{\mathsf{mim}}$ cannot set $\widetilde{\mathsf{cm}}$ (the right Step 1 commitment) to a commitment to 1^λ, and give a convincing proof in the right sWIAoK with probability more than negligible. This is guaranteed by the simulation soundness of Π_{SS} (Def. 3). Thus, for the standalone execution (to show non-malleability property), we can build a simulator Sim that internally runs H_3. Sim will extract a witness $\widetilde{w}$ from the right sWIAoK with probability negligibly close to p, and use it to honestly convince an external verifier of $\langle P, V\rangle_{\mathsf{final}}$. This finishes the proof of non-malleability. □

Fast NMCom Protocol. Our non-malleable commitment protocol is presented in Prot. 6. At a high level, Prot. 6 works in the same way as the non-malleable zero-knowledge protocol above,

except that x is replaced with a commitment to the desired value. Its security proof follows closely from the proofs of Thm. 3 and Lem. 2. Thus, we omit the details.

Protocol 6: $\langle C, R\rangle_{\mathsf{final}}$: Non-Malleable Commitment

The common input is a tag id and the security parameter λ. Private input of the committer is a value $v \in \{0,1\}^\lambda$. The protocol proceeds as follows:

1. C commits to v using two-round Naor commitment; let R's first message be ρ, and $c = \mathsf{Com}_\rho(v)$ denote the second message.
2. C further commits to 0^λ using ρ as first message. Let $\mathsf{cm} = \mathsf{Com}_\rho(0^\lambda)$.
3. C proves that cm is valid commitment to 0^λ using Π_{SS} with tag id.
4. C proves using Π_{OR} that:
   - there exists v such that c is a valid commitment to v, i.e., $(c, v) \in L_{\mathsf{Com}_\rho}$, **or**
   - cm is a valid commitment to 1^λ, i.e., $(\mathsf{cm}, 1^\lambda) \in L_{\mathsf{Com}_\rho}$.

6 Summary of Performance

We analyze both the communication and computational complexity of SSZK, NMZK, and NMCom protocols as well as a concrete evaluation of their cost. We provide a summary of the asymptotic performance in Table 2. The concrete performance was already provided toward the end of the introduction (Table 1).

We measure the complexity for given circuit size s in terms of number of AES_λ and SHA256_λ evaluations. We denote by AES_λ an evaluation of the AES block cipher on a λ-bit string and SHA256_λ in the same manner for SHA256. We denote the size of the consistency-checking circuit by C_{cons} and the "equality testing" circuit by C_{eq}—this circuit on input (c, v, r) tests whether c is a commitment to v with randomness r.

A more detailed analysis can be found in Appx. D.

Table 2: Asymptotic complexity of NMZK, Ligero, and NMCom. Only dominating terms are shown in each cell. Plain Ligero is shown for comparison purposes.

Protocol	**#AES$_\lambda$**	**#SHA256$_\lambda$**	**Communication**
NMZK	λ	$2s + \sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}} + \sqrt{C_{\mathsf{eq}}}\log C_{\mathsf{eq}} + 2C_{\mathsf{eq}}$	$\lambda(\lambda + k^2 + 2\sqrt{s} + \sqrt{C_{\mathsf{cons}}} + 3\sqrt{C_{\mathsf{eq}}} + 2\sqrt{C_{\text{SHA}}})$
NMCom	1	$2s + \sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}} + \sqrt{C_{\mathsf{eq}}}\log C_{\mathsf{eq}} + 3C_{\mathsf{eq}}$	$\lambda(\lambda + k^2 + 2\sqrt{s} + \sqrt{C_{\mathsf{cons}}} + 4\sqrt{C_{\mathsf{eq}}} + 2\sqrt{C_{\text{SHA}}})$
Ligero	0	s	$\lambda\sqrt{s}$

References

ADL14. Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th ACM STOC*, pages 774–783. ACM Press, May / June 2014. 2

AHI+17. Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. Low-complexity cryptographic hash functions. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 7:1–7:31, 67, January 2017. LIPIcs. 36

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017. 1, 3, 4, 8, 21, 34, 35, 41, 42, 43, 50

Bar01. Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001. 2

Bar02. Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd FOCS*, pages 345–355. IEEE Computer Society Press, November 2002. 1, 2

BCFW09. Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 524–541. Springer, Heidelberg, December 2009. 1

BCGT13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete efficiency of probabilistically-checkable proofs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 585–594. ACM Press, June 2013. 2

BDH+17. Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. Concurrently composable security with shielded super-polynomial simulators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 351–381. Springer, Heidelberg, April / May 2017. 1

BG08. Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008. 2

BGJ+18. Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 459–487. Springer, Heidelberg, August 2018. 1

BGR+15. Hai Brenner, Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. Fast non-malleable commitments. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 1048–1057. ACM Press, October 2015. 1, 2, 3, 4, 5, 8, 9, 47

BPS06. Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *47th FOCS*, pages 345–354. IEEE Computer Society Press, October 2006. 1, 2, 4

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. 2

Can00. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. 1

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. 1

CCG+20. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 291–319. Springer, Heidelberg, November 2020. 1

CDNO97. Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Heidelberg, August 1997. 3

CDPW07. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007. 3

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994. 3, 5, 19, 20, 36

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. 3

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. 1

CLP10. Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550. IEEE Computer Society Press, October 2010. 4

COSV17. Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 127–157. Springer, Heidelberg, August 2017. 1

Dam02. Ivan Damgård. On σ-protocols. http://www.cs.au.dk/~ivan/Sigma.pdf, 2002. 3, 5, 36

DDN91. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991. 1

DIO98. Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *30th ACM STOC*, pages 141–150. ACM Press, May 1998. 3

DKOS01. Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 40–59. Springer, Heidelberg, May 2001. 3

DNRS99. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, October 1999. 3

DPW10. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *ICS*, pages 434–452, 2010. 1

DS98. Cynthia Dwork and Amit Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 442–457. Springer, Heidelberg, August 1998. 3

DW09. Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 601–610. ACM Press, May / June 2009. 1

FGJ18. Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. On the existence of three round zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 3–33. Springer, Heidelberg, April / May 2018. 1

FKMV12. Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In *International Conference on Cryptology in India*, pages 60–79. Springer, 2012. 2

GGJS12. Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 99–116. Springer, Heidelberg, April 2012. 32

GK96. Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996. 21, 41, 42

GK03. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003. 3

GKP17. Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 194–224. Springer, Heidelberg, April / May 2017. 32

GLOV12. Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012. 1

GMO16. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 1069–1083, 2016. 4

GMPP16. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476. Springer, Heidelberg, May 2016. 1

GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. 1

Gol01. Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. 6, 33

Gol04. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. 6

Goy11. Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704. ACM Press, June 2011. 1

GPR16. Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016. 1, 2, 7

GR19. Vipul Goyal and Silas Richelson. Non-malleable commitments using goldreich-levin list decoding. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 686–699. IEEE, 2019. 1

GRRV14. Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th FOCS*, pages 41–50. IEEE Computer Society Press, October 2014. 1, 9

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. 1

HM96. Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 201–215. Springer, Heidelberg, August 1996. 33, 46

HR04. Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105. Springer, Heidelberg, August 2004. 32

IMS12. Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge PCPs. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 151–168. Springer, Heidelberg, March 2012. 8, 21, 34, 35, 36, 41, 42

IW14. Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 121–145. Springer, Heidelberg, February 2014. 8

JP14. Abhishek Jain and Omkant Pandey. Non-malleable zero knowledge: Black-box constructions and definitional relationships. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14*, volume 8642 of *LNCS*, pages 435–454. Springer, Heidelberg, September 2014. 4, 5, 7

Khu17. Dakshita Khurana. Round optimal concurrent non-malleability from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 139–171. Springer, Heidelberg, November 2017. 1

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992. 5, 8, 34, 36

Kiy14. Susumu Kiyoshima. Round-efficient black-box construction of composable multi-party computation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 351–368. Springer, Heidelberg, August 2014. 4

KOS03. Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595. Springer, Heidelberg, May 2003. 1

KS17. Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In Chris Umans, editor, *58th FOCS*, pages 564–575. IEEE Computer Society Press, October 2017. 3

LP09. Huijia Lin and Rafael Pass. Non-malleability amplification. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 189–198. ACM Press, May / June 2009. 1

LP11a. Huijia Lin and Rafael Pass. Concurrent non-malleable zero knowledge with adaptive inputs. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 274–292. Springer, Heidelberg, March 2011. 4

LP11b. Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 705–714. ACM Press, June 2011. 1

LPS17. Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *58th FOCS*, pages 576–587. IEEE Computer Society Press, October 2017. 3

LPTV10. Huijia Lin, Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable zero knowledge proofs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 429–446. Springer, Heidelberg, August 2010. 4

LPV08. Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 571–588. Springer, Heidelberg, March 2008. 1, 7

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994. 5, 8

MOSV06. Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent zero knowledge without complexity assumptions. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 1–20. Springer, Heidelberg, March 2006. 32

MP03. Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 140–159. Springer, Heidelberg, May 2003. 2

MPR06. Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th FOCS*, pages 367–378. IEEE Computer Society Press, October 2006. 1

MX13. Mohammad Mahmoody and David Xiao. Languages with efficient zero-knowledge PCPs are in SZK. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 297–314. Springer, Heidelberg, March 2013. 35

Nao90. Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 128–136. Springer, Heidelberg, August 1990. 9, 10

Nao91. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991. 46

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003. 1

Nie02. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Heidelberg, August 2002. 3

OPV10. Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 535–552. Springer, Heidelberg, February 2010. 2, 4

OPW11. Adam O'Neill, Chris Peikert, and Brent Waters. Bi-deniable public-key encryption. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 525–542. Springer, Heidelberg, August 2011. 3

Pas03a. Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003. 2, 3

Pas03b. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003. 1

Pas11. Rafael Pass. Concurrent security and non-malleability (invited talk). In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, page 540. Springer, Heidelberg, March 2011. 1, 3

PPV08. Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2008. 1, 3

PR05a. Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *46th FOCS*, pages 563–572. IEEE Computer Society Press, October 2005. 1

PR05b. Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 533–542. ACM Press, May 2005. 1, 2, 6

PRS02. Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *43rd FOCS*, pages 366–375. IEEE Computer Society Press, November 2002. 32

PS04. Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th ACM STOC*, pages 242–251. ACM Press, June 2004. 1

PW09. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Heidelberg, March 2009. 32

Rog06. Phillip Rogaway. Formalizing human ignorance: Collision-resistant hashing without the keys. vietcrypt'06, lncs vol. 4341, 2006. 46

Ros04. Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 191–202. Springer, Heidelberg, February 2004. 32

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. 5, 7

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. 3

Unr07. Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 205–223. Springer, Heidelberg, August 2007. 3

Wee09. Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 417–434. Springer, Heidelberg, December 2009. 3

Wee10. Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010. 1

A Additional Preliminaries

A.1 Collision-Resistant Hash Families

We use the definition from [HR04].

Definition 6 (Collision-Resistant Hash Families). *A collision-resistant hash family (CRHF) is a collection of functions $\mathcal{H} = \{h_i\}_{i \in I}$ for some index set I, where $h_i : \{0,1\}^{\ell(|i|)} \to \{0,1\}^{\ell'(|i|)}$ and $\ell(|i|) > \ell'(|i|)$. It satisfies the following requirements:*

- **Key Generation.** *There exists a PPT key generating algorithm* KGen*, so that* $\mathsf{KGen}(1^\lambda) \in \{0,1\}^{m(\lambda)} \cap I$*, where $m(\lambda)$ is a polynomial on λ representing the length of the key.*
- **Efficient Evaluation.** *There exists a (deterministic) polynomial time algorithm* Eval *such that $\forall i \in I$ and $\forall x \in \{0,1\}^{\ell(|i|)}$, $\mathsf{Eval}(i, x) = h_i(x)$.*
- **Non-Uniform Collision Resistance.** *For any non-uniform PPT machine $\mathcal{A}$, the following holds:*
$$\Pr[i \xleftarrow{\$} \mathsf{KGen}(1^\lambda), (x, x') \leftarrow \mathcal{A}(i) : x \neq x' \wedge h_i(x) = h_i(x')] \leq \mathsf{negl}(\lambda).$$
- **Public-Coin:** *if the index set I is $\{0,1\}^*$ and* $\mathsf{KGen}(1^\lambda)$ *outputs a uniformly distributed string from $\{0,1\}^{m(\lambda)}$, then we say that it is a* public-coin *CRHF, i.e., the family remains collision-resistant even if the randomness used to generate the key is known to the adversary. Also, we emphasize that the collision-resistance property holds against* non-uniform *PPT adversaries.*

A.2 Extractable Commitment

We present here the definition of extractable commitments. A commitment scheme is extractable if there exist an efficient extractor such that, as long as the committer behaves honestly, the committed value can be extracted. Constructions for such commitment already existed implicitly in the implementation of concurrent zero-knowledge protocols in [PRS02, Ros04]. This concept and constructions were later made explicit in [MOSV06], which also inherited the concurrent extractability from [PRS02]. The standalone version was later formalized and used in other works [PW09, GGJS12, GKP17]. Extractability in the standalone setting suffices our purpose. We now present the definition (in Def. 7) and construction (in Prot. 7) from [PW09].

Definition 7 (Extractable Commitment). *A commitment scheme* $\mathsf{ExtCom} = (S, R)$ *is extractable if there exists an expected polynomial-time probabilistic oracle machine (the extractor)* Ext *that given oracle access to any PPT cheating sender S^* outputs a pair (τ, σ^*) such that:*

- **Simulation:** *τ is identically distributed to the view of S^* at the end of interacting with an honest receiver R in commitment phase.*
- **Extraction:** *the probability that τ is accepting and $\sigma^* = \perp$ is negligible.*
- **Binding:** *if $\sigma^* \neq \perp$, then it is statistically impossible to open τ to any value other than σ^*.*

Protocol 7: ExtCom**: Extractable Commitment Scheme**

The extractable commitment scheme, based on any commitment scheme Com, works in the following way.

Input:

– both S and R get security parameter 1^λ as the common input.

– S gets a string σ as his private input.

Commitmment Phase:

– The sender (committer) S commits using Com to λ pairs[a] of strings $\{(v_i^0, v_i^1)\}_{i=1}^{\lambda}$ where $(v_i^0, v_i^1) = (\eta_i, \sigma \oplus \eta_i)$ and η_i are random strings in $\{0,1\}^{\ell(\lambda)}$ for $1 \leq i \leq \lambda$.

– Upon receiving a challenge $\overline{c} = (c_1, \ldots, c_\lambda)$ from the receiver R, S opens the commitments to $(v_1^{c_1}, \ldots, v_\lambda^{c_\lambda})$.

– R checks that the openings are valid.

Decommitment Phase:

– S sends σ and opens the commitments to all λ pairs of strings.

– R checks that all the openings are valid, and also that $\sigma = v_1^0 \oplus v_1^1 = \cdots = v_\lambda^0 \oplus v_\lambda^1$.

[a] Actually, the scheme will be secure as long as we use Com to commit $\omega(\log \lambda)$ such pairs.

A.3 The Halevi-Micali Commitment Scheme

We use the standard definition for statistically-hiding commitments [Gol01]. In particular, we will use the statistically-hiding commitment proposed by Halevi and Micali [HM96]. We refer the reader to [HM96] for details. In the following, we briefly recall the construction and its efficiency when committing to a message $m \in \{0,1\}^{n(\lambda)}$, where $n(\lambda)$ is some fixed polynomial on the security parameter λ.

Construction. Let $L \coloneqq 4\lambda + 2n + 4$. Let $h : \{0,1\}^L \to \{0,1\}^\lambda$ be a collision resistant hash function; let $\mathcal{H}$ be a pair-wise independent hash family from $\{0,1\}^L$ to $\{0,1\}^n$. Using the standard construction, each element in $\mathcal{H}$ can be described using $(L + 2\lambda)$ bits.

To commit to m, the sender picks a random $r \in \{0,1\}^L$ and computes $y = h(r)$. Then, it picks a random function $H \in \mathcal{H}$ for which $H(r) = m$. The commit-string is $\mathsf{SBCom} = (H, y)$ and the decommitment information is just r.

Efficiency. The communication complexity of this construction is summarized below

$$\begin{cases} \text{Commit Stage:} & |\mathsf{SBCom}| = |H| + |y| = (L + 2\lambda) + \lambda = 7\lambda + 2n + 4 \\ \text{Decommit Stage:} & |\mathsf{SBCom}| = |r| = L = 4\lambda + 2n + 4 \end{cases}.$$

The computational complexity is the same for the sender and the receiver. It only consists of evaluating the following functions

– the collision-resistant hash $h : \{0,1\}^{4\lambda+2n+4} \to \{0,1\}^\lambda$, **and**

– the pair-wise independent hash $H : \{0,1\}^{4\lambda+2n+4} \to \{0,1\}^n$.

B An Overview of the Zero-Knowledge Protocol Ligero

In the following, we recall the structure of the zero-knowledge protocol Ligero from [AHIV17]. We will recall related notation and techniques only to the extent that is adequate to understanding our construction. More details (e.g., ZKIPCP, parameter settings, etc.) can be found in [AHIV17].

The Honest-Verifier Version. In [AHIV17], the authors first built an efficient *zero-knowledge interactive PCP* (ZKIPCP) protocol for the satisfiability of arithmetic circuits[11]. Then, they applied the [Kil92, IMS12] technique to convert the ZKIPCP to a *sub-linear* honest-verifier ZKAoK in 6 rounds. We present this protocol in Prot. 8. We suppress technical details that are less relevant to our work. Interested readers can refer to [AHIV17] for related information.

Protocol 8: Ligero: Honest-Verifier ZKAoK [AHIV17]

This construction uses a two-round statistical-hiding commitment SHCom, and a collision-resistant hashing family $\{\mathcal{H}_\lambda\}_\lambda$. Note that SHCom can be constructed from any collision-resistant hashing family. This protocol is for the satisfiability of arithmetic circuits Cir over a finite field $\mathbb{F}$.

Besides the security parameter λ, this protocol is parameterized by several extra parameters n, t, m and ℓ. To achieve 2^λ soundness, one can set $|\mathbb{F}| > O(2^\lambda)$, $n = O(\lambda)$, $t = \delta n$ for some small constant $\delta \in (0, 1)$, $m = O(\sqrt{s/\lambda})$ and $\ell = O(\sqrt{s \cdot \lambda})$, where s is the number of gates in the circuit Cir. Since they are not crucial to understand our protocol, we refer the reader to [AHIV17] for more details.

Public Input: Security parameter λ, an arithmetic circuit Cir over a field $\mathbb{F}$.

Private Input: Prover P takes the witness x s.t. $\mathsf{Cir}(x) = 1$.

1. V samples a CRHF $h_\alpha \xleftarrow{\$} \mathcal{H}_\lambda$ and sends it to P. Looking ahead, h_α will be used both as the first-round message of SHCom and as the hash function to build a Merkle tree.
2. P computes the ZKIPCP oracle π for the proof of $\mathsf{Cir}(x) = 1$, where the aforementioned parameter n denotes the length of π. P commits to π using the second-round message of SHCom. P uses h_α to build a Merkle hashing tree on the committed π. P sends the root of Merkle tree root to V.
3. V sends to P a random vector $\Gamma_1 \xleftarrow{\$} \mathbb{F}^{5m+4m\ell}$. We remark that Γ_1 is the challenges corresponding to the "interactive" part of the ZKIPCP.
4. P answers V's challenge Γ_1.
5. V sends to P a size-t random subset $\Gamma_2 \subseteq [n]$. We remark that Γ_2 contains the positions of Merkle tree leaves where V wants to challenge P, i.e. the queries to the "oracle" part of the ZKIPCP.
6. P answers V's challenge Γ_2 (i.e., decommitting to the leaves specified by Γ_2 and sending the related Merkle paths).

Verifier's decision: V accepts iff P's answers to both Γ_1 and Γ_2 are convincing.

[11] They also talked about how to optimize their construction when working with Boolean circuits.

Notation: Throughout this paper, we will denote these six rounds as $(h, a, b, c, \widetilde{b}, \widetilde{c})$. We also remark that if the CRHF is picked from a *public-coin* CRHF family (as defined in Def. 6), then V can simply send (in Round 1) the randomness that specifies the specific hash function. Therefore, we view h as the randomness that specifies the underlying function. In this way, it is clear that this protocol is public-coin.

HVZK Simulator. The protocol shown in Prot. 8 is a honest-verifier statistical zero-knowledge argument of knowledge. In this paper, we will need to use the honest-verifier ZK simulator for this protocol. We denote this simulator as HVSim. Parametrically, HVSim takes the statement Cir and randomly sampled $(h, b, \widetilde{b})$ as input, and output $\mathsf{HVSim}(\mathsf{Cir}, h, b, \widetilde{b}) = (h, a, b, c, \widetilde{b}, \widetilde{c})$, which are negligibly close to the distribution of the view of an honest verifier.

In this work, we relies heavily on the fact that this HVSim can be implemented very efficiently. We provide more details on how it works in Algo. 1 of Sec. 3.

Against Malicious Verifiers. [AHIV17] also showed how to make the protocol secure against malicious verifiers, using the technique developed in [IMS12, MX13]. Roughly speaking, it works in the same way as the Goldreich-Kahan transformation to make an honest-verifier ZK secure against malicious verifiers. Concretely, we ask the verifier to commit to Γ_1 and Γ_2 with a statistically-hiding commitment scheme at the beginning, and decommit to them later to let P responds accordingly. We denote this fully-secure zero-knowledge protocol as Ligero' and present it formally in Prot. 9.

Protocol 9: Ligero': The Malicious-Verifier Version of Ligero [AHIV17]

This protocol is for the satisfiability of arithmetic circuits Cir over a large field $\mathbb{F}$. It assumes the existence of two-round statistically-hiding commitment SHCom, a non-interactive statistically-binding commitment Com, a collision resistant hash family $\{\mathcal{H}_\lambda\}_\lambda$ and the ZKIPCP protocol from [AHIV17]. (The parameters below share the same meaning as in Prot. 8.)

Public Input: Security parameter λ, an arithmetic circuit Cir over a field $\mathbb{F}$.

Private Input: Prover P takes the witness x s.t. $\mathsf{Cir}(x) = 1$.

1. P samples a CRHF $h_\beta \xleftarrow{\$} \mathcal{H}_\lambda$ and sends it to V. Looking ahead, this h_β will be used both as the first-round message of a SHCom^{h_β} scheme by which the verifier commits to its challenge string.
2. V performs the following two sub-steps:
   (a) V samples a random vector $\Gamma_1 \xleftarrow{\$} \mathbb{F}^{5m+4m\ell}$ and a size-t random subset $\Gamma_2 \subset [n]$ of size t. V commits to P the values Γ_1 and Γ_2 separately in parallel, using the second-round message of SHCom^{h_β}.
   (b) V also samples a CRHF $h_\alpha \xleftarrow{\$} \mathcal{H}_\lambda$ and sends it to P. Looking ahead, this h_α will be used both as the first-round message of a $\mathsf{SHCom}^{h_\alpha}$ scheme, and as the hash function to build the Merkle tree.
3. P computes the ZKIPCP oracle π for the proof of $\mathsf{Cir}(x) = 1$. P commits to π using the second-round message of SHCom. P uses h_α to build a Merkle hashing tree on the committed π. P sends the root of Merkle tree root to V.

4. V decommits to Γ_1 (w.r.t. its Round 2 commitment SHCom^{h_β}).
5. P checks if the decommitment to Γ_1 is valid. If not, P aborts; otherwise, P responds according to Γ_1 as in the honest-verifier version Ligero (Prot. 8).
6. V decommits to Γ_2 (w.r.t. its Round 2 commitment SHCom^{h_β}).
7. P checks if the decommitment to Γ_2 is valid. If not, P aborts; otherwise, P responds according to Γ_2 as in the honest-verifier version Ligero (Prot. 8).

Verifier's decision: V accepts iff P's answers to both Γ_1 and Γ_2 are convincing.

C OR-Composition of the Honest-Verifier Ligero (Full Version)

We now describe how to obtain efficient sWIAoK protocols that can be used in Step 4 in our NMZK protocols (Prot. 2). As noted earlier, the key point is that the simulators of sub-linear ZK protocols—particularly those based on PCPs [Kil92, IMS12, AHI+17], including Ligero—can be much faster than the honest prover algorithm. We show how to combine this observation with the OR-composition technique [CDS94, Dam02] to greatly improve the efficiency of our protocols.

Road Map. We present the construction and its security proof in Appx. C.1. Then, in Appx. C.2, we show how to instantiate different components of our construction such that the resulting protocol is practically fast. In particular, our construction will make use of a statistically WIAoK protocol; we will give an efficient construction based on Ligero; this construction is shown in Appx. C.3.

C.1 Construction

As we have already discussed in Sec. 5.2, the plain generalization of the OR-composition technique from [CDS94] is unlikely to work for Ligero as it is not a sigma protocol. In particular, Ligero has two challenge-response slots; a malicious prover may maul different parts of these two slots to break soundness.

Also, we already presented in Sec. 5.2 the high-level idea about how our construction addresses this problem: we ask P to commit in advance the shares for the session that it will run HVSim for; at the end, P will give a statistically WIAoK proving that it really behaves as required. We now present the formal description of our construction in Prot. 10. It makes use of the following building blocks:

- A statistically-hiding commitment scheme SHCom. In particular, we use the 2-round Halevi-Micali scheme shown in Appx. A.3.
- The Ligero protocol shown in Prot. 8.
- A statistically WIAoK protocol sWIAoK. We remark that any statistically WIAoK will work. But to achieve better efficiency, we will design a specific sWIAoK again based on Ligero. We will show how to do it in Appx. C.3.

Protocol 10: Π_{OR}: OR-Composition of Ligero

This protocol, which we denote as Π_{OR}, runs in parallel two instances of the (honest-verifier) Ligero (Prot. 8), whose 6 messages are denoted as $(h, a, b, c, \widetilde{b}, \widetilde{c})$. It proves some instance of an

OR-composed language $(x, x') \in L \vee L'$. W.l.o.g., we assume that P is given a witness w for x (i.e., P knows the *left* witness).

1. V sends the first message for SHCom.
2. P computes the following messages:
   (a) P runs the statistical HVZK simulator for Ligero to compute $(h_2, a_2, b_2, c_2, \widetilde{b}_2, \widetilde{c}_2) \leftarrow \mathsf{HVSim}(x')$.
   (b) P samples a random tape r and generates $\mathsf{com} = \mathsf{SHCom}(h_2 \| b_2 \| \widetilde{b}_2; r)$.

   P sends com to V.
3. V samples uniformly at random a string h, which has the same length as the 1st message of Ligero. It sends h.
4. P computes $h_1 = h \oplus h_2$. It then runs the prover's algorithm of Ligero on statement x, using w as the witness and h as the 1st message, to obtain the 2nd message a_1. P sends (h_1, a_1, h_2, a_2). (Recall that (h_1, a_2) were defined in Step 2b.)
5. V samples uniformly at random a string b, which has the same length as the 3rd message of Ligero. It sends b.
6. P defines $b_1 := b \oplus b_2$ and continues to run the prover of Ligero using b_1 as the 3rd message to obtain next message c_1. It sends (b_1, c_1, b_2, c_2).[a]
7. V samples uniformly at random a string $\widetilde{b}$, which has the same length as the 5th message of Ligero. It sends $\widetilde{b}$.
8. P defines $\widetilde{b}_1 := \widetilde{b} \oplus \widetilde{b}_2$ and continues to run the prover of Ligero using $\widetilde{b}_1$ as the 5th message to obtain the final message $\widetilde{c}_1$. It sends $(\widetilde{b}_1, \widetilde{c}_1, \widetilde{b}_2, \widetilde{c}_2)$.
9. P and V execute sWIAoK where P proves that

$$\exists r \text{ s.t. } \mathsf{com} = \mathsf{SHCom}(h_1 \| b_1 \| \widetilde{b}_1; r) \ \vee \ \mathsf{com} = \mathsf{SHCom}(h_2 \| b_2 \| \widetilde{b}_2; r), \tag{1}$$

   where com is defined in Step 2b, $(b_1, \widetilde{b}_1)$ are defined in Step 6, and $(b_2, \widetilde{b}_2)$ are defined in Step 8. Note that the honest P uses the $(h_2, b_2, \widetilde{b}_2, r)$ as the witness to finish this sWIAoK, because this is exactly how it generated com in Step 2b.

Verifier's Decision: V accepts iff the following conditions hold:

1. $h_1 \oplus h_2 = h$, $b_1 \oplus b_2 = b$ and $\widetilde{b}_1 \oplus \widetilde{b}_2 = \widetilde{b}$,
2. both $(h_1, a_1, b_1, c_1, \widetilde{b}_1, \widetilde{c}_1)$ and $(h_2, a_2, b_2, c_2, \widetilde{b}_2, \widetilde{c}_2)$ are convincing w.r.t. the Ligero protocol, **and**
3. V accepts as the verifier of the Step 9 sWIAoK.

[a] Note that P does not really need to send b_2 because B can derive it from b_1 and b. We asks P to send it only for conceptual clearness.

We prove the following Lem. 5, which establishes the security of Prot. 10.

Lemma 5. *Let L and L' be $\mathcal{NP}$ languages. Protocol Π_{OR} (Prot. 10) is an interactive argument of knowledge for $L \vee L'$; furthermore, for any statement $(x, x') \in L \vee L'$ such that $x \in L$ and $x' \in L'$, Π_{OR} is statistically witness-indistinguishable.*

The completeness of Prot. 10 is straightforward. In the following, we first show the argument of knowledge property in Sec. C.1.1, and then establish the statistical WI property (when both x and x' are true) in Sec. C.1.2.

C.1.1 Proving Argument of Knowledge

We reduce the AoK property of Prot. 10 to that of Ligero. Specifically, fix any (deterministic[12]) cheating prover P^* which is able to convince the (honest) verifier of Prot. 10 with probability at least $\varepsilon(\lambda)$. We show how to *efficiently* construct a new PPT adversarial prover P' that convinces the Ligero verifier with some probability that is polynomially related to $\varepsilon(\lambda)$. This implies the AoK property of Prot. 10, because we can invoke the knowledge extractor of Ligero on P' to extract a valid witness.

The description of P' is given in Algo. 2.

Algorithm 2: Malicious Prover P' Attacking Ligero

The P' runs P^* internally, while externally interacting with the Ligero verifier V_{Ligero}. It proceeds as follows:

1. **(Obtaining the Prefix.)** P' executes Prot. 10 internally with P^* until the end of Step 2, where P' plays the role of the honest verifier. We denote the snapshot of P^* at this moment as τ, and use P^*_τ to denote the remaining strategy of P^* with the prefix τ.
2. **(Extracting Committed Value.)** Starting from this prefix τ, P' continues to run the internal P^*, playing the role of the Prot. 10 verifier honestly. At the end of this execution, if P^*'s proof is not accepted, P' aborts and outputs a special symbol Fail; otherwise, it runs the knowledge extractor of the final sWIAoK (i.e., Step 9) to extract a valid witness of the language defined in Eq. (1). We denote this (extracted) witness as $(\mathbf{h}, \mathbf{b}, \widetilde{\mathbf{b}}, \mathbf{r})$. Note that we should have $\mathsf{com}^* = \mathsf{SBCom}(\mathbf{h} \| \mathbf{b} \| \widetilde{\mathbf{b}}; \mathbf{r})$, where com^* is the Step 2 message contained in τ.
3. **(Interacting with V_{Ligero}.)** P' now starts interacting with V_{Ligero}.
   (a) First, P' picks a random bit $i \xleftarrow{\$} \{0, 1\}$. Looking ahead, P' will forward a session to the external V_{Ligero} according to this bit—if $i = 0$ (resp. $i = 1$), the left (resp. right) session will be forwarded out.
   (b) P' receives the 1st Ligero message h from V_{Ligero}. It then sets $h^* = h \oplus \mathbf{h}$, and feeds h^* to P^*_τ as the Step 3 message of the internal execution of Prot. 10.
   (c) P^*_τ responds with $(h^*_1, a^*_1, h^*_2, a^*_2)$ as the Step 4 message of Prot. 10. If $h^* \neq h^*_1 \oplus h^*_2$ or $h^*_{i-1} \neq \mathbf{h}$, P' aborts and outputs a special symbol Fail. Otherwise, we must have $h^*_i = h$. In the latter case, P' forwards a^*_i to V_{Ligero} as the 2nd message of Ligero.
   (d) P' receives the 3rd Ligero message b from V_{Ligero}. It then sets $b^* = b \oplus \mathbf{b}$, and feeds b^* to P^*_τ as the Step 5 message of the internal execution of Prot. 10.

[12] This is w.l.o.g. since P^* is non-uniform.

(e) P^*_τ responds with $(b^*_1, c^*_1, b^*_2, c^*_2)$ as the Step 6 message of Prot. 10. If $b^* \neq b^*_1 \oplus b^*_2$ or $b^*_{i-1} \neq \mathbf{b}$, P' aborts and outputs a special symbol Fail. Otherwise, we must have $b^*_i = b$. In the latter case, P' forwards c^*_i to V_{Ligero} as the 4th message of Ligero.

(f) P' receives the 5th Ligero message $\widetilde{b}$ from V_{Ligero}. It then sets $\widetilde{b}^* = \widetilde{b} \oplus \widetilde{\mathbf{b}}$, and feeds $\widetilde{b}^*$ to P^*_τ as the Step 7 message of the internal execution of Prot. 10.

(g) P^*_τ responds with $(\widetilde{b}^*_1, \widetilde{c}^*_1, \widetilde{b}^*_2, \widetilde{c}^*_2)$ as the Step 8 message of Prot. 10. If $\widetilde{b}^* \neq \widetilde{b}^*_1 \oplus \widetilde{b}^*_2$ or $\widetilde{b}^*_{i-1} \neq \widetilde{\mathbf{b}}$, P' aborts and outputs a special symbol Fail. Otherwise, we must have $\widetilde{b}^*_i = \widetilde{b}$. In the latter case, P' forwards $\widetilde{c}^*_i$ to V_{Ligero} as the 6th message of Ligero.

It follows immediately from the description that P' is an expected (also see Rmk. 6) PPT machine. We only need to show that P' convinces V_{Ligero} with some probability polynomially related to $\varepsilon(\lambda)$. To do that, observe the following facts:

1. By a standard averaging argument, it follows that for $\varepsilon/2$ fractions of all possible τ, P^*_τ (defined in Step 1) gives a convincing proof with probability at least $\varepsilon/2$.
2. It is easy to see that the view of the internal P^* is identically distributed as its view in a real execution of Prot. 10, regardless of the bit i picked by P'. Consider the following Good event:
   - Good: for the $(h^*_1, h^*_2, b^*_1, b^*_2, \widetilde{b}^*_1, \widetilde{b}^*_2)$ defined in Steps 3c, 3e and 3g, there exists a bit $j \in \{0, 1\}$ s.t. $(h^*_{j-1}, b^*_{j-1}, \widetilde{b}^*_{j-1}) = (\mathbf{h}, \mathbf{b}, \widetilde{\mathbf{b}})$. (Note that this implies $(h^*_j, b^*_j, \widetilde{b}^*_j) = (h, b, \widetilde{b})$, where $(h, b, \widetilde{b})$ come from the external V_{Ligero}.)

   We know that $\neg$Good can only happen with probability $\mathsf{negl}(\lambda)$—Otherwise, P^* breaks the the computational binding property of SHCom.
3. As explained earlier, all P^* does is simply pick one of the two sessions (i.e., the session specified by i) to relay its messages to the external V_{Ligero}. Since i is picked uniformly at random from $\{0, 1\}$, it holds that $i = j$ (recall that j is in $\{0, 1\}$ also) with probability exactly $1/2$.

Therefore, the outside V_{Ligero} will be convinced with probability at least

$$\varepsilon/2 \cdot \varepsilon/2 \cdot (1 - \mathsf{negl}(\lambda)) \cdot 1/2 = \varepsilon^2/8 - \mathsf{negl}(\lambda),$$

which is polynomially related to ε. We can therefore invoke the knowledge extractor of Ligero on P' to extract the witness. This finishes the proof of the argument of knowledge property of Prot. 10.

Remark 6 (On the Expected Running Time of P'). One subtlety of the above argument is that in the standard definition of AoK, the knowledge extractor needs access to a *strictly* polynomial time P'. However, the P' we constructed runs in *expected* polynomial time. This can be addressed by a standard technique—simply pick a large enough polynomial as an upper-bound for the running time of P', and truncate the machine P' if it runs beyond this upper-bound. Then, by Markov inequality, P' still convinces V_{Ligero} with polynomial-inverse probability (i.e., it is good enough for Ligero's knowledge extractor to succeed).

C.1.2 Proving Statistical WI

At a high-level, we will define 9 hybrids $\{H_0, \ldots, H_8\}$, where H_0 is identical to the real execution of Prot. 10 with P using the witness w for x, and H_8 is identical to the real execution of Prot. 10 with P using the witness w' for x'. For any $i \in \{0, \ldots, 8\}$, the output of H_i, dubbed $\mathsf{OUT}(H_i)$, is

defined to be the output of the malicious V^* in H_i. Then, the proof will be done once we show that the outputs of all adjacent hybrids are statistically close.

Hybrid H_0: This is the real execution of Prot. 10 with P using the witness w for x.

Hybrid H_1: This hybrid is identical to the previous one, except for the following changes: At the beginning of Step 9, this hybrid checks if there exists r' such that $\mathsf{com} = \mathsf{SHCom}(h_1 \| b_1 \| \widetilde{b}_1; r')$. If not, it aborts and output Fail; otherwise, it finishes the execution identically to H_0. Some remarks about this hybrid follow:

1. This hybrid is not efficient;
2. There could exist more than one r' satisfying the above requirement. In that case, any r' will work.
3. Starting from this hybrid, the final sWIAoK (i.e. Step 9) will be executed only if there exists such an r' that can "explain" com to the commitment of $h_1 \| b_1 \| \widetilde{b}_1$.

$\underline{\mathsf{OUT}(H_0) \overset{\mathrm{s}}{\approx} \mathsf{OUT}(H_1)}$: First, observe that these two hybrids are identical conditioned on that there exists an r' as required above. Also, it follows from the statistical hiding property of SHCom that there must exist such an r' except for with negligible probability. Therefore, we have $\mathsf{OUT}(H_1) \overset{\mathrm{s}}{\approx} \mathsf{OUT}(H_2)$.

Hybrid H_2: This hybrid is identical to the previous one except that in the Step 9 sWIAoK, the P uses $(h_1, b_1, \widetilde{b}_1, r')$ as the witness (see Item 3).

$\underline{\mathsf{OUT}(H_1) \overset{\mathrm{s}}{\approx} \mathsf{OUT}(H_2)}$: This follows immediately from the statistical WI property of sWIAoK.

Hybrid H_3: This hybrid is identical to the previous one except that in Step 2b, the P generates com as $\mathsf{com} = \mathsf{SHCom}(0^{|h_2|} \| 0^{|b_2|} \| 0^{|\widetilde{b}_2|}; r)$. Note that in this hybrid, P only needs to know the length of $(h_2, b_2, \widetilde{b}_2)$, which is public information depends only on λ. Also note that $|h_2| = |h_1| = |h|$, $|b_2| = |b_1| = |b|$, and $|\widetilde{b}_2| = |\widetilde{b}_1| = |\widetilde{b}|$.

$\underline{\mathsf{OUT}(H_2) \overset{\mathrm{s}}{\approx} \mathsf{OUT}(H_3)}$: This follows immediately from the statistical hiding property of SHCom.

Hybrid H_4: This hybrid is identical to the previous one except for the way the messages $(h_2, a_2, b_2, c_2, \widetilde{b}_2, \widetilde{c}_2)$ are generated in Step 2a. Recall that in H_3 (and also H_0 to H_2), these messages are generated by $\mathsf{HVSim}(x')$. In the current hybrid, we give the witness w' to H_4. It internally emulates a real execution of Ligero where the prover uses witness w' to give a proof for x'. We use the $(h_2, a_2, b_2, c_2, \widetilde{b}_2, \widetilde{c}_2)$ from the transcript of this emulated execution in place of those generated by HVSim in the previous hybrid.

$\underline{\mathsf{OUT}(H_3) \overset{\mathrm{s}}{\approx} \mathsf{OUT}(H_4)}$: This follows immediately from the statistical HVZK property of Ligero.

Hybrid H_5: This hybrid is exactly the same as H_4, but we interpret it in a different way: Note that in H_5, both the left and right session consists of real Ligero executions—the left session is given by the (honest) Ligero prover $P_{\mathsf{Ligero}}(x, w)$, resulting in a transcript $(h_1, a_1, b_1, c_1, \widetilde{b}_1, \widetilde{c}_1)$; the right session is given by the Ligero prover $P_{\mathsf{Ligero}}(x', w')$, resulting in a transcript $(h_2, a_2, b_2, c_2, \widetilde{b}_2, \widetilde{c}_2)$.

In H_4, we "view" it as the following game: the hybrid internally emulates the *right* session by picking random $(h_2, b_2, \widetilde{b}_2)$, and the $(h_1, b_1, \widetilde{b}_1)$ are generated by an XOR operation such that $h_1 \oplus h_2 = h$, $b_1 \oplus b_2 = b$, and $\widetilde{b}_1 \oplus \widetilde{b}_2 = \widetilde{b}$, where $(h, b, \widetilde{b})$ are picked by $V*$. In this hybrid, we view it in the following manner: the hybrid internally emulates the *left* session by picking random $(h_1, b_1, \widetilde{b}_1)$, and then generates $(h_2, b_2, \widetilde{b}_2)$ by an XOR operation such that $h_1 \oplus h_2 = h$, $b_1 \oplus b_2 = b$, and $\widetilde{b}_1 \oplus \widetilde{b}_2 = \widetilde{b}$. Obviously, these two interpretations are equivalent.

$\mathsf{OUT}(H_4) \overset{\text{id}}{=} \mathsf{OUT}(H_5)$: As mentioned above, H_5 is just a different interpretation of H_4, without any real changes.

Hybrid H_6: This hybrid is identical to the previous one, except that in Step 2a, P computes $(h_1, a_1, b_1, c_1, \widetilde{b}_1, \widetilde{c}_1) \leftarrow \mathsf{HVSim}(x)$.

$\mathsf{OUT}(H_5) \overset{\text{s}}{\approx} \mathsf{OUT}(H_6)$: H_6 simply replaces the messages yielded by a real left session by the output of HVSim. Thus, the indistinguishability follows by the statistical HVZK property of Ligero. This is analogous to the (reverse) hop from H_3 to H_4.

Hybrid H_7: This hybrid is identical to the previous one except that in Step 2b, P generates com as $\mathsf{com} = \mathsf{SHCom}(h_1 \| b_1 \| \widetilde{b}_1; r)$.

$\mathsf{OUT}(H_6) \overset{\text{s}}{\approx} \mathsf{OUT}(H_7)$: This follows immediately from the statistical-hiding property of SHCom. This is analogous to the (reverse) hop from H_2 to H_3.

Hybrid H_8: Recall that starting from H_1, the hybrid checks if there is an r' satisfying $\mathsf{com} = \mathsf{SHCom}(h_1 \| b_1 \| \widetilde{b}_1; r')$, right before the Step 9 sWIAoK starts. In H_8, the hybrid does not check the existence of r' anymore. It simply use the $(h_1, b_1, \widetilde{b}_1, r)$ as the witness to finish the sWIAoK. This can be done because in H_7, the com in Step 2b is generated as $\mathsf{SHCom}(h_1 \| b_1 \| \widetilde{b}_1; r)$; thus, $(h_1, b_1, \widetilde{b}_1, r)$ constitute a valid witness for the language specified in Eq. (1).

Observe that H_8 is exactly the real execution of Prot. 10 with the honest prover using the right witness w' (and thus the left session is simulated by HVSim).

$\mathsf{OUT}(H_7) \overset{\text{s}}{\approx} \mathsf{OUT}(H_8)$: The only difference is the witness used in the final sWIAoK. Thus, the indistinguishability follows form the statistical WI property of sWIAoK. Note that we do not claim that these two hybrids are identical, as the r' used in H_7 may not be the same as the r used in H_8; but both of them serve as valid random tape that explains com as a SHCom to $h_1 \| b_1 \| \widetilde{b}_1$.

The above argument implies $\mathsf{OUT}(H_1) \overset{\text{s}}{\approx} \mathsf{OUT}(H_8)$, which finishes the proof for the statistical WI property of Prot. 10.

C.1.3 Regarding Malicious-Verifier Zero Knowledge

It is easy to see that Π_{OR} (Prot. 10) is also an honest-verifier ZK argument (of knowledge). Using the Goldreich-Kahan style argument [GK96] (as done in [IMS12, AHIV17]), we can convert it to a fully-secure ZK argument, i.e., against malicious verifiers. We denote the resulting protocol as Π'_{OR}, and present it formally in Prot. 11. Note that Π'_{OR} will not be an argument of knowledge any more. But it is good enough to be used in the consistency proof of the $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$ commitment (as this application does not require AoK property).

Protocol 11: Π'_{OR}: Maliciously-Secure ZK from OR-Composition

Let L, L', (x, x') and Π be the same as in Prot. 10. Our Π'_{OR} for proving $(x, x') \in L \vee L'$ proceeds as follows:

1. P and V execute a two-round statistically-hiding commitment scheme SHCom, where V commits to the messages h, b, and $\widetilde{b}$ separately in parallel (in form of the verifier's challenge in Π_{OR}).
2. P and V then engage in executing Π_{OR} (Prot. 10) with the following modifications:
   - In Step 3 of Π_{OR}, instead of sending h, V decommits to h; P continues only if this decommitment is valid.
   - In Step 5 of Π_{OR}, instead of sending b, V decommits to b; P continues only if this decommitment is valid.
   - In Step 7 of Π_{OR}, instead of sending $\widetilde{b}$, V decommits to $\widetilde{b}$; P continues only if this decommitment is valid.

   Everything else remains the same as in Π_{OR}.

Lemma 6. *Let $L, L' \in \mathcal{NP}$. Protocol Π'_{OR} is zero-knowledge argument for $L \vee L'$.*

Proof. The proof follows from Goldreich-Kahan technique [GK96], as done in [IMS12, AHIV17]). □

C.2 Efficient Instantiation

We now discuss how to implement Π_{OR} (Prot. 10) efficiently. As shown in Appx. C.1, Π_{OR} is simply the plain OR-composition of Ligero plus the following two extra gadgets:

- A SHCom at the beginning;
- A sWIAoK at the end.

To measure its efficiency, we mainly focus on the cost of these two gadgets. This is because the cost due to the plain OR-composition cannot be avoid; thus, we want to minimize the additional cost. We will call the additional cost due to the above two gadgets *the overhead compared with the plain OR-Composition*.

Committing to Hash Tags. First, we note that the message committed by the SHCom in Step 2b is $h_2 \| b_2 \| \widetilde{b}_2$, which is of the same length as the corresponding Ligero messages (i.e. $|h| + |b| + |\widetilde{b}|$). Since we need to execute a statistical WIAoK on this commitment in Step 9, we want to avoid committing long strings. We use a simple trick to shorten the committed message to be of length 3λ. To do that, we ask the prover to use a CRHF $H : \{0,1\}^* \to \{0,1\}^\lambda$:

- In Step 2b, we ask the prover to compute the hash tags: $\mathsf{tag}_{h_2} = H(h_2)$, $\mathsf{tag}_{b_2} = H(b_2)$ and $\mathsf{tag}_{\widetilde{b}_2} = H(\widetilde{b}_2)$; and instead of committing to $h_2 \| b_2 \| \widetilde{b}_2$, the prover commits to $\mathsf{tag}_{h_2} \| \mathsf{tag}_{b_2} \| \mathsf{tag}_{\widetilde{b}_2}$.
- In Step 4, the prover additionally computes $\mathsf{tag}_{h_1} = H(h_1)$, and additionally sends $(\mathsf{tag}_{h_1}, \mathsf{tag}_{h_2})$; The verifier additionally checks if $\mathsf{tag}_{h_1} = H(h_1)$ and $\mathsf{tag}_{h_2} = H(h_2)$.

– In Step 6, the prover additionally computes $\mathsf{tag}_{b_1} = H(b_1)$, and additionally sends $(\mathsf{tag}_{b_1}, \mathsf{tag}_{b_2})$; The verifier additionally checks if $\mathsf{tag}_{b_1} = H(b_1)$ and $\mathsf{tag}_{b_2} = H(b_2)$.

– In Step 8, the prover additionally computes $\mathsf{tag}_{\widetilde{b}_1} = H(\widetilde{b}_1)$, and additionally sends $(\mathsf{tag}_{\widetilde{b}_1}, \mathsf{tag}_{\widetilde{b}_2})$; The verifier additionally checks if $\mathsf{tag}_{\widetilde{b}_1} = H(\widetilde{b}_1)$ and $\mathsf{tag}_{\widetilde{b}_2} = H(\widetilde{b}_2)$.

In this way, the sWIAoK in in Step 9 will be done for the following short statement:

$$\exists r \text{ s.t. } \mathsf{com} = \mathsf{SHCom}(\mathsf{tag}_{h_1} \| \mathsf{tag}_{b_1} \| \mathsf{tag}_{\widetilde{b}_1}; r) \ \vee \ \mathsf{com} = \mathsf{SHCom}(\mathsf{tag}_{h_2} \| \mathsf{tag}_{b_2} \| \mathsf{tag}_{\widetilde{b}_2}; r).$$

Note that $|\mathsf{tag}_{h_2}| + |\mathsf{tag}_{b_2}| + |\mathsf{tag}_{\widetilde{b}_2}| = 3\lambda$. Thus, the honest prover only needs to commits to a string of length 3λ.

In summary, the efficiency comes from the fact that the task of verifying the hash tags are done *outside of the circuit* to be proven by ZKAoK.

Overhead of Prot. 10. To analyze the efficiency of Prot. 10, we can view it as the following 3 stages:

1. **Committing to P's Share.** In Steps 1 and 2, P commits to a string of length 3λ (i.e. using the above trick of committing to hash tags) using SHCom.
2. **Plain OR Composition of Ligero.** Step 1 to Step 8 is actually a standard OR-composition of the Ligero protocol.
3. **Statistical WIAoK:** This is Step 9, where P conducts a sWIAoK on the commitment generated in Stage 1.

In terms of efficiency, the cost of Stage 1 is negligible compared with that of the last two stages. As mentioned earlier, the cost of Stage 2 is something we have to pay. In the following, we discuss the cost of Stage 3.

As mentioned earlier, we will show how to construct the sWIAoK used in Stage 3 from the Ligero protocol later in Appx. C.3. But looking ahead, our sWIAoK construction will enjoy the same efficiency of the plain Ligero. Thus, the cost of Stage 3 is the same as running Ligero to prove the consistency of a statistically-binding commitment (to a length-3λ message). We use the Halevi-Micali scheme to implement SHCom, where we use SHA256 as the underlying CRHF. As discussed in Appx. A.3, to commit to a length-3λ message, the cost consists of

– Using SHA256 to hash a length $(13\lambda + 4)$-bit string, **and**

– Evaluating a pair-wise independent hash with domain $\{0,1\}^{10\lambda+4}$ and range $\{0,1\}^{3\lambda}$.

The size of these computation is upper-bounded by the size of sixteen SHA256 circuits. According to [AHIV17, Figures 3 and 4], running a Ligero protocol on the above computation has the following cost:

– **P's Running Time:** 32 milliseconds

– **V's Running Time:** 3.2 milliseconds

– **Communication Cost:** 6400KB

C.3 Constructing the Statistically WIAoK

C.3.1 Construction

We now construct the sWIAoK that can be used in Step 9 of Prot. 10. Recall that this sWIAoK is only used to prove the consistency of Halevi-Micali's SHCom, which is nothing but several calls to a CRHF (which will be implemented using SHA256). Thus, we do not need to do the OR-composition (as we did for Prot. 10) anymore—Ligero is already very fast for this task. But note that Ligero is only a *honest-verifier* statistical AKZoK, not a statistical WIAoK yet. Therefore, our goal is to make minimal modifications to Ligero to convert it to a statistical WIAoK.

To do that, first observe that Ligero is already an *honest-verifier* statistical WIAoK. This means that if we can somehow enforce the honest behavior of the verifier, we will obtain the desired statistical WIAoK. This turns out easy because Ligero is a public-coin protocol: instead of asking V to send its challenge coins, we simply use a coin-tossing to determine these coins. Moreover, since we are going for WI (in contrast to ZK), we only need a IND-secure coin-tossing (in contrast to a simulatable coin-tossing). This can be done very efficiently.

In more detail[13], we ask P to commit to its shares h_1, b_1, and $\widetilde{b}_1$ before Ligero starts, using a statistically-hiding commitment SHCom. Then, P and V execute Ligero with the following modifications: when Ligero asks V to send h, V sends a h_2 instead; P then decommits to h_1; both parties will set $h = h_1 \oplus h_2$ and use this h to execute Ligero. The other two challenges b and $\widetilde{b}$ will be determined similarly. We present the formal description in Prot. 12.

Protocol 12: Statistically WIAoK from Ligero

Let $(h, a, b, c, \widetilde{b}, \widetilde{c})$ denote the 6 messages of the Ligero protocol (Prot. 8). Fix an instance $x \in \mathcal{L}$ and a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$. The prover P has (x, w) as input, and the verifier V has x as its input. The protocol proceeds as follows:

1. V sends the first message for SHCom.
2. P computes the following messages:
   (a) P samples $h_1 \xleftarrow{\$} \{0,1\}^{|h|}$, $b_1 \xleftarrow{\$} \{0,1\}^{|b|}$, and $\widetilde{b}_1 \xleftarrow{\$} \{0,1\}^{|\widetilde{b}|}$.
   (b) P samples random tapes (r_1, r_2, r_3) and generates:
   $$\mathsf{com}_1 = \mathsf{SHCom}(h_1; r_1),\ \mathsf{com}_2 = \mathsf{SHCom}(b_1; r_2),\ \mathsf{com}_3 = \mathsf{SHCom}(\widetilde{b}_1; r_3).$$
   P sends $(\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$ to V.
3. V samples $h_2 \xleftarrow{\$} \{0,1\}^{|h|}$ and sends h_2 to P.
4. P proceeds as follows:
   (a) P computes $h = h_1 \oplus h_2$.
   (b) P computes the 2nd Ligero message a, assuming the 1st Ligero message is h.
   P sends (h_1, r_1, a) to V.
5. V computes $h = h_1 \oplus h_2$. It then checks if (1) r_1 validly decommits com_1 to h_1, and (2) a is a valid 2nd Ligero message consistent with h, using the honest Ligero verifier's algorithm. If not, V rejects; otherwise, V samples $b_2 \xleftarrow{\$} \{0,1\}^{|b|}$ and sends b_2 to P.

[13] Recall that the 6 messages of Ligero are denoted as $(h, a, b, c, \widetilde{b}, \widetilde{c})$.

6. P proceeds as follows:
   (a) P computes $b = b_1 \oplus b_2$.
   (b) P computes the 4th Ligero message c, assuming the previous Ligero messages are (h, a, b).

   P sends (b_1, r_2, c) to V.
7. V computes $b = b_1 \oplus b_2$. It then checks if (1) r_2 validly decommits com_2 to b_1, and (2) c is a valid 4th Ligero message consistent with (h, a, b), using the honest Ligero verifier's algorithm. If not, V rejects; otherwise, V samples $\widetilde{b}_2 \xleftarrow{\$} \{0,1\}^{|\widetilde{b}|}$ and sends $\widetilde{b}_2$ to P.
8. P proceeds as follows:
   (a) P computes $\widetilde{b} = \widetilde{b}_1 \oplus \widetilde{b}_2$.
   (b) P computes the 6th Ligero message $\widetilde{c}$, assuming the previous Ligero messages are $(h, a, b, c, \widetilde{b})$.

   P sends $(\widetilde{b}_1, r_3, \widetilde{c})$ to V.

Verifier's Decision: V computes $\widetilde{b} = \widetilde{b}_1 \oplus \widetilde{b}_2$. It then checks if (1) r_3 validly decommits com_3 to $\widetilde{b}_1$, and (2) $\widetilde{c}$ is a valid 6th Ligero message consistent with $(h, a, b, c, \widetilde{b})$, using the honest Ligero verifier's algorithm. If not, V rejects; otherwise, V accepts.

The security of this protocol is established by Lem. 7.

Lemma 7. *Prot. 12 is a statistical witness-indistinguishable argument of knowledge.*

Proof. Completeness follows straightforwardly. It turns that the both the AoK property and statistically WI property of Prot. 12 can be proven in almost the same way as those of Prot. 10. Therefore, we only provide a sketch.

AoK Property. Similar as in Sec. C.1.1, we reduce the AoK property to that of Ligero: given a P^* that attacks Prot. 12, we construct a P' attacking Ligero; and the advantage of P' is polynomially related to that of P^*. P' runs P^* internally once, to learn the h_1, b_1, and $\widetilde{b}_1$ committed by P^* at the beginning. With this information, P' will rewind P^* and enforce the coin-tossing by setting $h_2 = h_1 \oplus h$ (similarly for b_2 and $\widetilde{b}_2$), where h is the external V_{Ligero}'s challenge. In this way, P^*'s answer can be used to reply the external V_{Ligero}. Following the same argument as in Sec. C.1.1, one can show that P' wins with probability polynomially related to that of P^*.

Statistical WI Property. To prove statistical WI, we consider the following *inefficient* hybrid: at the beginning, ask P to commit to arbitrary values, say all-0 strings, instead of committing to h_1, b_1, and $\widetilde{b}_1$. Then, P runs the HVZK simulator for Ligero to get the simulated transcript $(h, a, b, c, \widetilde{b}, \widetilde{c})$. Next, P will enforce the coin-tossing such that it can use the simulated transcript to answer V^*'s challenge. To do that, once P receives h_2 from V^*, it will "decommit" SHCom to $h_1 = h_2 \oplus h$. This can be done because SHCom is statistically-hiding, so it can be "explained" to any other messages[14]. Then, P can simply use the simulated a as the response to h. P will handle the (b, c) and $(\widetilde{b}, \widetilde{c})$ slots in a similar way.

[14] We note that this step is not efficient. But this is fine as we are proving *statistical* WI. The hybrids can run in exponential time, as long as they are statistically close.

The above procedure yields a view of V^* that is statistically close to that of the real execution, while P does not need to know the witness (though it needs exponential time to equivocate the SHCom to enforce the desired coin-tossing result). Thus, V^*'s view should be statistically close when P switches from one witness to another. This argument can be formalized in exactly the same way as we did in Sec. C.1.2. □

C.3.2 Efficiency Analysis

Compared with the plain Ligero protocol, the only overheads of Prot. 12 are

- three calls of SHCom in Round 2.
- The decommitment of the above three calls of SHCom (that will be checked by V) in Rounds 5 and 7 and the final verifier's decision stage.

Using Halevi-Micali SHCom, these are simply some SHA256 evaluations. Note that a normal desktop can hash 200MB data per second using SHA256. Thus, the cost of Prot. 12 can be thought as being identical to that of Ligero.

D Performance Analysis

In this section, we discuss the asymptotic and concrete performance of our protocols. We do not focus on full optimization of the concrete numbers as we do not intend to provide an out-of-shelf implementation. Rather, our primary goal is to show that our technique brings non-malleable commitments and ZK protocols (in the plain model) within reach of practical computing.

We implement the following building blocks using AES and SHA256:

- **Statistically-Binding Commitment (SBCom).** We use the construction suggested from [Nao91] for messages longer than a bit. Instead of running the statistically-binding bit commitment scheme in parallel over the message, we run it in parallel over a seed for a PRG that is then used as a one-time pad for the message. The PRG is implemented using AES in CTR mode.
- **Statistically-Hiding Commitment (SHCom).** We use the construction from [HM96] with SHA256 as the underlying CRHF (see Appx. A.3).
- **Collision-Resistant Hash Functions.** Some of our protocols (e.g. Prot. 10) require collision-resistance hash *families*. But standard hash functions (in Minicrypt) are usually un-keyed. One way in practice is to sample a random string r as the key, and hash it together with the message using SHA256, i.e. $h(m) = \text{SHA256}(r\|m)$. Another way is to restate our claims regarding the security of related protocols following the suggestions from [Rog06]. We take the first approach.

For clarity of presentation, we will measure the asymptotics in terms of number of AES_λ and SHA256_λ evaluations.[15] We will denote by AES_λ an evaluation of the AES block cipher on a λ-bit string (padded appropriately). We define SHA256_λ in the same manner for SHA256.

Recall that our IB-NMC (Prot. 3) is based on $\Pi^{\mathsf{Mini}}_{\text{BGRRV}}$. It runs a proof in the second stage to check consistency of the preamble stage. The size of the circuit for checking consistency depends on k and λ, where k is the bit length of the identities. We denote the size of the consistency circuit

[15] We remark that we are counting operations in terms of λ-sized inputs. To count operations on fixed block sizes, we simply multiply the values by B/λ for block length B.

by $C_{\mathsf{cons}(k,\lambda)}$. When clear from the context, we will drop k and λ to avoid cluttering the notation. We also let $C_{\mathrm{AES}(\cdot)}$ be the AES circuit size, $C_{\mathsf{add}(\cdot)}$ be the adder size, $C_{\mathsf{mul}(\cdot)}$ be the multiplier size, where all of these circuit sizes are dependent on the value of the input length which in turn only depend on λ. Finally, let C_{eq} be the "equality testing" circuit; this circuit on input (c, v, r) tests whether c is a commitment to v with randomness r.

D.1 Asymptotic Complexity

We first compute the complexities of our main component: the simulation-sound zero-knowledge argument of knowledge. The costs for final non-malleable protocols will be pretty close and derived later.

D.1.1 Complexity of Our Simulation-Sound ZK

Computational complexity of SSZK. We analyze the computational complexity of the prover and verifier for each stage of this protocol. We assume randomness is for free.[16]. Note that protocol Π_{SS} (Prot. 4) is obtained by instantiating Prot. 2 with proper building blocks.

Step 1 is the ExtCom. The verifier computes 2λ SBCom, one for each share. Each commitment is to a value with a length of λ. Using the our implementation of the SBCom, this requires 3λ runs of AES_λ in total.

Step 2 is the IB-NMC shown in Prot. 3, which is obtained by modifying Prot. 1. Recall that in this case, the prover uses the witness for a statement y to run the IB-NMC and simulates the consistency proof for Ligero′ (which in turn involves running HVSim of Ligero). The two main parameters to IB-NMC are k, the length of the identities, and q, the length of the prime in bits. We can set $q = 2\lambda$, which follows the original paper [BGR+15]. To compute the size of the circuit for the consistency proof, we see that we have to run the AES circuit as well as run circuits to compute dot products as per the protocol description. We parameterize the consistency circuit size in terms of $C_{\mathrm{AES}(q)}, C_{\mathsf{add}(q)}$, and $C_{\mathsf{mul}(q)}$. We construct the dot product circuit from the adders and multipliers, all corresponding to our prime size 2λ. If we compose in the standard way, we end up needing about k^2 adders and multipliers, as well as k AES_q circuits. Additionally, we need on the order of k additional equality checking circuits, which are of size on the order of λ.

However, we can shrink the size of this circuit by taking advantage of the fact that we are only committing to a message of length 2λ (or of the form $(m, 0, \ldots, 0)$) using $\Pi^{\mathsf{Mini}}_{\mathrm{BGRRV}}$, which commits vectors from $\mathbb{Z}_q^\ell$ (see Prot. 1). Due to this special message structure, the consistency circuit will only require on the order of k adders and multipliers, instead of k^2. Consequently, $C_{\mathsf{cons}(k,\lambda)} = O\big(k(C_{\mathsf{add}(2\lambda)} + 2C_{\mathsf{mul}(2\lambda)} + 3C_{\mathrm{AES}(2\lambda)})\big)$. The O notation here is merely to take into account trivial computations such as copying wire values to feed into multiple gates, and so on. We will ignore these trivial computations and drop the O notation.

Given C_{cons}, we get that the number of SHA256 operations we will need is on the order of $\lambda\sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}}$. For the honest proof for statement y, the prover must compute the full Merkle tree; thus, for a circuit with s gates, the prover runs $s\lambda$ executions of SHA256. We recall that for

[16] It could be generated offline or through a run of AES. A modern laptop can generate a cryptographically-secure random string of length 600MB in one second. These costs do not significantly change the analysis but are cumbersome to incorporate

the IB-NMC, we have to add malicious security and thus, we incur the cost of an extra SHCom for the challenge bits. However, the length of the challenge is on the order of $\lambda \log C_{\mathsf{cons}}$, which is asymptotically small compared to our previous cost.

Step 3 is the decommitment of ExtCom from Stage-1. There is no new verifier computation. The prover verifies the remaining decommitments and that they combine to σ. With our instantiation of SBCom, this requires running 2λ runs of AES_λ.

Step 4 is the sWIAoK instantiated by Π_{OR} (Prot. 10). Recall that in this case, the prover uses the witness for the given statement x and runs the HVZK simulator HVSim (of Ligero) for C_{eq}. To construct C_{eq}, we need the circuit for AES_λ as well as an equality checking circuit, which is on the order of λ. We will use $C_{\mathsf{eq}} = C_{\mathrm{AES}_\lambda} + \lambda$ as our estimate for the size of this circuit. Since only the HVSim is required, the main computation is in computing the hashes for the queried nodes, but this is only on the order of $\lambda\sqrt{C_{\mathsf{eq}}} \log C_{\mathsf{eq}}$. The prover for the statement x would still have to run hashes for the whole tree, which is on the order of $s\lambda$. In Prot. 10, we also ask the prover to commit to $h\|b_2\|\widetilde{b}_2$ using SHCom and run Prot. 12 for its consistency. This add at most 16 SHA256 calls, which can be ignored asymptotically.

We summarize the complexity of each stage as well as the overall complexity of SSZK in Table 3.

Communication complexity of SSZK. We now analyze the communication complexity of the prover and verifier for each stage of this protocol. We remark that the field size used in this protocol is on the order of 2^λ; thus, the number of bits required to send one field element is on the order of λ.

Step 1 is the ExtCom. The verifier sends 3λ commitments, where each commitment is to a value with a length of λ. Then, the prover sends λ challenge bits, upon which the verifier opens the corresponding λ commitments. In total, the communication complexity of this stage is on the order of λ^2 bits.

Step 2 is the IB-NMC. The parameters are the same as described in the computational complexity. In the preamble, $k^2\lambda$ bits are exchanged between both the verifier and the prover. For the consistency proof, the communication complexity corresponding to the consistency proof part is the same as that of an honest Ligero execution for the consistency circuit, i.e., on the order of square root of the circuit size. Thus, the total communication complexity of IB-NMC is the sum of the communication for the preamble and the communication for the proof stage; the latter is one honest Ligero proof plus one simulated consistency proof. Moreover, we also ask the prover to commit to $h\|b_2\|\widetilde{b}_2$ using SHCom and run Prot. 12 for its consistency. This add $\lambda + \lambda\sqrt{C_{\mathrm{SHA}}}$ communication cost due to Ligero.

Step 3 is the decommitment of ExtCom from Step 1. The verifier sends the decommitment information, which is on the order of λ^2 bits.

Step 4 is the sWIAoK. The communication complexity analysis is the same as that for the final phase of Step 2 except that, instead of the consistency circuit, we have the equality checking circuit. Thus, the communication complexity of this stage is $\lambda(1 + \sqrt{s} + \sqrt{C_{\mathsf{eq}}} + \sqrt{C_{\mathrm{SHA}}})$.

A summary of communication complexity appears in Table 3.

Table 3: Asymptotic complexity of SSZK. Only dominating terms are shown in each cell.

Stage	Description	#AES$_\lambda$	#SHA256$_\lambda$	Communication
ExtCom (Step 1 & 3)	Total	λ	0	λ^2
IB-NMC (Step 2)	Preamble	0	0	$k^2\lambda$
	"$x \in L$"	0	s	$\lambda\sqrt{s}$
	Fake Ligero$'$ (Consistency)	0	$\sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}}$	$\lambda(\sqrt{C_{\mathsf{cons}}}+\sqrt{C_{\text{SHA}}})$
sWIAoK (Step 4)	Preamble	0	16	λ
	"$x \in L$"	0	s	$\lambda\sqrt{s}$
	Fake Ligero (Equality)	0	$\sqrt{C_{\mathsf{eq}}}\log C_{\mathsf{eq}}$	$\lambda\sqrt{C_{\mathsf{eq}}}$
	sWIAoK from Ligero	0	16	$\lambda\sqrt{C_{\text{SHA}}}$
Full SSZK	Total	λ	$2s+\sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}}+\sqrt{C_{\mathsf{eq}}}\log C_{\mathsf{eq}}$	$\lambda(\lambda+k^2+2\sqrt{s}+\sqrt{C_{\mathsf{cons}}}+\sqrt{C_{\mathsf{eq}}}+2\sqrt{C_{\text{SHA}}})$

D.1.2 Complexity of Our NMZK and NMCom

We can calculate the computational and communication complexity of these protocols in the same manner as done for SSZK. We do not show the calculations for each stage anymore for these protocols, because we already did it for their major component—SSZK dominates. We directly present a summary of dominating terms for both of these protocols in Table 4.

Table 4: Asymptotic complexity of NMZK, Ligero, and NMCom. Only dominating terms are shown in each cell. Plain Ligero is shown for comparison purposes.

Protocol	#AES$_\lambda$	#SHA256$_\lambda$	Communication
NMZK	λ	$2s+\sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}}+\sqrt{C_{\mathsf{eq}}}\log C_{\mathsf{eq}}+2C_{\mathsf{eq}}$	$\lambda(\lambda+k^2+2\sqrt{s}+\sqrt{C_{\mathsf{cons}}}+3\sqrt{C_{\mathsf{eq}}}+2\sqrt{C_{\text{SHA}}})$
NMCom	1	$2s+\sqrt{C_{\mathsf{cons}}}\log C_{\mathsf{cons}}+\sqrt{C_{\mathsf{eq}}}\log C_{\mathsf{eq}}+3C_{\mathsf{eq}}$	$\lambda(\lambda+k^2+2\sqrt{s}+\sqrt{C_{\mathsf{cons}}}+4\sqrt{C_{\mathsf{eq}}}+2\sqrt{C_{\text{SHA}}})$
Ligero	0	s	$\lambda\sqrt{s}$

Note that the cost of field operations in our case corresponds to the simulated ZKIPCP transcript, which is the square root of the circuits involved. This cost is insignificant in our case compared to the hash function evaluations corresponding to the simulated Merkle tree paths. We remark that the honest Ligero execution requires field operations on the order of the size of the circuit, which is not shown as part of the stand-alone Ligero computational costs.

Table 5: Sample times and communication for different parameter choices.

Param.	**NMZK**			**NMCom**		
(k, λ)	P time (s)	V time (s)	Comm. (MB)	P time (s)	V time (s)	Comm. (MB)
(32, 40)	1.68	0.74	19.68	2.52	1.12	19.74
(32, 80)	3.56	1.49	24.88	4.68	2.06	24.97
(64, 80)	5.04	2.23	28.84	6.72	3.09	28.93

D.2 Practical Performance

We now discuss the numbers we use for our concrete evaluation. Our results for NMZK are based on proving a witness for SHA256.

From [AHIV17], we are given that SHA256 has around 33,000 gates and takes approximately 0.14 seconds for the prover and 0.06 seconds for the verifier for soundness error of 2^{-40} [AHIV17]. If we consider a soundness error of 2^{-80}, then the costs double and the time would take approximately 0.28 and 0.12 seconds respectively instead.[17]

To work with concrete times, we measure the throughput of each on an Intel Core i7 9th Gen 9700K (3.60 GHz) processor. From this, we approximate the processing speed of AES to be 1 billion bytes per second (assuming use of AES-NI assembly instructions), and the speed of SHA256 to be 200 million bytes per second.[18]

Regarding the actual size of the circuits we are working with, we use the standard Bristol circuits[19] to obtain rough estimates for the number of gates in the AES, adder, and multiplier circuits. Given that we are using AES-256, we estimate a size of 50,666. We scale the numbers for the adder and multiplier to match our estimated input size—2λ. We estimate the size of the adder to be 1,500 and the size of the multiplier to be 100,000.

[17] We remark that [AHIV17] only provides concrete times for these two soundness parameters and that these times are for their non-interactive protocol.

[18] These are rough approximations for our setting. The actual times may differ depending on the length of the input, but we chose conservative estimates.

[19] https://homes.esat.kuleuven.be/~nsmart/MPC/