Round-Optimal Fully Black-Box Zero-Knowledge Arguments from One-Way Permutations

Carmit Hazay[1] and Muthuramakrishnan Venkitasubramaniam[2(✉)]

[1] Bar-Ilan University, Ramat Gan, Israel
carmit.hazay@biu.ac.il

[2] University of Rochester, Rochester, USA
muthuv@cs.rochester.edu

Abstract. In this paper, we revisit the round complexity of designing zero-knowledge (ZK) arguments via a black-box construction from minimal assumptions. Our main result implements a 4-round ZK argument for any language in NP, based on injective one-way functions, that makes black-box use of the underlying function. As a corollary, we also obtain the first 4-round perfect zero-knowledge argument for NP based on claw-free permutations via a black-box construction and 4-round input-delayed commit-and-prove zero-knowledge argument based on injective one-way functions.

Keywords: One-way permutations · Zero-knowledge arguments Black-box constructions

1 Introduction

Zero-knowledge (ZK) interactive proofs [GMR89] are paradoxical constructs that allow one player (called the prover) to convince another player (called the verifier) of the validity of a mathematical statement $x \in L$, while providing zero additional knowledge to the verifier. This is formalized by requiring that the view of every "efficient" adversary verifier $\mathcal{V}^*$ interacting with the honest prover $\mathcal{P}$ be simulated by an "efficient" machine $\mathcal{S}$ (a.k.a. the simulator). The idea behind this definition is that whatever $\mathcal{V}^*$ might have learned from interacting with $\mathcal{P}$, it could have actually learned by itself (by running the simulator $\mathcal{S}$). As "efficient" adversaries are typically modelled as probabilistic polynomial-time machines (PPT), the traditional definition of ZK models both the verifier and the simulator as PPT machines.

Several variants of ZK systems have been studied in literature. In this work, we are interested in computational ZK argument systems with black-box simulation, where the soundness is required to hold only against non-uniform PPT provers whereas the zero-knowledge property holds against PPT verifiers which get an auxiliary input. Such systems are referred to as computational zero-knowledge argument systems. We will further focus on the case of black-box constructions[1] and black-box simulation.[2] The main question we address is the the round-complexity of computational zero-knowledge argument systems based on minimal assumptions via a fully black-box construction. First, we survey prior work in this area.

Goldreich et al. [GMW91] constructed the first zero-knowledge proof systems for all of NP based on the minimal assumption of one-way functions, where they required polynomially many rounds to achieve negligible soundness. For arguments, Feige and Shamir [FS89] provided a 4-round zero-knowledge system based on algebraic assumptions. In [BJY97], Bellare, Jackobson and Yung showed how to achieve the same assuming only one-way functions.

On the negative side, Goldreich and Oren [GO94] demonstrated that three rounds are necessary for designing zero-knowledge for any non-trivial language (i.e. outside BPP) against non-uniform verifiers. When further restricting to black-box simulation, Goldreich and Krawcyzk [GK96b] showed that four rounds are necessary for achieving zero-knowledge of non-trivial languages. For the specific case of proofs (i.e. unconditional soundness), Katz [Kat12] showed that only languages in MA can have 4-round zero-knowledge proof systems.

As such, the works of [BJY97] and [GK96b] identify the round-complexity of zero-knowledge arguments as four when restricting to black-box simulation. However, when considering constructions that are black-box in the underlying primitives, Pass and Wee [PW09] provided the first black-box construction of a 6-round zero-knowledge argument for NP based on one-way permutations[3] and seven rounds based on one-way functions. Ishai, Mahmoody and Sahai provided the first black-box sublinear zero-knowledge arguments based on collision-resistant hash-functions [IMS12]. Ostrovsky, Richelson and Scafuro [ORS15] showed how to construct black-box two-party secure computation protocols in four rounds where only one party receives the output from enhanced trapdoor permutations. As zero-knowledge can be seen as an instance of such a secure computation, their work provides a round-optimal black-box construction based on enhanced trapdoor permutations.

This sequence of prior works leaves the following fundamental question regarding black-box constructions of zero-knowledge arguments open:

What is the weakest hardness assumption for a black-box construction of a 4-round zero-knowledge argument system for all of NP?

We remark that when considering non-black-box simulation, a recent work due to Bitansky et al. [BKP18] demonstrates how to obtain 3-round zero-knowledge arguments for NP based on multi-collision resistance hash functions. On the negative side, Fleischhacker et al. [FGJ18] proved that 3-round private-coin ZK proofs for NP do not exist, even with respect to non-black-box simulation assuming the existence of certain program obfuscation primitives.

[1] Where the construction is agnostic of the specific implementation and relies only on its input/output behavior.

[2] Where the simulator is only allowed to make black-box use of the verifier's code.

[3] Where injective one-way functions are sufficient.

Our Results. In this work we present the first 4-round ZK argument of knowledge protocols based on one-way permutations (injective one-way functions) and claw-free permutations. Specifically,

Theorem 1.1 (Informal). *Assuming injective one-way functions, there exists a fully black-box 4-round black-box computational zero-knowledge argument for all of* NP.

As a corollary we obtain the following result regarding perfect zero-knowledge argument systems.

Corollary 1.2 (Informal). *Assuming claw-free permutations, there exists a fully black-box 4-round black-box perfect zero-knowledge argument for all of* NP.

Commit-and-Prove Input-Delayed ZK Proofs. In [LS90], Lapidot and Shamir provided a three-round witness-indistinguishable (WI) proof for Graph Hamiltonicity with a special "input-delayed" property: namely, the prover uses the statement to be proved only in the last round. Recently, in [CPS+15] it was shown how to obtain efficient input-delayed variants of the related "Sigma protocols" when used in a restricted setting of an OR-composition. In [HV16], starting from a randomized encoding scheme with an additional robustness property and security against adaptive inputs, it was shown how to obtain general constructions of input-delayed zero-knowledge proofs that yield an efficient version of the protocol of [LS90] for arbitrary NP-relations.

The "commit-and-prove" paradigm considers a prover that first commits to a witness w and then, in a second phase upon receiving a statement x asserts whether a particular relation $R(x, w) = 1$ without revealing the committed value. This paradigm, which is implicit in the work of [GMW87] and later formalized in [CLOS02], is a powerful mechanism to strengthen semi-honest secure protocols to maliciously secure ones. The MPC-in-the-head approach of [IKOS09] shows how to obtain a commit-and-prove protocol based on one-way functions that relies on the underlying primitives in a black-box way. In [HV16] it was further shown how to extend the above input-delayed ZK proof to further support the commit-and-prove paradigm which is additionally black-box in the underlying one-way functions or permutations.

Instantiating the 3-round honest verifier zero-knowledge proof required in Theorem 1.1 with the commit-and-proof and input-delayed protocol from [HV16] implies the following corollary.

Corollary 1.3 (Informal). *Assuming injective one-way functions, there exists a fully black-box 4-round black-box commit-and-prove input-delayed zero-knowledge argument for all of* NP.

We prove the main theorem in Sect. 3 and the corollaries in Sect. 4.

1.1 Our Techniques

We begin with an overview of our 4-round ZK argument that is obtained by compiling 3-round (i.e. sigma) protocols of some special form. Consider a sigma protocol where the prover simply relies on commitments to generate its first round message and decommits to some subset of the commitments depending on the challenge provided by the verifier. Following [PW09], we require a special soundness guarantee in the protocol, where there exists at most one "easy challenge" that allows the prover to cheat for false instances. Furthermore, this easy challenge can be efficiently reconstructed from the set of messages committed to by the prover. An example of a sigma protocol with these properties, is the Blum Hamiltonicity zero-knowledge protocol [Blu]. Here, the prover commits to the adjacency matrix of a permutation of the underlying graph in the first round, and either decommits all entries in the matrix along with the permutation or decommits just the entries that form a Hamiltonian cycle depending on the verifier's challenge. Given the prover's commitments, the easy challenge can be extracted by observing whether the prover commits to the adjacency matrix of the permutation of original graph or just the entries of a Hamiltonian cycle.

This 3-round protocol already yields a zero-knowledge argument system, but only with constant soundness. To amplify soundness, one can have the entire protocol repeated in parallel, and have the verifier commit to all the parallel challenges in a first round of the protocol while decommitting in the third round. This 4-round protocol will indeed be zero-knowledge. However, one cannot prove that it is negligibly sound. Specifically, there could be a malleability attack, where, the prover upon receiving the verifier's commitment in the first round, can maul it to another commitment that can be open to a valid accepting response depending on the decommitment provided by the verifier in the third round. Another way of looking at this is that, one cannot have a black-box reduction of a cheating prover to the hiding property of the commitment used by the verifier in the first round to commit to the challenge. A standard way to circumvent this issue would be to require the verifier to use a perfectly hiding commitment and the prover a statistically binding commitment. However, this will result in a 5-round protocol (as perfectly hiding commitments require two rounds), and stronger assumptions, such as collision resistant hash functions.

The approach taken by Pass and Wee is to have the prover and verifier commit using a computationally hiding commitment scheme (that can be based on injective one-way functions) but additionally require the prover to prove "knowledge" of the messages in its commitment before the verifier decommits its challenge. This can be done generically using an extractable commitment scheme (introduced in the same work) which is a commitment scheme that has a "proof-of-knowledge" property. Before we go into the details of this construction, we point out that an extractable commitment scheme can be constructed from injective one-way function in three rounds which results in an overall zero-knowledge argument system with six rounds.

To collapse this protocol into four rounds we follow a *cut-and-choose* paradigm. Namely, our protocol will comprise of n parallel instances of the basic

4-round protocol. In the third round, the verifier chooses a random $S \subseteq [n]$ of some size t and decommits to the challenges made in those indices while providing a challenge for the extractable commitment for repetitions outside S. Then in the fourth round, the prover will complete the zero-knowledge protocol for the parallel repetitions with indexes in S and respond to the proof-of-knowledge challenge for the extractable commitment for the remaining indexes. The high-level idea here is that this allows to regain soundness in a simple way. Since the prover does not know the subset S revealed by the verifier in the third round, the prover has to "cheat" in most of the parallel invocations. This means we can argue by a simple averaging argument that there is an index $i \in [n]$ such that the probability that the prover cheats in the i^{th} repetition, i is not included in S and the prover convinces the verifier of a false statement is non-negligible. This means that we can now use the prover to violate the hiding of the commitment made by the verifier for the i^{th} repetition by running the proof-of-knowledge extractor on the prover's commitment in the i^{th} repetition and extracting the easy challenge.

However, proving zero-knowledge of this compilation is subtle and non-trivial. Recall that the verifier only reveals the challenges for a chosen subset S in the third round. A simple strategy for the simulator is to obtain the challenge, i.e. "trapdoor" for the indexes in S rewind and setup the prover messages in such a way that will allow for it to cheat in all repetitions in S. Now, the simulator can conclude with an accepting transcript if the verifier opens the same set S. However, the verifier can choose to reveal different subsets in different "rewindings". Nevertheless, in any rewinding, either the simulator has succeeded in cheating in all the indexes of the subset revealed by the verifier or has learned a new trapdoor. Now it suffices to show that the simulator will only require to perform a bounded number of rewindings before it has extracted most (if not all) trapdoors to complete the execution. A minor subtlety arises as a malicious verifier can abort before revealing the third message and this affects the number of rewindings that needs to be performed. However, this can be dealt with via a standard probability analysis. There is, however, a bigger issue in proving indistinguishability of this simulation. As described above, the simulator tries to extract trapdoors and outputs the "first" accepting transcript when it has managed to cheat in all indexes in the revealed subset. This simple idea however has a subtle flaw. The issue is that one can come up with a strategy for a malicious verifier where the distribution of the views output by the simulator is not indistinguishable from the real view. Roughly speaking, the distribution of the subset S in the transcript output by the simulator will be biased towards indexes revealed earlier in the rewindings. Our main technical contribution is to determine a "stopping" condition for the simulator that will result in the right distribution and we describe this below.

We abstract the simulation strategy to the following game. The game proceeds in iterations where in the i^{th} iteration the adversary outputs a subset $S_i \subset [n]$ from some unknown but pre-determined distribution D. The goal is to determine the iteration j to stop the game and output S_j such that the following two conditions are met:

- First, $S_j \subseteq S_1 \cup \cdots \cup S_{j-1}$, and
- Second, if D' is the distribution of the subset S_j output, then $D' = D$. In other words, the distribution of the subset output when the game is stopped is identical to the original distribution D.

Our main technical contribution is to show that the following simple strategy achieves the required goal.

- In any iteration if $S_j \subseteq S_1 \cup \cdots \cup S_{j-1}$, then halt if $S_j \not\subseteq S_1 \cup \cdots \cup S_{j-2}$, and proceed to the next iteration otherwise.

We prove this formally in Sect. 3.

2 Preliminaries

Basic Notations. We denote the security parameter by n. We say that a function $\mu : \mathbb{N} \rightarrow \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < \frac{1}{p(n)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. We further denote by $a \leftarrow A$ the random sampling of a from a distribution A, and by $[n]$ the set of elements $\{1, \ldots, n\}$. For an NP relation $\mathcal{R}$, we denote by $\mathcal{R}_x$ the set of witnesses of x and by $\mathcal{L}_{\mathcal{R}}$ its associated language. That is, $\mathcal{R}_x = \{\omega \mid (x, \omega) \in \mathcal{R}\}$ and $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists\ \omega\ s.t.\ (x, \omega) \in \mathcal{R}\}$. We specify next the definition of computationally indistinguishable.

Definition 2.1. *Let $X = \{X(a, n)\}_{a \in \{0,1\}^*, n \in \mathbb{N}}$ and $Y = \{Y(a, n)\}_{a \in \{0,1\}^*, n \in \mathbb{N}}$ be two distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every* PPT *machine $\mathcal{D}$, every $a \in \{0, 1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large n:*

$$\left|\Pr\left[\mathcal{D}(X(a, n), 1^n, a) = 1\right] - \Pr\left[\mathcal{D}(Y(a, n), 1^n, a) = 1\right]\right| < \frac{1}{p(n)}.$$

2.1 Commitment Schemes

Commitment schemes are used to enable a party, known as the *sender* Sen, to commit itself to a value while keeping it secret from the *receiver* Rec (this property is called *hiding*). Furthermore, in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing phase (this property is called *binding*). In this work, we consider commitment schemes that are *statistically binding*, namely while the hiding property only holds against computationally bounded (non-uniform) adversaries, the binding property is required to hold against unbounded adversaries. Formally,

Definition 2.2 (Commitment schemes). *A* PPT *machine* $\mathsf{Com} = \langle S, R\rangle$ *is said to be a non-interactive commitment scheme if the following two properties hold.*

Computational hiding: *For every (expected)* PPT *machine* Rec^*, *it holds that the following ensembles are computationally indistinguishable.*
- $\{\mathbf{View}^{\mathrm{Rec}^*}_{\mathsf{Com}}(m_1, z)\}_{\kappa\in N, m_1, m_2\in\{0,1\}^\kappa, z\in\{0,1\}^*}$
- $\{\mathbf{View}^{\mathrm{Rec}^*}_{\mathsf{Com}}(m_2, z)\}_{\kappa\in N, m_1, m_2\in\{0,1\}^\kappa, z\in\{0,1\}^*}$

where $\mathbf{View}^{R^*}_{\mathsf{Com}}(m, z)$ *denotes the random variable describing the output of* Rec^* *after receiving a commitment to* m *using* Com.

Statistical binding: *For any (computationally unbounded) malicious sender* Sen^* *and auxiliary input* z, *it holds that the probability that there exist valid decommitments to two different values for a view* v, *generated with an honest receiver while interacting with* $\mathrm{Sen}^*(z)$ *using* Com, *is negligible.*

We refer the reader to [Gol01] for more details. We recall that non-interactive perfectly binding commitment schemes can be constructed based on one-way permutation, whereas two-round statistically binding commitment schemes can be constructed based on one-way functions [Nao91]. To set up some notations, we let $\mathsf{com}_m \leftarrow \mathsf{Com}(m; r_m)$ denote a commitment to a message m, where the sender uses uniform random coins r_m. The decommitment phase consists of the sender sending the decommitment information $\mathsf{decom}_m = (m, r_m)$ which contains the message m together with the randomness r_m. This enables the receiver to verify whether decom_m is consistent with the transcript com_m. If so, it outputs m; otherwise it outputs $\perp$. For simplicity of exposition, in the sequel, we will assume that random coins are an implicit input to the commitment functions, unless specified explicitly.

2.2 Extractable Commitment Schemes

A core building block of our protocol is an extractable commitment scheme ExtCom introduced by Pass and Wee in [PW09].

Definition 2.3 (Extractable commitment schemes). *Let* $\mathsf{ExtCom} = (\mathrm{Sen}, \mathrm{Rec})$ *be a statistically binding commitment scheme. We say that* ExtCom *is an extractable commitment scheme if there exists an expected* PPT *oracle machine (the extractor)* E *that given oracle access to any* PPT *cheating sender* Sen^* *outputs a pair* (τ, m^*) *such that:*

Simulation: τ *is identically distributed to the view of* Sen^* *at the end of interacting with an honest receiver* Rec *in commit phase.*

Extraction: *The probability that* τ *is accepting and* $m^* = \perp$ *is negligible. We remark here that, we only need a weak extraction property where the extraction succeeds if the commitment is well formed. In other words, we allow for "over extraction" where the commitment could be invalid, yet, the extraction returns a value.*

Binding: *If $m^* \neq \perp$, then it is statistically impossible to open τ to any value other than m^*.*

In Fig. 1 we describe their 3-round extractable commitment scheme ExtCom that is based on one-way permutations. In order to commit to a bit m the sender splits m into two shares which are committed using a statistically binding commitment scheme Com. Next, the receiver sends a challenge bit e where the sender must open one of the two commitments that lie in the eth position. Later, in the decommit phase the sender opens the remaining commitments enabling the receiver to verify that all opening are valid and that all pairs correspond to the same bit m. Loosely speaking, hiding follows from hiding of the underlying commitment scheme Com. Whereas extractability follows from repetitively rewinding the sender obtaining two shares of a particular instance.

Extractable Commitment Scheme ExtCom [PW09]

The commitment scheme ExtCom uses a statistically binding commitment scheme Com and runs between sender Sen and receiver Rec.

Input: Sen holds a message $m \in \{0,1\}$.

Commit Phase:

Sen $\rightarrow$ Rec: Sen proceeds as follows:

1. Sen chooses $\eta_1, \ldots, \eta_\kappa \leftarrow \{0,1\}^\kappa$.
2. For all $i \in [\kappa]$, Sen commits to the following matrix:

$$\left(\mathsf{com}_{\eta_i} \ \mathsf{com}_{m \oplus \eta_i} \right) = \left(\mathsf{Com}(\eta_i) \ \mathsf{Com}(m \oplus \eta_i) \right).$$

Rec $\rightarrow$ Sen: Rec sends a challenge $e = e_1, \ldots, e_\kappa \leftarrow \{0,1\}^\kappa$ to Sen.

For all $i \in [\kappa]$, Sen sends the decommitment information $\mathsf{decom}_{(e_i \cdot m) \oplus \eta_i}$ for which the receiver checks the validity of openings.

Decommit Phase:

1. The sender sends m and opens the commitments to all κ pairs of strings.
2. The receiver checks that all the openings are valid, and also that all pairwise decommitments correspond to m.

Fig. 1. Extractable commitment scheme

2.3 Zero-Knowledge Arguments

We denote by $\langle A(\omega), B(z) \rangle(x)$ the random variable representing the (local) output of machine B when interacting with machine A on common input x, when the random-input to each machine is uniformly and independently chosen, and A (resp., B) has auxiliary input ω (resp., z).

Definition 2.4 (Interactive argument system). *A pair of* PPT *interactive machines* $(\mathcal{P}, \mathcal{V})$ *is called an* interactive proof system for a language $\mathcal{L}$ *if there exists a negligible function* negl *such that the following two conditions hold:*

1. Completeness: *For every* $x \in \mathcal{L}$ *there exists a string* ω *such that for every* $z \in \{0,1\}^*$,

$$\Pr[\langle \mathcal{P}(\omega), \mathcal{V}(z)\rangle(x) = 1] \geq 1 - \mathsf{negl}(|x|).$$

2. Soundness: *For every* $x \notin \mathcal{L}$, *every interactive* PPT *machine* $\mathcal{P}^*$, *and every* $\omega, z \in \{0,1\}^*$

$$\Pr[\langle \mathcal{P}^*(\omega), \mathcal{V}(z)\rangle(x) = 1] \leq \mathsf{negl}(|x|).$$

Definition 2.5 (Zero-knowledge). *Let* $(\mathcal{P}, \mathcal{V})$ *be an interactive proof system for some language* $\mathcal{L}$. *We say that* $(\mathcal{P}, \mathcal{V})$ *is* computational zero-knowledge with respect to an auxiliary input *if for every* PPT *interactive machine* $\mathcal{V}^*$ *there exists a* PPT *algorithm* $\mathcal{S}$, *running in time polynomial in the length of its first input, such that*

$$\{\langle \mathcal{P}(\omega), \mathcal{V}^*(z)\rangle(x)\}_{x \in \mathcal{L}, \omega \in \mathcal{R}_x, z \in \{0,1\}^*} \overset{\mathrm{c}}{\approx} \{\langle \mathcal{S}\rangle(x, z)\}_{x \in \mathcal{L}, z \in \{0,1\}^*}$$

(when the distinguishing gap is considered as a function of $|x|$*). Specifically, the left term denote the output of* $\mathcal{V}^*$ *after it interacts with* $\mathcal{P}$ *on common input* x *whereas, the right term denote the output of* $\mathcal{S}$ *on* x.

If further the distributions are identically distributed, we refer to the proof system as perfect zero-knowledge.

Definition 2.6 (Σ-protocol). *A protocol* π *is a* Σ-protocol *for relation* $\mathcal{R}$ *if it is a 3-round public-coin protocol and the following requirements hold:*

- Completeness: *If* $\mathcal{P}$ *and* $\mathcal{V}$ *follow the protocol on input* x *and private input* ω *to* $\mathcal{P}$ *where* $\omega \in \mathcal{R}_x$, *then* $\mathcal{V}$ *always accepts.*
- Special soundness: *There exists a polynomial-time algorithm* A *that given any* x *and any pair of accepting transcripts* $(a, e, t), (a, e', t')$ *on input* x, *where* $e \neq e'$, *outputs* ω *such that* $\omega \in \mathcal{R}_x$.
- Special honest-verifier zero knowledge: *There exists a* PPT *algorithm* $\mathcal{S}$ *such that*

$$\{\langle \mathcal{P}(\omega), \mathcal{V}(e)\rangle(x)\}_{x \in \mathcal{L}} \overset{\mathrm{c}}{\approx} \{\mathcal{S}(x, e)\}_{x \in \mathcal{L}}$$

where $\mathcal{S}(x, e)$ *denotes the output of* $\mathcal{S}$ *upon input* x *and* e, *and* $\langle \mathcal{P}(\omega), \mathcal{V}(e)(x)\rangle$ *denotes the output transcript of an execution between* $\mathcal{P}$ *and* $\mathcal{V}$, *where* $\mathcal{P}$ *has input* (x, ω), $\mathcal{V}$ *has input* x, *and* $\mathcal{V}$*'s random tape (determining its query) equals* e.

2.4 Claw-Free Permutations

Definition 2.7 (Claw-free permutations). *A triple of algorithms, (I, D, F), is called a claw-free collection if the following conditions hold.*

- *Both I and D are probabilistic polynomial-time, whereas F is deterministic polynomial-time. We denote by $f_i^\sigma(x)$ the output of F on input (σ, i, x), and by D_i^σ the support of the random variable $D(\sigma, i)$.*
- *For every i in the range of algorithm I, the random variables $f_i^0(D(0,i))$ and $f_i^1(D(1,i))$ are identically distributed.*
- *For every probabilistic polynomial-time algorithm, A', every polynomial $p(\cdot)$, and all sufficiently large n's*

$$\Pr[f_{I_n}^0(X_n) = f_{I_n}^1(Y_n)] \leq 1/p(n)$$

where I_n is a random variable describing the output distribution of algorithm I on input 1^n, and (X_n, Y_n) is a random variable describing the output of algorithm A' on input (random variable) I_n.

A construction for perfectly hiding commitment scheme based on claw-free permutations can be found in [GK96a].

3 The Feasibility of 4-Round BB ZK Arguments from OWPs

In this section we will prove our main theorem, demonstrating the feasibility of black-box 4-round zero-knowledge argument of knowledge. More formally, we prove the following theorem.

Theorem 3.1. *Assuming one-way permutations, Protocol 1 is a 4-round fully black-box zero-knowledge argument for any* NP *language.*

Building Blocks. Our protocol will employ the following cryptographic primitives.

Non-interactive perfectly binding commitment scheme: Such commitment schemes can be based on one-way permutations. We denote this scheme by Com and employ it for the verifier in the first message of our protocol.

Extractable commitment scheme: We recall that an extractable commitment scheme is a commitment scheme that has in addition an extraction algorithm, such that given an adversarial sender Sen*, can extract the committed message or output $\perp$ if the commitment is invalid. A 3-round extractable commitment scheme can be constructed based on any non-interactive commitment scheme [PW09]. We denote this scheme by ExtCom; see Sect. 2.2 for more details. We employ that commitment scheme for the prover.

3-round public-coin honest-verifier zero-knowledge proof: A 3-round zero-knowledge proof with constant soundness for any language in NP, denoted by $\pi_{\text{ZK}} = (a, e, t)$, that can be constructed starting from a non-interactive commitment scheme Com and where the witness is only used to in computing the third message t. For instance, the Blum's Hamiltonicity protocol [Blu] or [IKOS07,HV16]. For concreteness, let us consider the former protocol, where given a public input graph G, proceeds as follows. In the first message, the prover commits to the elements of the adjacency matrix $\{\mathsf{com}_{a_{ij}}\}_{i,j\in[m]}$ of a random permutation of the input graph G. The verifier responds with a challenge bit e. If $e = 0$, then the prover decommits all entries of the matrix and gives the permutation, and the verifier accepts if the permutation maps the input graph to the revealed graph. If $e = 1$, then the prover only decommits to elements in the adjacency matrix that form a Hamiltonian cycle. The verifier accepts if the revealed entries form an Hamiltonian cycle.

Protocol's Description: Our protocol executes the honest verifier zero-knowledge proof π_{ZK} in parallel n times, where a t subset of these executions (that is picked by the verifier) are completed till end while the rest are used for completing the extractable commitment algorithm.

Protocol 1 (Black-box 4-round zero-knowledge argument)

- **Inputs:** *A public statement $x \in \mathcal{L}$ for both and a witness $\omega \in \mathcal{R}_x$ for the prover $\mathcal{P}$.*
- **The protocol:**
  1. $\mathcal{V} \rightarrow \mathcal{P}$: *The verifier picks n challenges for the parallel invocations of protocol π_{ZK}, say $e_1, \ldots, e_n$, and commits to them using algorithm* Com. *Denote this set of commitments by* $(\mathsf{com}_{e_1}, \ldots, \mathsf{com}_{e_n})$.
  2. $\mathcal{P} \rightarrow \mathcal{V}$: *The prover generates n first-messages $(a_1, \ldots, a_n)$ according to π_{ZK}. Here each a_i contains commitments to entries of an adjacency matrix $\{\mathsf{extcom}_{a_i[r,c]}\}_{r,c\in[m]}$ of an independently and randomly chosen permutation of the input graph G where the commitment is computed using* ExtCom *where m is the number of nodes in the graph.*
  3. $\mathcal{V} \rightarrow \mathcal{P}$: *The verifier chooses a random t subset $T \subset \{1, \ldots, n\}$ and sends $\{\mathsf{decom}_{e_i}\}_{i\in T}$ where decom_{e_i} is the decommitment of com_{e_i}. It also sends a challenge $\mathsf{ch} \in ([n] - T)$ for all the extractable commitments.*
  4. $\mathcal{P} \rightarrow \mathcal{V}$: *Condition on valid decommitments sent by the verifier, for every ZK iteration $i \in T$, the prover completes protocol π_{ZK}, answering challenge e_i with the message t_i and sends $\{t_i, \mathsf{decom}_{a_i}\}_{i\in T}$. For the remaining ZK iterations, the prover simply responds to the challenge* ch *according to the extractable commitment protocol* ExtCom.

     The verifier accepts if all decommitments are valid, if (a_i, e_i, t_i) is a valid transcript for π_{ZK} for all $i \in T$ and if the extractbale commitments protocol has been concluded correctly for all remaining iterations $i \notin T$.

Proof **(Theorem 3.1).** Completeness follows directly from the completeness of the underlying honest verifier zero knowledge protocol π_{ZK}. Below we prove soundness and zero-knowledge of our protocol.

Soundness. On a high-level, the special soundness of the underlying zero-knowledge protocol implies that, on a false statement, and a set of commitments provided by the prover in its first message, there is only one "easy challenge" for which the prover can complete the protocol and convince the verifier. Pass and Wee in [PW09] formalized the notion of "easy challenge" by requiring that the zero-knowledge protocol satisfies the property that there is an efficient procedure that given the input statement x and values in the commitments made by the prover in the second message $v_1, \ldots, v_k$, outputs a string e such that if an easy challenge exists then it must equal e, and if this challenge is revealed by the verifier the (malicious) prover can convince the verifier even on a false statement. For example, the Blum Hamiltonicity zero-knowledge protocol satisfies this requirement and the easy challenge can be extracted as follows. If the value committed to by the prover is a permutation π and the adjacency matrix A such that A represents the graph $\pi(G)$, then set the easy challenge to be 0 and otherwise 1. We argue soundness based on the following two steps.

1. We show that for a false statement an adversarial prover has to guess the challenge from the commitments made by the verifier before it is revealed in the third message for most of the n parallel instances. More precisely, the "easy challenge" extracted from the messages committed by the prover in most of the n iterations must match exactly the challenge committed to by the verifier.
2. There is an extraction procedure to extract the messages committed by the prover in one of these iterations without having to reveal the challenge committed to in the first message.

Combining these two ideas, we can reduce the soundness of the zero-knowledge to the hiding property of the commitments made by the verifier. We remark that our protocol and proof are different from those presented in [PW09] in that the verifier only reveals a subset of the challenges, where essentially the prover is only required to convince the verifier in the executions corresponding to this subset. In contrast, in the protocol presented in [PW09] the verifier opens all challenges. Specifically, as their protocol includes additional rounds between the prover's second message and when the verifier reveals the challenge in order to extract the prover's committed message, their analysis becomes easier. In our protocol, on the other hand, we will be able to extract the values in the commitments made by the prover only in the repetitions for which the challenge was not revealed by the verifier. We now proceed to the formal proof.

Assume for contradiction that there exists a PPT prover $\mathcal{P}^*$ and polynomial $p(\cdot)$ such that for infinitely many n's, there exists $x_n \notin \mathcal{L} \cap \{0,1\}^n$ such that the prover successfully convinces the verifier on the statement x_n with probability $\frac{1}{p(n)}$. Fix an arbitrary n for which this happens. We will construct an adversary $\mathcal{B}$ that uses $\mathcal{P}^*$ to break the hiding property of the non-interactive commitment scheme Com. More formally, $\mathcal{B}$ will internally incorporate the code of the prover $\mathcal{P}^*$ on input $(1^n, x_n)$ and feed it with messages according to the honest verifier. That is, on input $(1^n, x_n)$ and a commitment c from the external challenger, $\mathcal{B}$ proceeds as follows.

1. It will begin an internal emulation with $\mathcal{P}^*$. To simulate the first message from the verifier, it will choose a random index i to feed the challenge commitment c and the rest of them it will generate honestly internally.
2. Next, it will continue the execution to completion where it picks a random subset $S_1 \subseteq [n]$ conditioned on $i \notin S_1$. Let ch_1 be the challenge it feeds for the extractable commitment. If the prover aborts in the internal emulation then $\mathcal{B}$ aborts.
3. Otherwise, it will record the response to challenge e_i for the extractable commitment in repetition i. Next, it will rewind the prover to the third message, giving another set $S_2 \subseteq [n]$ subject to $i \notin S_2$ and an independent challenge ch_2 for the extractable commitment. If the prover aborts, $\mathcal{B}$ aborts as well. Otherwise, it will use the extractor for the underlying extractable commitment scheme on the commitment made for iteration i and the responses given for two challenges. We remark here that our extractor could "over extract". Namely, extract in case of an invalid commitment. To deal with this, we stipulate that if the extractor extracts a valid graph, the bit b is set to 1 and otherwise 0. If the extractor successfully extracts the committed messages, then $\mathcal{B}$ extracts the easy challenge b, outputs b and halts.

We next prove in the claim that $\mathcal{B}$ breaks the hiding property of the challenge commitment c with non-negligible probability.

Claim 3.1. *There exists polynomial $q(\cdot)$ such that,*

$$\Pr[b \leftarrow \{0,1\}^n : c \leftarrow \mathsf{Com}(1^n, b) : \mathcal{B}(1^n, x, c) = b] \geq \frac{1}{q(n)}.$$

Proof: Define the random variable Γ to be the set that contains the indexes where the prover commits to the adjacency matrix according to the easy challenge. We will further restrict Γ to be those indices where if $b = 1$ (meaning the prover commits to the graph), the index will be included only if the commitment is valid. This means that the prover can successfully convince the verifier only if $T \subseteq \Gamma$. Note that this set (even if not efficiently computable) is well-defined as we rely on statistically binding commitments. Our analysis relies on the following two cases:

Case $|\Gamma| \leq \frac{3n}{4}$: Here the probability that $T \subseteq \Gamma$ can be bounded by $\frac{\binom{3n/4}{t}}{\binom{n}{t}}$ which is negligible. We remark here that if $b = 1$ and the commitment is invalid, then the Prover can not convince the verifier in that index because all the commitments are decommitted in the fourth message. Based on the observation that T must be contained in γ, the prover successfully completes the protocol only with negligible probability.

Case $|\Gamma| > \frac{3n}{4}$: We begin by showing that there exists an index $i \in \Gamma$ such that $\mathcal{P}^*$ convinces $\mathcal{V}$ with non-negligible probability conditioned on $i \notin T$. Define p_i to be the probability that $\mathcal{P}^*$ successfully convinces the verifier conditioned on $i \notin T$ where recall that T is the set of challenges revealed by the verifier. By a union bound, we have that $\sum_{i \in \Gamma} p_i \geq \frac{1}{p(n)}$. Therefore, there exists i

such that $p_i \geq \frac{1}{|\Gamma|p(n)} \geq \frac{4}{3np(n)}$.
Now, we have that if $\mathcal{B}$ picks this index i and the prover completes the proof in both the executions performed by $\mathcal{B}$, then with overwhelming probability the extractor reveals the messages committed to by the prover. This in turn reveals the easy challenge for all indexes outside $S_1 \cup S_2$. In particular, it will obtain the easy challenge b_i which $\mathcal{B}$ outputs as its guess for the challenge commitment c. By definition of Γ we have that b_i is correct.
$\mathcal{B}$ succeeds if it picks this index i to feed the external challenge, convinces $\mathcal{V}^*$ in the two executions, the extractor succeeds. The right index is chosen with probability $\frac{1}{2^n}$. for the specific extractable commitment used in the construction (namely, the construction from [PW09]), the extractor succeeds except with negligible probability if the $\mathsf{ch}_1 \neq \mathsf{ch}_2$ which happens with probability at most $1 - 2^{-n}$. Furthermore, even if the extractor "over-extracts", if the extracted value is the valid graph, it cannot be the case that the prover can convince with $b = 0$ and we know that if $i \in \Gamma$ and $b = 1$ then the commitment is valid. Therefore the probability that $\mathcal{B}$ succeeds is at least $\frac{1}{n}p_i^2 - \frac{1}{2^n} - \nu(n) \geq \frac{1}{2n^3(p(n))^2}$. □

This concludes the proof of soundness.

Zero-Knowledge. We describe our black-box simulator and prove correctness of simulation.

Description of Simulator $\mathcal{S}$: More formally, let $\mathcal{V}^*$ be a malicious verifier. We define simulator $\mathcal{S}$ as follows:

1. $\mathcal{S}$ receives the first message $\mathcal{V}^*(x, z)$ from the malicious verifier.
2. $\mathcal{S}$ continues the execution by generating the second message according to the honest prover's algorithm. If the verifier aborts, the simulator outputs the transcript and halts.
3. Otherwise, $\mathcal{S}$ records the challenges that the verifier reveals; denote this t subset by T_1. Set $T_0 = \emptyset$.
4. Next, $\mathcal{S}$ repeatedly rewinds the verifier to the second message to extract some trapdoor information, namely, decommitments of the challenges committed by the verifier. It proceeds in iterations. In iteration ℓ, we assume that the $\mathcal{S}$ holds the sets $T_1, \ldots, T_\ell$ and at the end of the iteration either the simulator learns a new trapdoor (and adds a new set $T_{\ell+1}$) or halts outputting a transcript. More precisely, for $\ell = 1$ through $n - t + 1$,[4] the simulator proceeds as follows:
   (a) It generates the second prover's message $(a_1, \ldots, a_n)$ as follows:
      – For $i \notin T_1 \cup \cdots \cup T_\ell$, run the honest prover strategy to generate the second message a_i. In the particular Blum's Hamiltonian proof that we use, this amounts to simply generating commitments to the adjacency matrix of a random permutation of the original graph G.

[4] Note that this is the maximum number of iterations as at least one new element is added in each iteration and $|T_1| = t$.

- For $i \in T_1 \cup \cdots \cup T_\ell$, let e_i be the challenge revealed for index i. The simulator runs $\mathcal{S}_{\text{ZK}}(x, e_i)$ of the underlying honest-verifier zero-knowledge proof in order to generate the second and fourth messages (p_2^i, p_4^i) using the knowledge of the challenge e_i. It then sets $a_i = p_2^i$.

Let T' be the challenge set revealed by the verifier. The simulator repeats until one of the following cases occur:

Case 1. $T' \not\subseteq T_1 \cup \cdots \cup T_\ell$: This case implies that the verifier reveals a challenge of a new ZK repetition that the simulator did not record before. In this case, the simulator sets $T_{\ell+1} = T'$ and proceeds to the next iteration under ℓ (i.e. go to Step 4).

Case 2. $T' \subseteq T_1 \cup \cdots \cup T_\ell$: This case implies two subcases.

Case 2.1. $T' \subseteq T_1 \cup \cdots \cup T_{\ell-1}$: The simulator ignores this case and continues to rewind, i.e. go to step 4(a). We remark here that in this case, the simulator could complete the execution as it has simulated all the second messages according to the challenges corresponding to the set T'. Nevertheless, we deliberately make the simulator ignore this case so as to not skew the probability distributions of the simulator's output.

Case 2.2. $T' \not\subseteq T_1 \cup \cdots \cup T_{\ell-1}$**and** $T' \subseteq T_1 \cup \cdots \cup T_\ell$: This case considers the event where the revealed subset T' is not contained in the first $\ell - 1$ collected sets, but is contained in first ℓ sets. In this case, the simulator continues the simulation and generates the fourth message $(r_1, \ldots, r_n)$ for every $i \in [n]$ as follows:

- If $i \notin T'$, the simulator needs to respond to the challenge given for the extractable commitment scheme. In this case, the simulator simply responds to the challenge honestly.
- If $i \in T'$, then recall that the second message a_i was set to p_2^i, where (p_2^i, p_4^i) were generated using the honest verifier zero-knowledge simulator based on the challenge e_i (which is implied by the fact that $T' \subseteq T_1 \cup \cdots \cup T_\ell$). Therefore, if the revealed challenge for this repetition i is e_i, then the simulator sets the fourth message $r_i = p_4^i$. On the other hand, if the verifier reveals a different challenge for repetition i, then the simulator aborts. Note that the simulator will never abort because the challenges are committed using a perfectly binding commitment scheme Com.

The simulator then feeds this last message and outputs the view of the verifier.

Proof of Indistinguishability. Denote by $\mathbf{View}_{\mathcal{V}^*}(\mathcal{P}(x, \omega), \mathcal{V}^*(x, z))$ the view of the verifier $\mathcal{V}^*(z)$ when interacting with the honest prover on input ω and common input x. We prove the indistinguishability of real and simulated proofs by defining the following intermediate hybrid experiments.

Hybrid Hyb_0: In this experiment, we consider the view of the verifier when it interacts with the honest prover with witness ω.

Hybrid Hyb_1: In this experiment, we define a simulator $\mathcal{S}_1$ that proceeds with the rewinding strategy as simulator $\mathcal{S}$ does, with the exception that the prover's messages are generated according to the honest prover's strategy. Define $\mathcal{S}_1(x, \omega, z)$ to be the output of the simulator $\mathcal{S}_1$ in this hybrid. We next prove indistinguishability and analyze the running time of $\mathcal{S}_1$ in the following claims.

Claim 3.2. *The following distributions are identical.*

- $\mathcal{D}_0 = \{\mathbf{View}_{\mathcal{V}^*}(\mathcal{P}(x, \omega), \mathcal{V}^*(x, z))\}_{x \in \mathcal{L}, \omega \in \mathcal{R}_x, z \in \{0,1\}^*}$
- $\mathcal{D}_1 = \{\mathcal{S}_1(x, \omega, z)\}_{x \in \mathcal{L}, \omega \in \mathcal{R}_x, z \in \{0,1\}^*}$

Proof: Fix a random tape r for $\mathcal{V}^*$. Let $\psi = (V_1, P_1^\psi, V_2, P_2^\psi)$ be the transcript of a random execution between $\mathcal{V}^*(x, z; r)$ and an honest prover $\mathcal{P}(x, \omega)$. We will show that the probability with which this transcript is returned is identical in both distributions. Let p_ψ be the probability with which this transcript appears in $\mathcal{D}_0$ conditioned on $\mathcal{V}^*$'s random tape being fixed to r. Clearly, the resulting first message will always be V_1, if $\mathcal{S}_1$ emulates the interaction with $\mathcal{V}^*$ on a random tape r. Then we prove that transcript $(V_1, P_1^\psi, V_2, P_2^\psi)$ is generated by $\mathcal{S}_1(x, \omega, z)$ with the same probability p_ψ conditioned on the random tape of $\mathcal{V}^*$ being r.

Note first, that by the definition of $\mathcal{S}_1$, the probability with which an aborting transcript appears in both distributions is identical. We therefore focus on non-aborting transcripts. Therefore, it suffices to compute the probability that $\mathcal{S}_1(x, \omega, z)$ outputs the (non-aborting) transcript of messages $(V_1, P_1^\psi, V_2, P_2^\psi)$ conditioned on $\mathcal{V}^*$'s random tape being fixed as r. We continue with some more conventions and notations:

- We denote by S the set that occurs in the target transcript ψ, namely, the set contained in message V_2.
- We denote by p_T the probability the subset T occurrs in the real execution. We let $p_\perp$ denote the probability that the verifer aborts before sending its second message in the real execution. In this notation $p_T = \sum p_\psi$ where the summation is over all transcripts ψ that contains the subset T.
- We denote a tuple of sets by $\mathbf{T} = (T_1, \ldots, T_\ell)$ to denote the sets collected by the simulator before it enters the ℓ^{th} iteration. Typically, given a tuple $\mathbf{T}$, we use $\widetilde{\mathbf{T}}$ to denote the tuple $(T_1, \ldots, T_{\ell-1})$ and use :: for appending a set. In this notation, $\mathbf{T} = \widetilde{\mathbf{T}} :: T_\ell$.
- For $1 \leq \ell \leq n - t + 1$, let Valid_ℓ denote the set of all ℓ-tuples $(T_1, \ldots, T_\ell)$ that satisfy the following two conditions.
  1. All sets T_i are of size t.
  2. For every $1 \leq i \leq \ell$, it holds that $T_i \not\subseteq T_1, \ldots, T_{i-1}$. (Recall that the simulator moves to the next iteration only if it finds a new trapdoor).

  Intuitively, valid sequences captures all sequences that can be obtained by the simulator when entering the ℓ^{th} iteration.[5]

[5] By possibly we mean that it might be the case that the verifier never opens some particular t subset T in any execution, in which case any tuple that involves T will never occur in a simulation.

- For any ℓ-tuple $\mathbf{T} = (T_1, \ldots, T_\ell)$, we define $q_{\mathbf{T}}$ the probability *conditioned on not aborting* that in a random execution between $\mathcal{V}^*(x, z; r)$ and the honest prover, the set opened by the verifier is covered by the elements $T_1, \ldots, T_\ell$, i.e. $T \subseteq \cup_{i=1}^{\ell} T_i$. We set $q_{\{\emptyset\}} = 0$. We next observe that, for any tuple $\mathbf{T} = (T_1, \ldots, T_\ell)$, it holds that $q_{\mathbf{T}} = (\sum p_T)/(1-p_\perp)$ where the summation is over all T such that $|T| = t$ and $T \subseteq \cup_{i=1}^{\ell} T_i$.
- For a tuple $\mathbf{T} = (T_1, \ldots, T_\ell)$, let $P_{\mathbf{T}}^{\psi}(\ell)$ denote the probability that, starting with sets $\mathbf{T}$ and iteration ℓ, the simulator $\mathcal{S}_1$ outputs the transcript ψ.

Without loss of generality we assume $p_\perp < 1$, since, if the verifier aborts w.p. 1, the simulator outputs the transcript from the first execution and will be distributed identically to the real execution. We begin with the following claim which will be sufficient to prove Claim 3.2.

Subclaim 3.3. *For $1 \leq \ell \leq n-t+1$ and every tuple $\mathbf{T} = (T_1, \ldots, T_\ell) \in \mathsf{Valid}_\ell$,*

$$P_{\mathbf{T}}^{\psi}(\ell) = \begin{cases} \frac{p_\psi}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})} & \textit{if } \widetilde{\mathbf{T}} \textit{ does not cover } S, \textit{ and} \\ 0 & \textit{otherwise.} \end{cases}$$

where $\widetilde{\mathbf{T}} = (T_1, \ldots, T_{\ell-1})$.

Before we prove this claim, we conclude Claim 3.2 using the preceding subclaim. As argued above, the probability that the simulator outputs aborting transcripts is identical to the real execution. Observing that $q_\emptyset = 0$, conditioned on not aborting, the probability that the simulator outputs non-aborting ψ is given by $P_{\mathbf{T}}^{\psi}(0)$ which from the preceeding claim is $p_\psi/(1-p_\perp)$. Since the probability that the simulator continues after the first execution is $(1-p_\perp)$, Claim 3.2 follows.

Now we proceed to prove Subclaim 3.3.

Proof: Given $\mathbf{T} = (T_1, \ldots, T_\ell)$, suppose $\widetilde{\mathbf{T}} = (T_1, \ldots, T_{\ell-1})$ covers S, then from the description of our simulation it follows that it is not allowed to output ψ in iterations ℓ or higher. In other words, when $\widetilde{\mathbf{T}}$ covers S, $P_{\mathbf{T}}^{\psi}(\ell) = 0$ as in the claim.

Therefore, it suffices to prove the subclaim when $\widetilde{\mathbf{T}}$ does not cover S. We prove this case using a reverse induction on ℓ from $n-t+1$ to 1.

Base Case: $\ell = n-t+1$. Let $\mathbf{T} = (T_1, \ldots, T_{n-t+1})$ be an arbitrary valid tuple and let $\widetilde{\mathbf{T}} = (T_1, \ldots, T_{n-t})$. Recall that, for a general iteration ℓ, the simulator rewinds until it obtains $T \not\subseteq \cup_{i=1}^{\ell-1} T_i$. Then, if $T \subseteq \cup_{i=1}^{\ell} T_i$ it outputs the transcript. Otherwise, it has obtained a new trapdoor, sets T to be the new set T_{i+1} and proceeds to the next iteration. However, if $\ell = n-t+1$, we have that $\cup_{i=1}^{n-t+1} T_i$ must be $[n]$ as at least one new element is added in each iteration and $|T_1| = t$. Therefore, in this base case, we have that $S \not\subseteq \cup_{i=1}^{n-t} T_i$ and $S \subseteq \cup_{i=1}^{n-t+1} T_i$. This means that if the simulator encounters the transcript

ψ in iteration $n-t+1$, it will output it. The probability can be computed as follows:

$$\begin{aligned}&\Pr[\psi \text{ occurs in the iteration} \mid \text{no } t \text{ -subset of } \cup_{i=1}^{\ell-1} T_i \text{ occurs}]\\&\quad= \frac{\Pr[\psi \text{ occurs in the iteration }]}{\Pr[\text{ no } t\text{-subset of } \cup_{i=1}^{\ell-1} T_i \text{ occurs}]}\\&\quad= \frac{p_\psi}{(1-p_\perp)} \times \frac{1}{(1-q_{\widetilde{\mathbf{T}}})}\end{aligned}$$

This completes our base case.

Induction Step: $1 \le \ell \le n-t$. Let $\mathbf{T} = (T_1, \ldots, T_\ell)$ be an arbitrary tuple in Valid_i. Set $\widetilde{\mathbf{T}} = (T_1, \ldots, T_{\ell-1})$. Recall that we only need to show the subclaim when $\widetilde{\mathbf{T}}$ does not cover S. There are two cases w.r.t $\mathbf{T}$:

Case 1: T covers S: In this case, the simulator can output ψ only in this iteration and not higher. Recall that the simulator in this iteration will rewind until it obtains a set $T \not\subseteq \widetilde{\mathbf{T}}$. Therefore, the probability that the simulator outputs ψ is same as in the base case and given by $p_\psi/((1-p_\perp)(1-q_{\widetilde{\mathbf{T}}}))$.
Case 2: T does not cover S: This means that the simulator can output ψ only in iterations $\ell+1$ or higher. Then for any subset T not covered by $\mathbf{T}$ the probability that the simulator outputs ψ in iteration $\ell+1$ or higher is given by

$$\begin{aligned}&\Pr[T \text{ occurs in the current iteration} \mid \text{no } t\text{-subset of } \cup_{i=1}^{\ell-1} T_i \text{ occurs}]\\&\quad\times \Pr[\psi \text{ occurs in iteration } \ge \ell+1 \text{ with } \mathbf{T} :: T \text{ occuring in the first } \ell \text{ iterations}]\\&\quad= \Pr[T \text{ occurs in the current iteration} \mid \text{no } t\text{-subset of } \cup_{i=1}^{\ell-1} T_i \text{ occurs}]\\&\quad\times P^{\psi}_{\mathbf{T}::T}(\ell+1)\\&\quad= \frac{p_\psi}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})} \times P^{\psi}_{\mathbf{T}::T}(\ell+1)\end{aligned}$$

This means that the overall probability can be obtained by summing the proceeding expression over all sets T not covered by $\mathbf{T}$, namely, $T \not\subseteq T_1, \ldots, T_\ell$.

$$\begin{aligned}P^{\psi}_{\mathbf{T}}(\ell) &= \sum_{T \not\subseteq T_1 \cup \cdots \cup T_{\ell-1}} \frac{p_T}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})} \times P^{\psi}_{\mathbf{T}::T}(\ell+1)\\&= \sum_{T \not\subseteq T_1 \cup \cdots \cup T_{\ell-1}} \frac{p_T}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})} \times \frac{p_\psi}{(1-p_\perp)(1-q_{\mathbf{T}})}\\&= \frac{p_\psi}{(1-p_\perp)(1-q_{\mathbf{T}})} \times \frac{1-q_{\mathbf{T}}}{1-q_{\widetilde{\mathbf{T}}}}\\&= \frac{p_\psi}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})}.\end{aligned}$$

where in the second step we invoke our induction hypothesis that $P^{\psi}_{\mathbf{T}::T}(\ell+1) = p_\psi/((1-p_\perp)(1-q_{\mathbf{T}}))$.

This completes our inductive step and concludes the proof of our subclaim. $\square$

Claim 3.4. *The expected running time of $\mathcal{S}_1$ is polynomial.*

Proof: We argue by induction on the iterations that the expected running time of the simulator $\mathcal{S}_1$ defined in this hybrid is polynomial. Define $\mathsf{RunTime}_{\mathbf{T}}(\ell)$ to be the expected total running time of the simulator in iterations ℓ and above conditioned on $\mathbf{T} = (T_1, \ldots, T_\ell)$ being the sets obtained by the simulator in the first $\ell - 1$ iterations.

Subclaim 3.5. *There exists a constant c such that, for any valid tuple $(T_1, \ldots, T_\ell)$, $\mathsf{RunTime}_{\mathbf{T}}(\ell) \leq \frac{n^c(n-\ell)}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})}$ where $1 \leq \ell \leq n - t + 1$ and $\widetilde{\mathbf{T}} = (T_1, \ldots, T_{\ell-1})$.*

Proof: As in the previous proof we do reverse induction on iteration ℓ.

Base Case $\ell = n - t + 1$. Let $\mathbf{T} = (T_1, \ldots, T_{n-t+1})$. Recall that in iteration $\ell = n - t + 1$ we have $\cup_{i=1}^{n-t+1} T_i = [n]$. Therefore, there are no more iterations and the simulator stops whenever it finds any T such that $T \not\subseteq \cup_{i=1}^{n-t} T_i$. The probability of observing such an execution using our notation defined above is given by $(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})$. Therefore, the expected number of rewindings that the simulator needs to perform in the $(n-t+1)^{st}$ iteration is $1/((1-p_\perp)(1-q_{\widetilde{\mathbf{T}}}))$. This in turn means the expected time spent by the simulator conditioned on entering iteration $n - t + 1$ with sets $(T_1, \ldots, T_{n-t+1})$, i.e.

$$\mathsf{RunTime}_{\mathbf{T}}(n - t + 1) = \frac{n^c}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})}$$

where n^c is an upper bound on the time spent by the simulator in a single rewinding with the verifier.

Induction Step: $1 \leq \ell \leq n-t$**.** We will compute the expected time spent in this iteration. Suppose that the simulator collected the sets $(T_1, \ldots, T_\ell)$ in the first $\ell - 1$ iterations. Recall that the simulator rewinds until it obtains $T \not\subseteq \cup_{i=1}^{\ell-1} T_i$ and either outputs the transcript (if $T \subseteq \cup_{i=1}^{\ell} T_i$) or moves on to the next iteration otherwise. The number of rewindings in this iteration is therefore $\frac{1}{1-q_{\widetilde{\mathbf{T}}}}$ in expectation. Now, the total expected running time in iterations ℓ and above can be computed as

$$\begin{aligned}
&E[\#\text{rewindings in iteration } \ell \text{ until it obtains } T \not\subseteq \cup_{i=1}^{\ell-1} T_i] \times n^c \\
&\quad + E[\text{time spent in iterations } > \ell \text{ with } T] \\
&\quad = \frac{n^c}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})} + \sum_{T' \not\subseteq \cup_{i=1}^{\ell} T_i} \Pr[T = T' | T \not\subseteq \cup_{i=1}^{\ell-1} T_i] \times \mathsf{RunTime}_{\mathbf{T}::T}(\ell + 1) \\
&\quad \leq \frac{n^c}{(1-p_\perp)(1-q_{\widetilde{\mathbf{T}}})} + \frac{n^c(n-\ell-1)}{(1-p_\perp)(1-q_{\mathbf{T}})} \times \sum_{T' \not\subseteq \cup_{i=1}^{\ell} T_i} \Pr[T = T' | T \not\subseteq \cup_{i=1}^{\ell-1} T_i]
\end{aligned}$$

$$= \frac{n^c}{(1-p_\perp)(1-q_{\tilde{\mathbf{T}}})} + \frac{n^c(n-\ell-1)}{(1-p_\perp)(1-q_{\mathbf{T}})} \times \frac{\sum_{T' \not\subseteq \cup_{i=1}^{\ell} T_i} \Pr[T = T']}{1-q_{\tilde{\mathbf{T}}}}$$
$$= \frac{n^c}{(1-p_\perp)(1-q_{\tilde{\mathbf{T}}})} + \frac{n^c(n-\ell-1)}{(1-p_\perp)(1-q_{\mathbf{T}})} \times \frac{1-q_{\mathbf{T}}}{1-q_{\tilde{\mathbf{T}}}}$$
$$= \frac{n^c(n-\ell)}{(1-p_\perp)(1-q_{\tilde{\mathbf{T}}})}$$

where the third step follows from the induction hypothesis. □

The expected total running time of the simulation is given by

$$p_\perp \times n^c + (1-p_\perp) \times \mathsf{RunTime}_\emptyset(1) = n^c + n^c(n-1)$$

and this concludes the proof of the claim. □

Hybrid Hyb_2: In this experiment we consider the actual simulation as defined by $\mathcal{S}(x,z)$. The output of the experiment will then be $\mathcal{S}(x,z)$.

Claim 3.6. *The following distributions are identical.*

– $\{\mathcal{S}_1(x,\omega,z)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_x,z\in\{0,1\}^*}$
– $\{\mathcal{S}(x,z)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_x,z\in\{0,1\}^*}$

Proof. Assume for contradiction that there exists a malicious verifier $\mathcal{V}^*$, a distinguisher $\mathcal{D}$ and a polynomial $p(n)$ such that for infinitely many n's, $\mathcal{D}$ distinguishes $\mathcal{S}_1(x,\omega,z) = \langle \mathcal{S}_1(\omega), \mathcal{V}^*(z)\rangle(x)$ and $\mathcal{S}(x,z) = \mathcal{S}^{\mathcal{V}^*}(x,z)$ with probability $\frac{1}{p(n)}$. Fix any n for which this event occurs.

First, we consider truncated experiments $\overline{\mathrm{Hyb}}_1(n,x,z)$ (resp. $\overline{\mathrm{Hyb}}_2(n,x,z)$) which proceeds exactly as $\mathrm{Hyb}_1(n,x,z)$ (resp. $\mathrm{Hyb}_2(n,x,z)$) with the exception that the simulation is aborted if it runs more than $np(n)t(n)$ steps where $t(n)$ is the polynomial bounding the expected running time of $\mathcal{S}_1$. If the experiment is aborted then $\overline{\mathrm{Hyb}}_1$ (resp. $\overline{\mathrm{Hyb}}_2$) is set to a special symbol $\perp$. By an averaging argument we can conclude that the truncated experiments $\overline{\mathrm{Hyb}}_1(n,x,z)$ and $\overline{\mathrm{Hyb}}_2(n,x,z)$ can be distinguished with probability at least $\frac{1}{2p(n)}$ by the distinguisher $\mathcal{D}$.

Next, we consider a sequence of intermediate hybrids $\mathrm{Hyb}_1^0,\ldots,\mathrm{Hyb}_1^{n-t+1}$, where in Hybrid Hyb_1^ℓ, we define a simulator $\mathcal{S}_1^\ell$ that will follow the real simulator's strategy $\mathcal{S}$ in the first ℓ iterations of the for loop and the remaining according to the honest prover using the real witness. If $\mathcal{S}_1^\ell$ runs over $np(n)t(n)$ steps then we stop the simulation and output $\perp$. Let $\overline{\mathrm{Hyb}}_1^\ell(n,x,z)$ be the output of the $\mathcal{S}_1^\ell$ in Hyb_1^ℓ. It follows from definition that $\overline{\mathrm{Hyb}}_1^0 = \overline{\mathrm{Hyb}}_1$ and $\overline{\mathrm{Hyb}}_1^{n-t+1} = \overline{\mathrm{Hyb}}_2$. Therefore, if $\mathcal{D}$ distinguishes $\overline{\mathrm{Hyb}}_1^0$ from $\overline{\mathrm{Hyb}}_1^{n-t+1}$ then there exists an index i such that $\mathcal{D}$ distinguishes $\overline{\mathrm{Hyb}}_1^i$ from $\overline{\mathrm{Hyb}}_1^{i+1}$ with probability $\frac{1}{2np(n)}$. Since the experiments are truncated after $np(n)t(n)$ steps the maximum number of rewindings that can occur in iteration i where the two experiments differ is

$np(n)t(n)$. We show that using $\mathcal{V}^*$ and $\mathcal{D}$ we can contradict the honest verifier zero-knowledge property (for many parallel repetitions).

Consider an adversary $\mathcal{A}$ that begins an emulation of $\overline{\text{Hyb}}_1^i(n, x, z)$ until it reaches iteration i. If it halts before, $\mathcal{A}$ simply outputs the output of the experiment. Otherwise, let $T_1, \ldots, T_i$ be the set of indexes that were obtained by the simulator in the internal emulation. Let $T = T_1 \cup \cdots \cup T_i$ and let $\{e_t\}_{t \in T}$ be the challenges in the indexes in T. $\mathcal{A}$ forwards these challenges to an external challenger $\mathcal{C}$. The challenger then produces $np(n)t(n)$ transcripts of the honest-verifier zero-knowledge protocol for each challenge e_t for $t \in T$. $\mathcal{A}$ uses the prover's messages in these transcripts to generate the prover messages in the internal emulation in iteration i. Then it completes the experiment, where from iteration $i+1$ the adversary plays the honest prover strategy and uses the real witness, and outputs the output of the experiment. By our construction, if the external challenger $\mathcal{C}$ produces transcripts according to the honest prover, then the internal emulation by $\mathcal{A}$ is identical to $\overline{\text{Hyb}}_1^i$. On the other hand if the transcripts received from $\mathcal{C}$ is according to the honest verifier simulator, then the internal emulation is identical to $\overline{\text{Hyb}}_1^{i+1}$. Therefore, $\mathcal{D}$ and $\mathcal{A}$ violates the honest verifier zero-knowledge property of π_{ZK}.

Claim 3.7. *The expected running time of $\mathcal{S}$ is polynomial.*

Proof: Assume for contradiction, the expected running time of $\mathcal{S}$ is not polynomial. Recall that the expected running time of $\mathcal{S}_1$ is some polynomial $t(n)$. Then we can construct a distinguisher that distinguishes the truncated experiments $\overline{\text{Hyb}}_1(n, x, z)$ and $\overline{\text{Hyb}}_2(n, x, z)$ defined above and this is a contradiction to the previous claim. We consider truncated experiments $\overline{\text{Hyb}}_1(n, x, z)$ and $\overline{\text{Hyb}}_2(n, x, z)$ where the experiments are truncated after $2t(n)$ steps. Next, consider a distinguisher $\mathcal{D}$ that outputs 1 if the experiment's output is $\perp$ and 0 otherwise. $\mathcal{D}$ on input view from $\overline{\text{Hyb}}_1(n, x, z)$ outputs 1 with probability at least $\frac{1}{2}$. However, $\mathcal{D}$ on input a view from $\overline{\text{Hyb}}_2(n, x, z)$ outputs 1 is negligible. Therefore, $\mathcal{D}$ distinguishes $\overline{\text{Hyb}}_1(n, x, z)$ and $\overline{\text{Hyb}}_2(n, x, z)$ with non-negligible probability and this is a contradiction. $\square$

4 Corollaries

In this section, we provide corollaries to our main techniques. We obtain the first round optimal fully black-box constructions of perfect zero-knowledge arguments and input-delayed commit-and-prove zero-knowledge argument.

4-round Perfect Zero-Knowledge Argument from Claw-free Permutations. As a corollary of Theorem 3.1, we prove that there exists a 4-round perfect zero-knowledge argument based on claw-free permutations. This is achieved by replacing the prover's commitments in Protocol 1 with perfectly hiding commitments which can be based on claw-free permutations. More formally,

Corollary 4.1. *Assuming claw-free permutations, there exists a 4-round fully black-box perfect zero-knowledge argument for any* NP *language.*

The protocol for the perfect zero-knowledge case is identical to the protocol described in Sect. 3 with the only exception that the commitments made by the prover is replaced with perfectly hiding commitments that can be based on claw-free permutations [GK96a]. The proof follows is analogous to the proof of Theorem 3.1. The soundness argument essentially remains unchanged; we only need to handle the case when the prover violates the binding property of the underlying commitment scheme. The zero-knowledge property follows essentially as before. We observe that the distributions in Hyb_0 and Hyb_1 are already proved to be identical. To conclude we observe that the distributions in Hyb_1 and Hyb_2 are also identical because the underlying commitment scheme is perfectly hiding.

4-round Input-Delayed Commit-and-Prove ZK Argument. As a second corollary, we prove that there exists a 4-round input delayed commit-and-prove zero-knowledge argument. This is achieved by replacing the three-round honest-verifier zero-knowledge argument based on Blum-Hamiltonicity with the three-round commit-and-prove input-delayed protocol of Hazay and Venkitasubramaniam [HV16] in Sect. 6.2. More formally,

Corollary 4.2. *Assuming injective one-way functions, there exists a fully black-box 4-round input-delayed commit-and-prove zero-knowledge argument for any* NP *language.*

Acknowledgments. We thank the anonymous TCC reviewers for their detailed comments and Rafael Pass for helpful suggestions. The first author was supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. The second author was supported by Google Faculty Research Grant and NSF Award CNS-1526377. This work was partly carried out by the second author during a visit to DIMACS supported by the National Science Foundation under grant number CNS-1523467.

References

[BJY97] Bellare, M., Jakobsson, M., Yung, M.: Round-optimal zero-knowledge arguments based on any one-way function. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 280–305. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_20

[BKP18] Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: STOC (2018)

[Blu] Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, USA, pp. 1444–1451 (1986). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.469.9048&rep=rep1&type=pdf

[CLOS02] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)

[CPS+15] Ciampi, M., Persiano, G., Scafuro, A., Siniscalchi, L., Visconti, I.: Improved OR composition of sigma-protocols. IACR Cryptol. ePrint Arch. **2015**, 810 (2015)

[FGJ18] Fleischhacker, N., Goyal, V., Jain, A.: On the existence of three round zero-knowledge proofs. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10822, pp. 3–33. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78372-7_1

[FS89] Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, New York (1990). https://doi.org/10.1007/0-387-34805-0_46

[GK96a] Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptol. **9**(3), 167–190 (1996)

[GK96b] Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. Comput. **25**(1), 169–192 (1996)

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)

[GMW87] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC, pp. 218–229, (1987)

[GMW91] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. J. ACM **38**(3), 691–729 (1991)

[GO94] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. J. Cryptol. **7**(1), 1–32 (1994)

[Gol01] Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, Cambridge (2001)

[HV16] Hazay, C., Venkitasubramaniam, M.: On the power of secure two-party computation. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 397–429. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_14

[IKOS07] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC, pp. 21–30 (2007)

[IKOS09] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. **39**(3), 1121–1152 (2009)

[IMS12] Ishai, Y., Mahmoody, M., Sahai, A.: On efficient zero-knowledge PCPs. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 151–168. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_9

[Kat12] Katz, J.: Which languages have 4-round zero-knowledge proofs? J. Cryptology **25**(1), 41–56 (2012)

[LS90] Lapidot, D., Shamir, A.: Publicly verifiable non-interactive zero-knowledge proofs. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 353–365. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_26

[Nao91] Naor, M.: Bit commitment using pseudorandomness. J. Cryptology **4**(2), 151–158 (1991)

[ORS15] Ostrovsky, R., Richelson, S., Scafuro, A.: Round-optimal black-box two-party computation. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 339–358. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_17

[PW09] Pass, R., Wee, H.: Black-Box constructions of two-party protocols from one-way functions. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 403–418. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_24