Which Languages Have 4-Round Fully Black-Box Zero-Knowledge Arguments from One-Way Functions?

Carmit Hazay[1], Rafael Pass[2], and Muthuramakrishnan Venkitasubramaniam[3]

[1] Bar-Ilan University
[2] Cornell Tech
[3] University of Rochester

Abstract. We prove that if a language $\mathcal{L}$ has a 4-round fully black-box zero-knowledge argument with negligible soundness based on one-way functions, then $\overline{\mathcal{L}} \in \mathsf{MA}$. Since $\mathsf{coNP} \subseteq \mathsf{MA}$ implies that the polynomial hierarchy collapses, our result implies that NP-complete languages are unlikely to have 4-round fully black-box zero-knowledge arguments based on one-way functions. In TCC 2018, Hazay and Venkitasubramaniam, and Khurana, Ostrovsky, and Srinivasan demonstrated 4-round fully black-box zero-knowledge arguments for all languages in NP based on injective one-way functions. Their results also imply a 5-round protocol based on one-way functions. In essence, our result resolves the round complexity of fully black-box zero-knowledge arguments based on one-way functions.

Keywords: One-Way Functions, 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 $x \in \mathcal{L}$, while providing zero additional knowledge to the verifier. Security against a cheating prover is formalized via soundness, which bounds its success probability to convince of the truthfulness of an incorrect statement. Whereas the zero-knowledge property 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 modeled as probabilistic polynomial-time machines (PPT), the traditional definition of ZK models both the verifier and the simulator as PPT machines.

Several different flavors of ZK systems have been studied in the 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 fully black-box constructions[4] and black-box simulation.[5] The main question we are interested in this work is the round-complexity of computational zero-knowledge argument systems based on minimal assumptions via a fully black-box construction.

We begin with a survey of prior work in this area. Goldreich, Micali and Wigderson [GMW91] constructed the first zero-knowledge proof system for all of NP based on any commitment scheme (which can be instantiated via a 2-round protocol based on one-way functions [Nao91,HILL99]), 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.

In this work, we are interested in fully black-box constructions based on the underlying assumptions. Pass and Wee [PW09] provided the first black-box construction of a 6-round zero-knowledge argument for NP based on one-way permutations,[6] and seven rounds based argument on one-way functions. Ishai, Mahmoody and Sahai provided the first black-box zero-knowledge arguments based on collision-resistant hash-functions that has total sublinear communication complexity [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, based on enhanced trapdoor permutations. More recently, in two independent works by Hazay and Venkitasubramaniam [HV18] and Khurana, Ostrovsky and Srinivasan [KOS18], 4-round fully black-box zero-knowledge arguments based on injective one-way function were demonstrated for all of NP.

On the negative side, Goldreich and Oren [GO94] demonstrated that three rounds are necessary for designing zero-knowledge arguments for any non-trivial language (i.e. outside BPP) against non-uniform verifiers. When further restricting to black-box simulation, Goldreich and Krawczyk [GK96] showed that four rounds are necessary for achieving zero-knowledge arguments of non-trivial languages. For the specific case of proofs, Katz [Kat12] showed that only languages in MA can have 4-round zero-knowledge proof systems. As

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

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

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

such, the works of [BJY97] and [GK96] identify the round-complexity of zero-knowledge arguments as four, when restricting to black-box simulation. The sequence of prior works leaves the following fundamental question regarding zero-knowledge arguments open:

> *What is the weakest hardness assumption for a fully black-box construction of a 4-round zero-knowledge argument system for all of* NP*?*
> *or*
> *Is there an inherent black-box barrier to design 4-round ZK arguments for all of* NP *based on one-way functions?*

We remark that when considering non-black-box simulation, a recent work due to Bitansky, Tauman Kalai and Paneth [BKP18] demonstrated how to obtain 3-round zero-knowledge arguments for NP based on multi-collision resistant hash functions. On the negative side, Fleischhacker, Goyal and Jain [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.

Our results. In this work we prove the infeasibility of 4-round black-box ZK arguments for all of NP from one-way functions. More formally, the main theorem we prove in this work is:

Theorem 1.1 (Main result.) *If $\mathcal{L}$ has a fully black-box construction of 4-round computational zero-knowledge argument for $\mathcal{L}$ with negligible soundness based on one-way functions, then $\overline{\mathcal{L}} \in \mathsf{MA}$.*

We remark that our result is essentially optimal on several fronts. In particular, if we relax the requirement of a black-box construction, then the work of [BJY97] showed how to construct 4-round ZK argument based on one-way functions. If we only required inverse polynomial soundness (as opposed to negligible soundness), then the classic GMW protocol [GMW91] when repeated in parallel a logarithmic number of times gives a 4-round ZK proof based on one-way functions with inverse polynomial soundness. If we relaxed one-way functions to injective one-way functions, then the works of [HV18,KOS18] demonstrates a 4-round zero-knowledge arguments for all of NP that is fully black-box based on one-way permutations. We highlight here that our impossibility result only requires that the zero-knowledge property holds w.r.t. one-way functions. In other words, we can show $\overline{\mathcal{L}} \in \mathsf{MA}$ even if the soundness of the underlying argument is based on one-way permutations. This matches the construction of [HV18]. Finally, we cannot hope to improve the theorem from MA to BPP as there exist languages (that are believed to be) outside of BPP (e.g., graph non-isomorphism) that have unconditional 4-round ZK proofs.

1.1 Our Techniques

On a high-level, our technique follows very closely the lower bound result of Katz [Kat12]. In this work, Katz proves that if a language $\mathcal{L}$ has a 4-round black-box zero-knowledge proof, then $\overline{\mathcal{L}} \in \mathsf{MA}$. As a warmup, we begin with an overview of this proof.

Suppose that we have a 4-round zero-knowledge proof for a language $\mathcal{L}$. The main idea is to design a malicious verifier $\mathcal{V}^*$ that satisfies the following properties:

- On a true statement $x \in \mathcal{L}$, $\mathcal{S}^{\mathcal{V}^*}$ will output an accepting transcript with high probability, where $\mathcal{S}$ is the simulator for this argument system.
- On a false statement $x \notin \mathcal{L}$, $\mathcal{S}^{\mathcal{V}^*}$ outputs an accepting transcript with a small probability.

Given such an algorithm $\mathcal{V}^*$, one can consider the following procedure to decide $\overline{\mathcal{L}}$: Run $\mathcal{S}^{\mathcal{V}^*}$. Then, reject if it outputs an accepting transcript and accept otherwise. If this procedure can be carried out via a PPT algorithm then it would imply $\mathcal{L} \in \mathsf{BPP}$. Since we know there are languages outside BPP which have 4-round zero-knowledge proofs (e.g., languages in SZKP), it is unlikely that we will be able to construct a $\mathcal{V}^*$ for which this decision procedure will be efficiently computable. Indeed, the algorithm $\mathcal{V}^*$ that is constructed in [Kat12] cannot be sampled via a PPT algorithm. Recall that the goal is to design an MA proof system for $\overline{\mathcal{L}}$. Katz shows that with some limited help from an unbounded Merlin, Arthur will be able to run the decision procedure, namely $\mathcal{S}^{\mathcal{V}^*}$. More concretely, Merlin will sample a string m from a prescribed distribution and send it to Arthur. Using m, Arthur will be able to run $\mathcal{S}^{\mathcal{V}^*}$. On a true statement (i.e. $x \in \overline{\mathcal{L}}$), Merlin will (honestly) provide the single message with the right distribution and Arthur will be able to decide correctly. Soundness, on the other hand, will require to argue that, for any *arbitrary* message sent by Merlin, Arthur rejects the statement with high probability. If the underlying zero-knowledge argument system admits perfect completeness then it becomes easy to argue that Merlin cannot provide "bad" messages that will make Arthur accept a false statement. The imperfect completeness case is more challenging. To make the proof system sound in the case of imperfect completeness, Katz showed a mechanism for Arthur to discard "bad" messages from Merlin. We now proceed to describe in more detail the lower bound in the case of imperfect completeness as we follow the ideas in this case closely.

We begin with a description of the malicious verifier $\mathcal{V}^*$ and then give our MA proof system. Roughly speaking, the malicious verifier $\mathcal{V}^*$ generates the first message according to the honest verifier $\mathcal{V}$ and will generate the third message depending on the second message of the prover by randomly sampling a random tape consistent with its first message. In more detail, we will consider $\mathcal{V}^*$ that takes as an auxiliary input random strings $r_1, \ldots, r_s$ under the promise that for every i, $\mathcal{V}(x; r_i)$ generates the same first message α. $\mathcal{V}^*$ then sends α as the first message and upon receiving the second message β from the prover, applies a pseudo-random function (a poly-wise independent hash-function is sufficient) on β to obtain an index $i \in [s]$. Finally, $\mathcal{V}^*$ uses r_i to generate the third message γ by running $\mathcal{V}$ with random tape r_i and the partial transcript so far.

We will need a procedure to sample a uniform α that is in the support of the verifier's first messages and then sample $r_1, \ldots, r_s$ uniformly over all consistent random tapes. This procedure will not be PPT computable (as otherwise, it would imply $\mathcal{S}^{\mathcal{V}^*}$ is efficiently computable and consequently $L \in \mathsf{BPP}$). As we only need to design an MA proof system, we will have Merlin (who is computationally unbounded) sample $r_1, \ldots, r_s$ and send these to Arthur. Before we describe the MA proof system, we first argue two properties:

1. If α is distributed according to the honest verifier algorithm with a uniform random tape, and r_i's are uniformly sampled conditioned on α, then the marginal distribution of any r_i will be uniform. This implies that, for $x \in \mathcal{L}$, if the r_i's were sampled correctly then for any i, $\mathcal{S}^{\mathcal{V}(x;r_i)}$ will output an accepting transcript with high probability. We show below that by the zero-knowledge property of the proof system, this implies that $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$ outputs an accepting transcript with high probability.
2. For $x \notin \mathcal{L}$ and r_i's sampled correctly, $\mathcal{S}^{\mathcal{V}^*}$ does not output an accepting transcript with high probability. This is argued by showing that if $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$ outputs an accepting transcript with high probability, then there exists a cheating prover $\mathcal{P}^*$ that can break soundness on input x with non-negligible probability. The idea here is, $\mathcal{P}^*$ will emulate $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$ internally and forward the outside execution inside in one of the rewinding sessions made by $\mathcal{S}$. In more detail, upon receiving the first message α from the verifier, $\mathcal{P}^*$ first samples $r_1, \ldots, r_s$ that are consistent with α as explained above. Next, it internally emulates $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$, with the exception that it forwards the messages of a random rewinding session to an external verifier. Now, if the chosen session is an accepting session then $\mathcal{P}^*$ convinces the external verifier to accept. Specifically, the analysis shows that $\mathcal{P}^*$ will convince the external verifier with probability at least μ/s where μ is the probability that $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$ outputs an accepting transcript.

Now consider the following MA proof system for $\overline{\mathcal{L}}$: Merlin samples a random first message α for the honest verifier and then samples several consistent random tapes $r_1 \ldots, r_s$, and sends them to Arthur. Arthur will run $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$.

If $\mathcal{S}$ outputs an accepting transcript, Arthur rejects and accepts otherwise. Completeness follows directly from Item 2, as Merlin will follow its actions honestly, making Arthur accept. Soundness, as mentioned before, requires that $r_1 \ldots, r_s$ are generated with the right distribution. If the underlying zero-knowledge protocol had perfect completeness, then arguing soundness becomes easy because for any set of random tapes $r_1, \ldots, r_s$ sent by Merlin, if they all are consistent with the same first message for the verifier, then by perfect completeness we will have that $\mathcal{S}^{\mathcal{V}^*}$ will output an accepting transcript with high probability. We discuss the case of imperfect completeness as it is more relevant to our techniques.

Handling imperfect completeness. If the original zero-knowledge system has imperfect completeness, then Merlin could select random tapes $r_1 \ldots, r_s$ that makes $\mathcal{S}^{\mathcal{V}^*}$ not output an accepting transcript, causing Arthur to accept.

To tackle this issue, as mentioned before, Katz introduces a procedure with which Arthur checks whether the r_i values are "good". First, we observe that if these strings were sampled correctly, then the marginal distribution of any of the r_i's will be uniform (Item 1). This implies that when running the simulator with the honest verifier with random tape r_i on a true statement, the simulator is expected to output an accepting transcript with high-probability.

Second, from the zero-knowledge property we have that for every set of random tapes $r_1, \ldots, r_s$:

$$\{i \leftarrow [t] : \mathcal{S}^{\mathcal{V}(x;r_i)}\} \approx \{i \leftarrow [t] : \langle \mathcal{P}(x), \mathcal{V}(x;r_i)\rangle\} \text{ and,}$$
$$\{\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}\} \approx \{\langle \mathcal{P}(x), \mathcal{V}^*(x,r_1,\ldots,r_s)\rangle\}.$$

Since the Verifier chooses r_i in its second round via pseudo-random function, we have that:[7]

$$\{i \leftarrow [t] : \langle \mathcal{P}(x), \mathcal{V}(x;r_i)\rangle\} \approx \{\langle \mathcal{P}(x), \mathcal{V}^*(x,r_1,\ldots,r_s)\rangle$$

This implies that, for any message $r_1, \ldots, r_s$ received from Merlin, if $\mathcal{S}^{\mathcal{V}(x;r_i)}$ outputs an accepting transcript for a randomly chosen i with high-probability, then $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$ must output an accepting transcript with high-probability. This gives rise to a checking procedure that can now be incorporated into the MA proof system. In more detail, the MA proof system is modified by asking Arthur to first check if $\mathcal{S}^{\mathcal{V}(x;r_i)}$ outputs an accepting transcript for a random i and reject otherwise. Only if the check passes, namely $\mathcal{S}^{\mathcal{V}(x;r_i)}$ outputs an accepting transcript, Arthur runs $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$ and decides accordingly. This gives an MA proof system that is sound. However, this modification alters the

[7] In fact, the distibutions are identical if the verifier uses poly-wise independent hash-functions.

completeness of the proof system, as $x \notin L$ could imply that $\mathcal{S}^{\mathcal{V}(x;r_i)}$ might not output an accepting transcript causing Arthur to reject immediately. This can be fixed by having Arthur first check if the simulator outputs an accepting transcript with the honest verifier on a uniformly sampled random tape by Arthur. More precisely, the final MA proof system has Arthur perform the following:

1. Run $\mathcal{S}^{\mathcal{V}(x;r)}$ several times. If $\mathcal{S}$ fails to output an accepting transcript with high probability where r is uniformly chosen in each trial, then accept and halt. Otherwise, proceed to the next step.
2. Pick a random index i and run $\mathcal{S}^{\mathcal{V}(x;r_i)}$. If $\mathcal{S}$ does not output an accepting transcript then reject and halt. Otherwise, proceed to the next step.
3. Run $\mathcal{S}^{\mathcal{V}^*(x,r_1,\ldots,r_s)}$. If $\mathcal{S}$ outputs an accepting transcript with high probability then reject, otherwise accept.

Our Approach. We now discuss how we extend this lower bound to our setting where we have a fully black-box construction of a 4-round zero-knowledge argument for $\mathcal{L}$. First, we observe that to consider the malicious verifier $\mathcal{V}^*$ as in Katz's proof, we need to provide $r_1, \ldots, r_s$ consistent with the first message in the presence of a one-way function oracle. Given an arbitrary oracle, we will not be able to sample randomness $r_1, \ldots, r_s$ even in unbounded time, if we are only allowed to make polynomially many queries to the oracle (which will be required as eventually, we want to use $\mathcal{V}^*$ to break soundness which is computational based on the one-wayness of the oracle). Instead, we will prescribe a joint distribution over $r_1, \ldots, r_s$ and random oracles for which we can carry out the proof. More precisely, given a statement x, we will specify a joint distribution over random oracles $\mathcal{O}$ and $r_1, \ldots, r_s$ such that for all i, $\mathcal{V}^{\mathcal{O}}(x; r_i)$ will output the same message and the following two properties hold:

Property P1 On a true statement x, $\mathcal{S}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\ldots,r_s)}$ will output an accepting transcript with high probability, where $\mathcal{S}$ is the simulator for this argument system.

Property P2 On a false statement x, $\mathcal{S}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\ldots,r_s)}$ outputs an accepting transcript with negligible probability.

Description of a malicious verifier strategy $\mathcal{V}^*$. We now proceed to describe our malicious verifier strategy and the corresponding random oracle distribution.

1. Run $\mathcal{V}^{\mathcal{O}}(x; r)$ where we emulate $\mathcal{O}$ as a random oracle and choose the verifier's random tape uniformly at random. Let α be the message output by $\mathcal{V}$. Discard r and the oracle $\mathcal{O}$.

2. Consider the oracle PPT algorithm $\mathcal{A}^\bullet$ that on random tape (r, r') outputs whatever $\mathcal{S}^{\bullet, \mathcal{V}^\bullet(x;r)}(x; r')$ outputs. We will next rely on the "heavy-query" learning procedure due to Barak and Mahmoody [BM07] who give a procedure to identify the most frequent queries made by an algorithm to the random oracle conditioned on its output being fixed to a particular message. We apply the heavy query learning procedure to the honest verifier algorithm $\mathcal{V}$ subject to the condition that it outputs α as its first message. Let $\mathcal{Q}$ be the set of queries output by this procedure for some oracle $\mathcal{O}'$ sampled as a random oracle.
3. Let R_α be the set that contains all the pairs $(r', \mathcal{Q}')$ such that $\mathcal{V}(x; r')$ outputs α as its first message while making queries only in $\mathcal{Q} \cup \mathcal{Q}'$ (where $\mathcal{Q}'$ are the non-frequent queries). Now, sample s elements $\{(r_i, \mathcal{Q}_i)\}_{i \in [s]}$ from R_α uniformly at random.
4. Output $(r_1, \ldots, r_s)$ and $(\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s)$.

Given a sample $(r_1, \ldots, r_s)$ and $(\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s)$, the distribution of oracles will be random oracles whose queries in $(\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s)$ are fixed and set to be random on all other points. Such oracles were previously considered in [MP12] and referred to as partially-fixed random oracles. The malicious verifier $\mathcal{V}^*$ is specified as a PPT algorithm that takes as auxiliary information $(r_1, \ldots, r_s)$ and proceeds as follows. For the first message, it runs $\mathcal{V}(x; r_1)$ and outputs whatever $\mathcal{V}$ does, say α. Given a second message β sent by the prover $\mathcal{V}^*$ applies a poly-wise independent hash function (also supplied as auxiliary information) $h(\beta)$ to a chosen index $i \in [s]$. Then it runs $\mathcal{V}(x; r_i)$ on the partial transcript α, β to output the third message δ and forwards that to the prover. Any oracle query made by $\mathcal{V}$ is forwarded to the oracle attached to $\mathcal{V}^*$.

Proving P1 follows essentially the same way as in [Kat12]. So we argue P2 next.

Proving P2. Just as in [Kat12], we will show that if the simulator can simulate $\mathcal{V}^*$ on a false statement with non-negligible probability, then there exists a cheating prover $\mathcal{P}^*$ that can break the soundness of the zero-knowledge argument, which, in turn, establishes the property P2 specified at the beginning of the outline. As before, in the security reduction, $\mathcal{P}^*$ will internally emulate the simulator with $\mathcal{V}^*$ and forward the message from the external interaction inside, for one of the random rewindings made by the simulator. Recall that $\mathcal{P}^*$ and the external verifier are equipped with an oracle $\mathcal{O}$ (for the one-way function).

Observe that $\mathcal{P}^*$ will not be able to use $\mathcal{O}$ for internally emulating $\mathcal{S}^{\mathcal{V}^*}$, as in the internal execution $\mathcal{P}^*$ it needs to run $\mathcal{S}$ and $\mathcal{V}^*$ from a prescribed distribution over $r_1, \ldots, r_s$ and random oracles. By applying the same learning heavy-query algorithm we can show that $\mathcal{P}^*$ will be able to sample $\mathcal{Q}$, $\mathcal{Q}_1, \ldots, \mathcal{Q}_s$ and $r_1, \ldots, r_s$ and an oracle $\mathcal{O}'$ where

– $\mathcal{Q}$ is consistent with $\mathcal{O}$.
– $\mathcal{O}'$ is consistent with $\mathcal{Q} \cup \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$ and with $\mathcal{O}$ everywhere else.
– If $\mathcal{O}$ is sampled according to a random oracle, then the distribution of $\mathcal{O}'$ and $r_1, \ldots, r_s$ is identical to the prescribed distribution.

Next, if the random rewinding chosen by $\mathcal{P}^*$ is the one that the simulator outputs as an accepting transcript, then we want to conclude that $\mathcal{P}^*$ succeeds in convincing the external verifier. There are two (related) issues to make this argument work:

– First, forwarding the messages from the external verifier internally in a random rewinding session could result in skewing the distribution internally simulated by $\mathcal{P}^*$.
– Second, the external oracle $\mathcal{O}$ and the internally emulated oracle $\mathcal{O}'$ are not identical. In particular, they could be different on $\mathcal{Q}_1, \ldots, \mathcal{Q}_s$.

We argue that the first item is not an issue and the distribution is, in fact, correct because we can view the random tape and queries made by the outside verifier as one of the elements in R_α. The second issue is problematic because if the messages generated by the simulator in the forwarded session makes the external verifier make one of the conflicting queries (namely a query on $\mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$), then we cannot claim that the external verifier will accept if the internal emulation results in an accepting transcript on that session. To resolve this issue, we weaken property P2 as follows:

P2' On a false statement x, $\mathcal{S}^{\mathcal{O}, \mathcal{V}^{*\mathcal{O}}(x, r_1, \ldots, r_s)}$ outputs an accepting transcript *while not making conflicting queries* with negligible probability. In particular, if a particular rewinding session (where r_j was used as the random tape) is the accepting transcript then the verifier on that transcript should not make any query to $\mathcal{Q}_i$ for $i \neq j$.

This modification will be the crux of making our MA proof system work.

MA **proof system.** Upon receiving $r_1, \ldots, r_s$, $\mathcal{Q}$, $\mathcal{Q}_1, \ldots, \mathcal{Q}_s$, Arthur continues as follows:

1. Emulate $\mathcal{S}^{\mathcal{O}, \mathcal{V}^{\mathcal{O}}(x; r)}$ where r is chosen at random and $\mathcal{O}$ according to the random oracle. If it does not output an accepting transcript, then accept and halt. Otherwise proceed.

2. Pick a random $i \leftarrow [s]$ and emulate $\mathcal{S}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}$ where $\mathcal{O}$ is sampled according to a partially fixed random oracle, fixed on the set $\mathcal{Q} \cup \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$. If it either does not output an accepting transcript or outputs an accepting transcript with conflicting queries, then reject and halt. Otherwise, proceed.
3. Emulate $\mathcal{S}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\ldots,r_s)}$. If it either does not output a transcript or an accepting transcript is output with conflicting queries then accept. Otherwise, reject.

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 $X \overset{\text{c}}{\approx} Y$, if for every* PPT *machine $\mathcal{D}$, every $a \in \{0,1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large n:*

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

We assume familiarity with the basic notions of an Interactive Turing Machine (ITM for brevity) and a protocol (in essence a pair of ITMs). We denote by PPT the class of probabilistic polynomial-time Turing machines. We denote by $M^\bullet$ an oracle machine; we sometimes drop $\bullet$ when it is clear from the context. As usual, if $M^\bullet$ is an oracle machine, $M^{\mathcal{O}}$ denotes the joint execution of M with oracle access to $\mathcal{O}$.

Definition 2.2 (Random Oracle) *A random oracle* **RO** *is a randomized stateful oracle that given a query $x \leftarrow \{0,1\}^n$ outputs y if the pair (x, y) is stored or outputs a random element y' from $\{0,1\}^{|x|}$ and stores (x, y').*

Following [BR93,MP12], we use randomized oracles as opposed to fixing a random oracle by sampling it once as in [IR89] as this is sufficient for refuting black-box constructions.

We recall the properties of the "heavy-query" learning algorithm (verbatim) from [BM07] that have typically been used in separation from one-way functions [IR89,MP12].

Lemma 2.1 (Learning Heavy Queries Efficiently [BM07]) *Let $\mathcal{A}$ be a randomized oracle algorithm which asks up to m oracle queries, denoted by $\mathcal{Q}(\mathcal{A}^{\mathcal{O}})$ and outputs some message C. Let $0 < \varepsilon < 1$ be a given parameter. There is a learning algorithm G in PSPACE (in fact, $\mathsf{BPP}^{\mathsf{NP}}$) which learns a list of $\mathcal{Y}$ of query-answer pairs from the oracle $\mathcal{O}$ such that:*

1. *$|\mathcal{Y}| \leq 10m/\varepsilon^2$.*
2. *With probability at least $1-\varepsilon$ over the choice of $\mathcal{O}$ from $\mathbf{RO}$ and the random coins of $\mathcal{A}$ and G, for every u that is not part of any query-answer pair in $\mathcal{Y}$,it holds that $\Pr[u \in \mathcal{Q}(\mathcal{A}) | (C, \mathcal{Y})] < \varepsilon$ where the latter probability is over the remaining randomness of $\mathbf{RO}$ and $\mathcal{A}$ conditioned on $(C, \mathcal{Y})$.*

Next, we recall the property about random oracles that they cannot be inverted by any oracle algorithm (possibly unbounded) that makes only polynomially many queries to the oracle. The following is repeated verbatim from [MP12].

Definition 2.3 (Security Threshold) *A primitive P has security threshold τ_P if an adversary "breaking" P has to "win" in the security game of P with probability $\tau_P + \varepsilon$ for a non-negligible ε.*

Lemma 2.2 ([BM07,MP12]) *Let P and Q be two cryptographic primitives and P has security threshold zero. For a randomized oracle $\mathcal{O}$, suppose one can break the black-box security of any implementation $Q^{\mathcal{O}}$ of Q with non-negligible probability and asking $poly(n)$ oracle queries to $\mathcal{O}$. Suppose also that there exists a black-box secure implementation $P^{\mathcal{O}}$ of P from $\mathcal{O}$. Then there is no black-box construction of Q from P.*

Definition 2.4 (Partially-Fixed Random Oracles) *We call a randomized function f a $k(n)$-partially-fixed random oracle if it is fixed over some sub-domain S and chooses its answers similarly to the random oracle $\mathbf{RO}$ at any point q out of S and it holds that $|S \cap \{0,1\}^n| \leq k(n)$ for every n. We simply call f partially-fixed random if it is $2^{o(n)}$-partially-fixed random.*

Lemma 2.3 ([MP12]) *One-way functions can be black-box securely realized from all partially-fixed random oracles.*

2.1 Fully Black-box Constructions

Following the terminology of [RTV04], we consider fully black-box constructions of zero-knowledge arguments from the underlying primitive.

Definition 2.5 (Fully black-box construction) *A black-box implementation of a primitive $\mathcal{Q}$ from a primitive $\mathcal{P}$ is an oracle algorithm Q (referred to as the implementation) such that Q^P is an implementation of $\mathcal{Q}$ whenever P is an implementation of $\mathcal{P}$. Q^P is said to have a black-box proof of security, if there exists an efficient machine $\mathcal{R}$ such that for any oracle P implementing $\mathcal{P}$ and machine $\mathcal{A}$ that breaks Q^P with non-negligible advantage for some security parameter n, then $\mathcal{R}^{P,\mathcal{A}}$ breaks the security of P over some security parameter $n' = \mathsf{poly}(n)$. A black-box construction $\mathcal{Q}$ from $\mathcal{P}$ requires a black-box implementation Q and a black-box proof of security $\mathcal{R}$.*

2.2 Interactive Systems

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).

A round of an interactive proof system consists of a message sent from one party to the other, and we assume that the prover and the verifier speak in alternating rounds. Following [BM88], we let MA denote the class of languages having a 1-round proof system and in this case refer to the prover as Merlin and the verifier as Arthur; that is:

Definition 2.6 (MA) $\mathcal{L} \in \mathsf{MA}$ *if there exists a probabilistic polynomial-time verifier $\mathcal{V}$, a non-negative function s, and a polynomial p such that the following hold for all sufficiently-long x:*

- *If $x \in \mathcal{L}$ then there exists a string w (that can be sent by Merlin) such that*

$$\Pr[\mathcal{V}(x, w) = 1] \geq s(|x|) + 1/p(|x|).$$

- *If $x \notin \mathcal{L}$ then for all w (sent by a cheating Merlin) it holds that*

$$\Pr[\mathcal{V}(x, w) = 1] \leq s(|x|).$$

Definition 2.7 (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 $\mu(\cdot)$ such that the following two conditions hold:*

1. COMPLETENESS: *For every $x \in \mathcal{L}$ there exists a string ω such that for every $z \in \{0,1\}^*$,*
$$\Pr[\langle \mathcal{P}(\omega), \mathcal{V}(z)\rangle(x) = 1] \geq c(|x|)$$
where c is the acceptance probability.
2. SOUNDNESS: *For every $x \notin \mathcal{L}$, every interactive* PPT *machine $\mathcal{P}^*$, and every $\omega, z \in \{0,1\}^*$*
$$\Pr[\langle \mathcal{P}^*(\omega), \mathcal{V}(z)\rangle(x) = 1] \leq s(|x|).$$
where s is the soundness error and will be negligible in this paper.

Definition 2.8 (Computational zero-knowledge (CZK)) *Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for some language $\mathcal{L}$. We say that $(\mathcal{P}, \mathcal{V})$ is a* 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*
$$\{\langle \mathcal{P}(\omega), \mathcal{V}^*(z)\rangle(x)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_x,z\in\{0,1\}^*} \overset{c}{\approx} \{\langle \mathcal{S}\rangle(x,z)\}_{x\in\mathcal{L},z\in\{0,1\}^*}$$
(when the distinguishing gap is considered as a function of $|x|$). Specifically, the left term denotes the output of $\mathcal{V}^*$ after it interacts with $\mathcal{P}$ on common input x whereas, the right term denotes the output of $\mathcal{S}$ on x.

Black-Box Construction of Zero-Knowledge Arguments

Definition 2.9 *A black-box construction of a zero-knowledge argument system for a language $\mathcal{L}$ from one-way functions is a tuple of oracle algorithms $(\mathcal{P}, \mathcal{V}, \mathcal{S})$ such that for any oracle $f = \{f_m : \{0,1\}^m \to \{0,1\}^m\}$, $\mathcal{P}, \mathcal{V}$ and $\mathcal{S}$ are oracle algorithms where completeness holds w.r.t to any oracle $\mathcal{O}$ and the soundness and zero-knowledge property are proved via a reduction to the underlying function f as follows:*

Soundness: *There is an efficient oracle reduction algorithm $\mathcal{R}_s$, such that for every oracle f, every malicious prover $\mathcal{P}^*$ (that could arbitrarily depend on f), if $\mathcal{P}^*$ convinces the verifier on input $x \in \{0,1\}^n \backslash \mathcal{L}$ with probability $1/p(n)$ for some polynomial $p(\cdot)$, $\mathcal{R}_s^{f,\mathcal{P}^{*f}}$ inverts f with probability $1/q(m)$ for some polynomial $q(\cdot)$ over a polynomially related $m = n^{\theta(1)}$, namely,*
$$\Pr[y \leftarrow f(U^m) : \mathcal{R}_s^{f,\mathcal{P}^{*f}}(y) \in f^{-1}(y)] \geq \frac{1}{q(m)}$$

Zero Knowledge: *This is defined analogously to the soundness property. There is an efficient oracle reduction algorithm $\mathcal{R}_{zk}$, such that for every oracle f, every malicious verifier $\mathcal{V}^*$ (that could arbitrarily depend on f), if $\mathcal{V}^*$ distinguishes the real execution from the simulation on input $x \in \mathcal{L} \cap \{0,1\}^n$ with probability $\frac{1}{p(n)}$ for some polynomial $p(\cdot)$, $\mathcal{R}_{zk}^{f,\mathcal{V}^{*f}}$ inverts f with probability $1/q(m)$ for some polynomial $q(\cdot)$ over a polynomially related $m = n^{\theta(1)}$, namely,*

$$\Pr[y \leftarrow f(U^m) : \mathcal{R}_{zk}^{f,\mathcal{V}^{*f}}(y) \in f^{-1}(y)] \geq \frac{1}{q(m)}$$

We remark that, by view of the verifier we include the transcript of the messages, random tape and the query and answers obtained by the verifier from its oracle.

Terminology. We will be concerned with 4-round CZK argument systems, where the verifier sends the first message and the prover sends the final message. We use $\alpha, \beta, \gamma, \delta$ to denote the first, second, third, and fourth messages, respectively. We let $\mathcal{P}$ (resp., $\mathcal{V}$) denote the honest prover (resp., honest verifier) algorithm when the common input is x.

3 Implausibility of 4-Round BB ZK Arguments from OWFs

We begin with an outline of the proof. Recall that any separation cannot rule out the existence of 4-round arguments with a random oracle, as a random oracle with high probability acts as a "one-way permutation" and we do know 4-round arguments based on one-way permutations [HV18,KOS18]. Instead, we follow the approach of [MP12], by considering partially-fixed random oracles that crucially rely on the fact that the distribution of oracles is not a permutation. A partially fixed random oracle behaves essentially as a random oracle with the exception that for a pre-specified subset $\mathcal{F}$ of its domain the answers are fixed.

3.1 Main Result

We are ready to prove our main result.

Theorem 3.1 *If $\mathcal{L}$ has a fully black-box construction of 4-round computational zero-knowledge argument for $\mathcal{L}$ with negligible soundness based on one-way functions, then $\overline{\mathcal{L}} \in \mathsf{MA}$.*

Proof. Assume for contradiction, there is a fully black-box construction of a 4-round ZK argument $(\mathcal{P}, \mathcal{V})$ from a one-way function with black-box simulator $\mathcal{S}$.

In the proof system, Merlin (namely, the prover) and Arthur (namely, the verifier) share in advance an input x of length n. Let $c(\cdot)$ be the completeness of $\langle \mathcal{P}, \mathcal{V} \rangle$. The soundness of $\langle \mathcal{P}, \mathcal{V} \rangle$ is negligible. Let $T_s(n)$ be a bound on the expected running time of the simulator. Let $m(n)$ be the total number of queries made by the prover and the verifier on inputs of length n. Let $T_v(n)$ be a bound on the runtime of the honest verifier. Let $\eta(n)$ denote the length of the prover's second message. We set $\varepsilon(n) = c(n)/20$, and $s'(n) = 4(T_s(n))^2(\varepsilon)^{-3}$. For sake of succinctness, we define $m = m(n)$, $c = c(n)$, $T = T_s(n)$, $\ell = T_v(n)$, $\eta = \eta(n)$, $\varepsilon = \varepsilon(n)$ and $s = s'(n)$. Finally, let $\widetilde{\mathcal{S}}$ be the algorithm that proceeds identically to $\mathcal{S}$ with the exception that it halts after $2T/\varepsilon$ steps on inputs of length n.

We will first describe a distribution of a malicious verifier $\mathcal{V}^*$ and oracles $\mathcal{O}$ and then describe and analyze the MA proof system.

Specifying the distribution of malicious verifier and the oracle.

1. Run $\mathcal{V}^{\mathcal{O}}(x; r)$ where we emulate $\mathcal{O}$ as a random oracle and choose the verifier's random tape uniformly at random. Let α be the message output by $\mathcal{V}$. Discard r and the oracle $\mathcal{O}$.
2. Consider the oracle PPT algorithm $\mathcal{A}$ that on random tape (r, r') outputs what $\mathcal{S}^{\bullet, \mathcal{V}^{\bullet}(x;r)}(x; r')$ outputs. We execute the heavy-query learning procedure for the algorithm $\mathcal{A}$ from Lemma 2.1 with parameter $\frac{\epsilon}{(2s^2 \cdot \ell)}$ subject to the condition that the output contains the view of the verifier where the first message generated by $\mathcal{V}$ is α. Let $\mathcal{Q}$ be the set of queries output by this procedure.
3. Let R_α be the set that contains all the pairs $(r', \mathcal{Q}')$ such that $\mathcal{V}(x; r')$ outputs α as its first message while only making oracle queries inside $\mathcal{Q} \cup \mathcal{Q}'$. Now sample s elements $\{(r_i, \mathcal{Q}_i)\}_{i \in [s]}$ from R_α uniformly at random.
4. Output $(r_1, \ldots, r_s)$ and $(\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s)$.

Description of a malicious verifier strategy $\mathcal{V}^*$: Given $r_1, \ldots, r_s$ from the distribution above, we consider an oracle PPT algorithm $\mathcal{V}^*$, that given an input x and auxiliary input $r_1, \ldots, r_s, h$, where r_i represents random coins for the honest verifier algorithm and h is a hash function, proceeds as follows:

1. $\mathcal{V}^*$ internally emulates the honest verifier oracle algorithm $\mathcal{V}$ on input x and random tape r_1 to generate its first message α which it forwards externally to the prover. If at any point during the emulation, $\mathcal{V}$ makes a query to its oracle, $\mathcal{V}^*$ forwards that query to its oracle and the response back to $\mathcal{V}$.
2. Upon receiving a message β from the prover, the verifier computes $i = h(\beta)$ and emulates $\mathcal{V}$ on input x with random tape r_i. It obtains α as $\mathcal{V}$'s first message and feeds β as the prover's message. It then obtains γ as the third message and $\mathcal{V}^*$ forwards γ to the external prover.

3. $\mathcal{V}^*$ receives the last message δ from the prover. Finally, $\mathcal{V}^*$ outputs its view.

Description of the family of oracles. Given $\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s$, we consider a partially-fixed random oracle $\widetilde{\mathbf{O}}$ that is defined as follows. It contains oracles that are fixed over the queries in $\mathcal{Q} \cup \mathcal{Q}_1 \cup \ldots \cup \mathcal{Q}_s$ and chooses its answers similarly to the random oracle $\mathbf{RO}$ at any point q not in the subdomain defined by $\mathcal{Q} \cup \mathcal{Q}_1 \cup \ldots \cup \mathcal{Q}_s$. We remark that such a family is well defined only if no two sets among $\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s$ have *conflicting queries*, where a query u is conflicting for query-answer sets A and B, if there exists v_1, v_2 (possibly equal) such that $(u, v_1) \in A$ and $(u, v_2) \in B$. Looking ahead, by the properties of the learning algorithm employed in the sampling procedure described above, we will have that there will be no conflicting queries with high probability.

Before proceeding with the proof we introduce some notation, borrowed verbatim from [Kat12]. For a given randomized experiment Expt that can be run in polynomial-time and outputs a bit, we let $\mathsf{Estimate}_\varepsilon(\Pr[\mathsf{Expt}])$ denote a procedure that outputs an estimate to the given probability (taken over randomness used in the Expt) to within an additive factor of ε, except with probability at most ε. That is:

$$|\Pr\left[\mathsf{Estimate}_\varepsilon(\Pr[\mathsf{Expt}=1]) - \Pr[\mathsf{Expt}=1]| \geq \varepsilon\right] \leq \varepsilon.$$

This can be done in the standard way using $\Theta(\varepsilon^{-2}\log\frac{1}{\varepsilon})$ independent executions of Expt. Observe that if ε is non-negligible then the estimation runs in polynomial time whenever Expt is a polynomial-time sampleable.

Description of the MA proof system: We are now ready to describe an MA proof for $\overline{\mathcal{L}}$. On input x, Arthur proceeds as follows:

1. Upon receiving Merlin's first message, Arthur interprets the message as strings $r_1, \ldots, r_s \in \{0,1\}^\ell$ and sets of query-answer pairs $\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s$. Next, it proceeds as follows:
   (a) Estimate the probability:

$$p_1 = \mathsf{Estimate}_\varepsilon\left(\Pr_{r',r,\mathcal{O}}\left[\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r')}(x;r) \text{ outputs an accepting transcript }\right]\right)$$

   where r and r' are chosen uniformly at random from $\{0,1\}^T$ and $\{0,1\}^\ell$ respectively, and $\mathcal{O}$ is sampled according to $\mathbf{RO}$. We remark that the estimation procedure requires sampling a random execution of $\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r')}(x;r)$ and this can be done in polynomial time by emulating $\mathcal{O}$ distributed according to a random oracle $\mathbf{RO}$. If $p_1 < c - 2\varepsilon$ then accept and halt. Otherwise, proceed to the next step.

(b) If any pair of the sets $\mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s$ have conflicting queries then reject and halt. Else, emulate the honest verifier algorithm $\mathcal{V}$ on input x and random tape r_1 until it generates its first message α. In this emulation, if $\mathcal{V}$ makes a query inside $\mathcal{Q} \cup \mathcal{Q}_1$ we respond with the corresponding answer from the set. If $\mathcal{V}$ makes a query outside $\mathcal{Q} \cup \mathcal{Q}_1$, then Arthur rejects. For every $i \in [s]$, internally emulate $\mathcal{V}(x; r_i)$ and reject if it does not output the same α as its first message or makes a query outside $\mathcal{Q} \cup \mathcal{Q}_i$.

(c) Denote by $\widetilde{\mathbf{O}}$ the distribution of partially-fixed random oracles fixed on the set $\mathcal{Q} \cup \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$. Let $E(v)$ denote the event when a view v of the verifier is consistent with $\mathcal{V}(x; r_i)$ for some $i \in [s]$ and contains no query from $\mathcal{Q}_j$ for any $j \neq i$ (where consistent with $\mathcal{V}(x; r_i)$ means that transcript in v can be regenerated when the prover messages in v are fed to the honest verifier's code on input x and randomness r_i). Pick a random $i \in [s]$ and emulate $\widetilde{\mathcal{S}}^{\mathcal{O}, \mathcal{V}^{\mathcal{O}}(x; r_i)}(x; r)$ where the oracle $\mathcal{O}$ is emulated according to $\widetilde{\mathbf{O}}$. Such an oracle can be emulated by answering all queries in the fixed set according to the query-answer pair and any other query randomly (but consistently). If either $\widetilde{\mathcal{S}}$ does not output an accepting transcript or $E(v)$ does not hold for the view v output by $\widetilde{\mathcal{S}}$, then reject.

(d) Let H denote a family of $2T/\varepsilon$-wise independent hash function $h : \{0,1\}^\eta \to \{1, \ldots, s\}$. We next estimate:

$$p_2 = \mathsf{Estimate}_\varepsilon \left(\Pr_{r,h,\mathcal{O}} \left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O}, \mathcal{V}^{*\mathcal{O}}(x, r_1, \ldots, r_s, h)}(x; r) : v \text{ is accepting } \wedge E(v) \right] \right),$$

where $r \leftarrow \{0,1\}^T$, $h \leftarrow H$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$. If $p_2 < c - 10\varepsilon$ accept, otherwise reject.

We now proceed to proving the completeness and soundness arguments of the above proof.

Lemma 3.1 *For any $x \notin \overline{\mathcal{L}} \cap \{0,1\}^n$ and sufficiently large n, and any message $r_1, \ldots, r_s, \mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s$ sent by Merlin, the probability that Arthur accepts is at most $c - 6\varepsilon$.*

Proof: In this case, we have $x \in \mathcal{L}$, so it must hold that for any oracle $\mathcal{O}$ that the probability with which the honest prover convinces the honest verifier on input x and oracle $\mathcal{O}$ is at least c. From the zero-knowledge property we have that, for sufficiently large n, $x \in \{0,1\}^n \cap \mathcal{L}$, we have

$$\Pr_{r,r',\mathcal{O}} \left[\widetilde{\mathcal{S}}^{\mathcal{O}, \mathcal{V}^{\mathcal{O}}(x; r')}(x; r) \text{ outputs an accepting transcript} \right] \geq c - \varepsilon$$

This means that with probability at most ε, the estimate p_1 obtained by Arthur will be smaller than $c - 2\varepsilon$. In other words, Arthur accepts the statement with probability at most ε in Step 1a.

Next recall that if the message sent by Merlin does not meet the conditions in Step 1b, then it rejects. Thus we will assume that these conditions hold. Now consider the following probability

$$\hat{p} = \Pr_{i,r,h,\mathcal{O}}\left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}(x;r) : \mathsf{v} \text{ is accepting } \wedge E(\mathsf{v})\right]$$

where $i \leftarrow [s]$, $r \leftarrow \{0,1\}^T$, $h \leftarrow H$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$. Recall that if $p_2 < c - 10\varepsilon$ then Arthur accepts, and otherwise rejects. There are two cases depending on $\hat{p}$.

Case $\hat{p} < c - 7\varepsilon$: Recall that, in Step 1c, Arthur picks a random i, emulates $\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}(x;r)$ and rejects if the simulator does not output an accepting transcript. Therefore, in this case, the probability with which Arthur accepts is at most the probability that Arthur proceeds beyond Step 1c which is at most $c - 7\varepsilon$.

Case $\hat{p} \geq c - 7\varepsilon$: In this case, by the zero-knowledge property, we have that the probability that the honest prover convinces the verifier with $\mathcal{O}$ and E does not occur, is at least $c - 8\varepsilon$. In other words,

$$\Pr_{i,\mathcal{O}}\left[v \leftarrow \mathbf{View}_{\mathcal{V}}(\langle \mathcal{P}^{\mathcal{O}}, \mathcal{V}^{\mathcal{O}}(r_i)\rangle(x)) : v \text{ is accepting } \wedge E(v)\right] \geq c - 8\varepsilon.$$

where $i \leftarrow [s]$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$. Recall that $\widetilde{\mathbf{O}}$ is partially-fixed random oracle fixed over a polynomial-sized subdomain and from Lemma 2.3 (as shown in [MP12]) we know it implies one-way functions. We remark that here we rely on the fact that the zero-knowledge property holds w.r.t such one-way functions.

By our construction of $\mathcal{V}^*$ and $2T/\varepsilon$-wise independence of H, it holds that

$$\Pr_{h,\mathcal{O}}\left[v \leftarrow \mathbf{View}_{\mathcal{V}^*}(\langle \mathcal{P}^{\mathcal{O}}, \mathcal{V}^{*\mathcal{O}}(r_1,\ldots,r_s,h)\rangle(x) : v \text{ is accepting } \wedge E(v)\right]$$
$$= \Pr_{i,\mathcal{O}}\left[v \leftarrow \mathbf{View}_{\mathcal{V}}(\langle \mathcal{P}^{\mathcal{O}}, \mathcal{V}^{\mathcal{O}}(r_i)\rangle(x)) : v \text{ is accepting } \wedge E(v)\right]$$

where $i \leftarrow [s]$, $h \leftarrow H$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$.

Using the zero-knowledge property again, but, with $\mathcal{V}^*$ this time we have that

$$\Pr_{r,h,\mathcal{O}}\left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\ldots,r_s,h)}(x;r) : v \text{ is accepting } \wedge E(v)\right] \geq c - 9\varepsilon$$

where $r \leftarrow \{0,1\}^T$, $h \leftarrow H$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$. This means that the probability with which Arthur accepts in Step 1d is at most ε.

Overall, the probability with which Arthur accepts is at most $\varepsilon + \max\{c - 7\varepsilon, \varepsilon\} = c - 6\varepsilon$ and this concludes the proof of the lemma. $\square$

Lemma 3.2 *For any $x \in \overline{\mathcal{L}} \cap \{0,1\}^n$ and sufficiently large n, there is a strategy for Merlin that makes Arthur accept with probability is at least $c - 5\varepsilon$.*

Proof: We first define Merlin's strategy. Merlin will internally maintain the state of an oracle $\mathcal{O}$ that is sampled according to **RO**. It chooses $\tilde{r} \leftarrow \{0,1\}^\ell$ uniformly at random and emulates $\mathcal{V}^{\mathcal{O}}(x;\tilde{r})$ and computes the verifier's first message α. Next, it runs the simulator with the honest verifier and tries to learn all the heavy queries made by the algorithm $\widetilde{\mathcal{S}}^{\bullet,\mathcal{V}^{\bullet}(x;r')}(x;r)$ subject to the verifier's first message being α and the oracle being $\mathcal{O}$ where the learning parameter is set to $\frac{\varepsilon}{(2s^2\cdot\ell)}$. Let $\mathcal{Q}$ be the set of the queries that Merlin learns. Let R_α be the set that contains all the pairs $(r', \mathcal{Q}')$ such that $\mathcal{V}^{\mathcal{Q}\cup\mathcal{Q}'}(x;r')$ outputs α as its first message. Then Merlin samples s elements $\{(r_i, \mathcal{Q}_i)\}_{i\in[s]}$ from R_α uniformly at random and sends $r_1, \dots, r_s, \mathcal{Q}, \mathcal{Q}_1, \dots, \mathcal{Q}_s$ to the Arthur.

We now proceed to analyze the probability Arthur accepts. Recall that in Step 1a, Arthur accepts if the estimate $p_1 < c - 2\varepsilon$. Let

$$\hat{p} = \Pr_{r,r',\mathcal{O}}\left[\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r')}(x;r) \text{ outputs an accepting transcript}\right]$$

where $r \leftarrow \{0,1\}^T$, $r' \leftarrow \{0,1\}^\ell$, $\mathcal{O} \leftarrow$ **RO**. We consider two cases:

Case $\hat{p} < c - 3\varepsilon$: In this case, by our estimation algorithm, we have that except with probability ε, Arthur will accept at the end of Step 1a.

Case $\hat{p} \geq c - 3\varepsilon$: In this case, we consider Step 1b, where Arthur checks if there are no conflicting queries. Since Merlin honestly samples from the right distribution, we have that for each $i \in [s]$, the Verifier $\mathcal{V}$ outputs α with random tape r_i while making queries only in $\mathcal{Q} \cup \mathcal{Q}_i$ where $\mathcal{Q}$ and $\mathcal{Q}_i$ dont have any conflicting queries. Second, it follows from the properties of the learning algorithm as stated in Lemma 2.1 and the parameters that was set, that the probability that any query from $\mathcal{Q}_i$ occurs in $\mathcal{Q}_j$ for $j \neq i$ with probability at most $\frac{\varepsilon}{(2s^2\cdot\ell)}$. Using a union bound we have that the probability that some two sets in $\mathcal{Q}_1, \dots, \mathcal{Q}_s$ have conflicting queries can be bounded by $s \times (|\mathcal{Q}_i| \times s \times \frac{\varepsilon}{2s^2\cdot\ell}) < \frac{\epsilon}{2}$. Therefore, the probability that Arthur rejects in Step 1b is at most $\frac{\varepsilon}{2}$.

In Step 1c, Arthur emulates $\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}$ for a randomly chosen i and aborts if it either does not output a transcript or the $E(v)$ holds for the view output by the simulator.

First, we observe that, from Merlin's algorithm, the following two distributions are identical:

- $\{\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}(x;r)\}$ where $r_1,\dots,r_s$, $\mathcal{Q}$, $\mathcal{Q}_1,\dots,\mathcal{Q}_s$ are sampled according to Merlin's algorithm, $i \leftarrow [s]$, $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$, $r \leftarrow \{0,1\}^T$
- $\{\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r')}(x;r)\}$ where $r \leftarrow \{0,1\}^T$, $r' \leftarrow \{0,1\}^\ell$, and $\mathcal{O} \leftarrow \mathbf{RO}$

This implies that

$$\Pr_{i,r,\mathcal{O}}[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}(x;r) : v \text{ is accepting }] = \hat{p}$$

where $r_1,\dots,r_s$, $\mathcal{Q}$, $\mathcal{Q}_1,\dots,\mathcal{Q}_s$ are sampled according to Merlin's algorithm, $i \leftarrow [s]$, $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$, $r \leftarrow \{0,1\}^T$.
Next, we compute the probability $E(v)$ holds, namely, the probability $\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}$ makes no query in $\mathcal{Q}_j$ for $j \neq i$. From Lemma 2.1, we have that each query in $\mathcal{Q}_j$ could occur in an emulation of $\widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{\mathcal{O}}(x;r_i)}(x;r)$ with probability at most $\frac{\varepsilon}{(2s^2\cdot\ell)}$. Therefore, applying a union bound, we have that the probability $E(v)$ does not hold is at most $\frac{\varepsilon}{(2s^2\cdot\ell)} \cdot |\cup_{j\in[s]/i}\mathcal{Q}_j| < \frac{\epsilon}{2}$.
This means that the probability with which Arthur rejects in Steps 1b or 1c is at most $1-\hat{p}+\frac{\epsilon}{2}+\frac{\epsilon}{2} \leq 1-c+3\varepsilon+\epsilon = 1-c+4\epsilon$.
Next, we compute the probability with which it rejects in Step 1d. Recall that, this happens if the final estimate exceeds $c-10\varepsilon$. We will show that the real probability is at most $c-11\varepsilon$, which means the estimate fails with probability at most ε and Arthur therefore rejects with probability at most ϵ. This means the overall probability Arthur rejects in this case is at most $1-c+4\varepsilon+\varepsilon = 1-c+5\varepsilon$. Therefore, Arthur accepts with probability at least $c-5\varepsilon$ and concludes the proof of the Lemma.
It only remains to show that

$$\Pr\left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\dots,r_s,h)}(x;r) : v \text{ is accepting } \wedge E(v)\right] < c-11\varepsilon \tag{1}$$

where $r_1,\dots,r_s$, $\mathcal{Q}$, $\mathcal{Q}_1,\dots,\mathcal{Q}_s$ is sampled according to Merlin's algorithm, $r \leftarrow \{0,1\}^T$, $h \leftarrow H$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$. In fact, we will show this is at most ε which is less than $c-11\varepsilon$ as ε was chosen to be less than $c/20$.
First, we consider the event coll if in the simulation by $\widetilde{\mathcal{S}}$ for two different rewindings (α,β_i) and (α,β_j) it holds that $h(\beta_i)=h(\beta_j)$. Since $\widetilde{\mathcal{S}}$ makes at most s queries and H is a family of $2T/\varepsilon$-wise independent hash functions, we have

$$\Pr[\mathsf{coll}] < \binom{2T/\varepsilon}{2} \cdot \frac{1}{s} < (2T/\varepsilon)^2/(2s) = \varepsilon/2.$$

where the last equality follows from the fact that $s = 4T^2/\varepsilon^3$. We can now upper bound the probability in Equation 1 by

$$\Pr\left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\dots,r_s,h)}(x;r) : v \text{ is accepting } \wedge E(v)|\overline{\mathsf{coll}}\right] + \Pr[\mathsf{coll}]$$

Next, we will show that the probability of the first term in the above expression is at most $\varepsilon/2$. Then we can conclude the proof of completeness as it implies Equation 1. More formally we prove the following claim.

Claim 3.3

$$\Pr\left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\ldots,r_s,h)}(x;r) : v \textit{ is accepting } \wedge E(v)|\overline{\mathsf{coll}}\right] < \frac{\varepsilon}{2}$$

Proof: We begin by defining,

$$\Pr\left[v \leftarrow \widetilde{\mathcal{S}}^{\mathcal{O},\mathcal{V}^{*\mathcal{O}}(x,r_1,\ldots,r_s,h)}(x;r) : v \text{ is accepting } \wedge E(v)|\overline{\mathsf{coll}}\right] = \mu$$

where the probability is over $r_1, \ldots, r_s$, $\mathcal{Q}$, $\mathcal{Q}_1, \ldots, \mathcal{Q}_s$ are sampled according to Merlin's algorithm, $r \leftarrow \{0,1\}^T$, $h \leftarrow H$ and $\mathcal{O} \leftarrow \widetilde{\mathbf{O}}$.
On a high-level, we will construct a cheating unbounded prover $\mathcal{P}^*$ that makes at most polynomially many queries to the oracle and convinces an honest verifier with probability at least $\frac{\mu}{T}$ when the oracle is sampled according to $\mathbf{RO}$. Since we have a black-box reduction from a cheating prover to inverting the oracle, we have from Lemma 2.2 and Lemma 2.3 that $\frac{\mu}{T}$ must be negligible. This means that for sufficiently large n, it will be at most $\frac{\varepsilon}{2}$ and concludes the proof of the Claim.
We now proceed to describe our malicious prover $\mathcal{P}^*$. On input x, $\mathcal{P}^*$ proceeds as follows:

1. $\mathcal{P}^*$ will internally begin an emulation of $\widetilde{\mathcal{S}}$ with $\mathcal{V}^*$. Externally $\mathcal{P}^*$ interacts with the honest verifier. Both $\mathcal{P}^*$ and the external verifier are equipped with an oracle $\mathcal{O}$.
2. Upon receiving the first message α from the external verifier, $\mathcal{P}^*$ uses a PSPACE algorithm to learn all the heavy queries made by the algorithm $\widetilde{\mathcal{S}}^{\bullet,\mathcal{V}^{\bullet}(x;r')}(x;r)$ conditioned on the verifier's first message in the transcript output being α where $\mathcal{P}^*$ uses its oracle $\mathcal{O}$ to learn the responses of the heavy queries. Let $\mathcal{Q}$ be the set of queries $\mathcal{P}^*$ learns.
3. Next, using a PSPACE algorithm it samples $r_i, \mathcal{Q}_i$ for $i \in [s]$ from R_α similar to Merlin's algorithm. Namely, it samples t views for $\mathcal{V}$ from the distribution where it outputs α as its first message and oracle queries are consistent with $\mathcal{Q}$. Let r_i be the verifier's random tape and $\mathcal{Q}_i$ be the query-answer pairs made in this view. By construction, we have that $\mathcal{Q}$ is consistent with the oracle $\mathcal{O}$, however, $\mathcal{Q}_i$ might not be consistent with $\mathcal{O}$.
4. Next, $\mathcal{P}^*$ continues the emulation of $\widetilde{\mathcal{S}}$ where it feeds α as $\mathcal{V}^*$'s first message and internally emulates a random oracle $\mathcal{O}'$ which answers according to $\mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$ for the queries in this set of query-answer

pairs and according to $\mathcal{O}$ otherwise. $\mathcal{P}^*$ picks a random index j from $[s]$ to forward the external execution internally in the j^{th} rewinding session. More precisely, in the internal emulation, $\mathcal{P}^*$ follows $\mathcal{V}^*$ strategy of selecting $i = h(\beta)$ and using r_i to generate the third message in all rewindings except the j^{th} rewinding. In the j^{th} rewinding, it sends β externally to $\mathcal{V}$ and the forwards γ received from $\mathcal{V}$ internally in that rewinding. If $\widetilde{\mathcal{S}}$ concludes its simulation outputting a transcript that does not corresponds to the j^{th} rewinding, then $\mathcal{P}^*$ halts. Otherwise, $\mathcal{P}^*$ takes the fourth message δ generated in that rewinding session and forwards externally to $\mathcal{V}$.

We will now argue that the probability with which $\mathcal{P}^*$ succeeds is at least μ/T.

1. Recall that, each of $(r_i, \mathcal{Q}_i)$ were uniformly sampled from R_α. Let r' be the external verifier's random tape and $\mathcal{Q}'$ be the set of query-answer pairs made to generate α. By construction, we have that $(r', \mathcal{Q}'/\mathcal{Q})$ is an element of R_α. This means that, unless the event coll occurs (i.e. for some two rewinding sessions i and i', we have $h(\beta_i) = h(\beta_{i'})$), the distribution of $\mathcal{V}^*$'s messages emulated internally by P^* is identically distributed to
$$\{\widetilde{\mathcal{S}}^{\mathcal{O}', \mathcal{V}^{*\mathcal{O}'}(x, r_1, \ldots, r_s, h)}\}$$
where $r_1, \ldots, r_s, \mathcal{Q}, \mathcal{Q}_1, \ldots, \mathcal{Q}_s$ sampled according to Merlin's algorithm and oracle $\mathcal{O}'$ is according to the partially-fixed random oracle fixed on $\mathcal{Q} \cup \mathcal{Q}_1 \cup \cdots \cup \mathcal{Q}_s$. This means that the probability that the simulator outputs the j^{th} rewinding session as the accepting transcript is $\frac{1}{T}$.
2. Whenever $E(v)$ occurs, it means that on the accepting transcript the honest verifier will not query any $\mathcal{Q}_i$ for $i \neq j$. This means that the only queries made by the verifier will be consistent with $\mathcal{O}$.

Therefore, we have that, $\mathcal{P}^*$ succeeds in convincing the external verifier with the probability at least μ as long as its guess for the accepting session j is correct. Therefore, the overall probability $\mathcal{P}^*$ succeeds is at least $\frac{\mu}{T}$. □

□

Acknowledgements. The first author is 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, and by ISF grant 1316/18. The second author is supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, and AFOSR Award FA9550-18-1-0267, and in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-19-020700006. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. The third author is supported by Google Faculty Research Grant and NSF Award CNS-1618884. The views expressed are those of the authors and do not reflect the official policy or position of Google, the Department of Defense, the National Science Foundation, or the U.S. Government.

References

BJY97. Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In *EUROCRYPT*, pages 280–305, 1997.

BKP18. Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: A paradigm for keyless hash functions. In *STOC*, 2018.

BM88. László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

BM07. Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *FOCS*, pages 680–688, 2007.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, pages 62–73, 1993.

FGJ18. Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. On the existence of three round zero-knowledge proofs. In *EUROCRYPT*, pages 3–33, 2018.

FS89. Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.

GK96. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

GMW91. Oded Goldreich, Silvio Micali, and Avi Wigderson. 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. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

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

HV18. Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In *TCC*, pages 263–285, 2018.

IMS12. Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge pcps. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 151–168, 2012.

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61, 1989.

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

KOS18. Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In *TCC*, pages 286–313, 2018.

MP12. Mohammad Mahmoody and Rafael Pass. The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In *CRYPTO*, pages 701–718, 2012.

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

ORS15. Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In *CRYPTO*, pages 339–358, 2015.

PW09. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

RTV04. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC*, pages 1–20, 2004.