Zero-Knowledge Proofs, Revisited: The Simulation-Extraction Paradigm

Mohammad Sadeq Dousti[1] and Rasool Jalili[1,2]

[1] Network Security Center, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran.
{dousti@ce., jalili@}sharif.edu
[2] Visiting Professor, Department of Computer Science,
UCDavis, California, USA.
rjalili@ucdavis.edu

Abstract. The concept of zero-knowledge proofs has been around for about 25 years. It has been redefined over and over to suit the special security requirements of protocols and systems. Common among all definitions is the requirement of the existence of some efficient machine simulating the view of the verifier (or the transcript of the protocol), such that the simulation is indistinguishable from the reality. In this paper, we will scrutinize the philosophy behind such definition, and show that the indistinguishability requirement is stated in a *conceptually* wrong way: Present definitions allow the knowledge of the *verifier* and *distinguisher* to be independent; while, the two entities are essentially coupled. Therefore, our main take on the problem will be *conceptual* and *semantic*, rather than *literal*. We formalize the concept by introducing a "knowledge extractor" into the definition, which tries to extract the extra knowledge hard-coded into the distinguisher (if any), and then helps the simulator to construct the view of the verifier. The new paradigm is termed *Simulation-Extraction Paradigm*. We also provide an important application of the new formalization: Using the simulation-extraction paradigm, we construct one-round (i.e. two-move) zero-knowledge protocols of proving "the computational ability to invert some trapdoor permutation" in the Random-Oracle Model. It is shown that the protocol cannot be proven zero-knowledge in the classical *simulation paradigm*. The proof of the zero-knowledge property in the new paradigm is interesting in that it does not require knowing the internal structure of the trapdoor permutation, or a polynomial-time reduction from it to another (e.g. an $\mathcal{NP}$-complete) problem.

Key words: Zero-Knowledge Proof, Proof of Computational Ability, Proof of Knowledge, Trapdoor One-Way Permutation, Random Oracle, Simulation Paradigm, Simulation-Extraction Paradigm.

1 Introduction

The notion of zero-knowledge proofs was put forward by Goldwasser, Micali, and Rackoff [GMR85]. In their seminal paper, they defined *knowledge* as the computational power of a party. Such definition differentiates "knowledge" from "information." As an example (adopted from [GMR85]), assume that party A sends party B a random string r. This protocol carries information (with high *entropy*), yet it does not carry any knowledge (since party B could itself generate a random string). On the other hand, assume that party A sends party B the factorization of some number n. This protocol carries no information (as factoring a number is a deterministic process, the entropy of such message is zero), but it *might* carry knowledge (since B might not be able to compute the factorization by itself). This example conceptualizes the way that knowledge is coupled with computational power.

Zero-knowledge protocols are valuable "spin-offs" of conceptualizing knowledge in the above sense. Such protocols are usually stated in a two-party setting: Assume we have a two-party protocol $\langle V \leftrightarrow P \rangle$. The protocol is said to be zero-knowledge for P on inputs restricted to some set (language) L, if every V^* (having some pre-specified computational power, such as being a probabilistic polynomial-time Turing machine), after interacting with P on any input $x \in L$, gains no knowledge. In other words, after finishing the protocol, no such V^* should be able to *compute* things that it could not *compute* before the protocol.

In order to use the above concepts in theory and practice, we need a formal definition. As we see below, many papers tried to formalize the concepts of *knowledge* and *zero-knowledge*. While definitions for *knowledge* were usually incomparable, definitions for *zero-knowledge* often fitted in hierarchical structures. That is, some definitions were more general, while the others were stricter. A rough sketch of several definitions of zero-knowledge is illustrated in Fig. 1. We will review some of these definitions in Section 2, but the important point is that all definitions share one thing in common: They all require the existence of some probabilistic polynomial-time Turing machine, called the *simulator*, which must satisfy some condition. The condition differs in various definitions, but the spirit is always the same: The simulator M must simulate the protocol such that V^* cannot tell whether it is interacting with P or with M. This way, it is guaranteed that V^*'s "knowledge" does not increase after the protocol terminates: Since V^* cannot "distinguish" the interaction with P and M, it gains computationally nothing more than what M can compute; and since M and V^* belong to the same complexity class, V^* could compute the interaction with M all by itself.

The preceding discussion makes reference to two important concepts:

– **V^*'s Knowledge:** As said, in a zero-knowledge protocol, V^*'s knowledge must not increase after the protocol terminates. Several researchers proposed formal definitions for measuring the knowledge of a machine. Specifically, [TW87, FFS87, FFS88, FS90] tried to accomplish this task, but none of them suggested a satisfactory formalism. This was until the work of Bellare and Goldreich [BG93], which put forward a formalized definition of knowledge.[3] Unfortunately, all these works focused on formalizing the knowledge of a *prover* (P), not the *verifier* (V^*), since, it was believed that the prover is in possession of knowledge, not the verifier. As we will see, this belief is not true for some protocols. Consequently, there should be a formalization for the knowledge of the verifier. For this, we use techniques similar to those of [BG93].

– **Distinguishing[4]:** For a protocol to be zero-knowledge, it is required that V^* cannot *distinguish* the interaction with M from the interaction with P. But V^* has a prescribed program (i.e. verification), which does not necessarily involve a distinguishing procedure. In order to overcome this problem, all definitions of zero-knowledge allow V^* to use another machine $\mathcal{A}$ to satisfy its distinguishing requirement. $\mathcal{A}$ is assumed to be as computationally powerful as V^*. Therefore, if $\mathcal{A}$ can distinguish two distribution ensembles, so can V^*.

  The problem with the above approach is that, by definition, $\mathcal{A}$ and V^* can be two separate, independent, and completely unrelated entities. Thus, while V^* may not be in possession of some knowledge (see the above discussion about V^*'s knowledge), one can assume that $\mathcal{A}$ possesses that knowledge (e.g. the knowledge might be hard-coded into $\mathcal{A}$'s program). As we will see, separating $\mathcal{A}$ and V^* can lead to counterintuitive results. That is, it is possible that no simulator satisfies the zero-knowledge requirement for protocols which are *conceptually* zero-knowledge. The main result of this paper is to propose a definition of zero-knowledge which reunites $\mathcal{A}$ and V^*, so as to get rid of such conceptual problems.

In the next two subsections, we will overview a problematic case, and the solution we suggest to overcome the problem. The case is problematic since it shows *conceptual* problems in the formalization of zero-knowledge.

1.1 Overview of the Problem

While the problem is formally stated in the body of the paper, it is instrumental to informally express the problem as well. The reader is urged not go deeply into the details, since the exposition is intrinsically informal. For example, while "proofs of knowledge" are defined over relations, the informal protocol defines

[3] Yet their definitions were refined/specialized by other papers, as in [HM98, BGGL01].

[4] In this paper, we only stress *computational indistinguishability*: two distribution ensembles are said to be computationally indistinguishable if no probabilistic polynomial-time Turing machine can tell them apart. Other types of indistinguishability (perfect and statistical) are illustrated in Fig. 1.

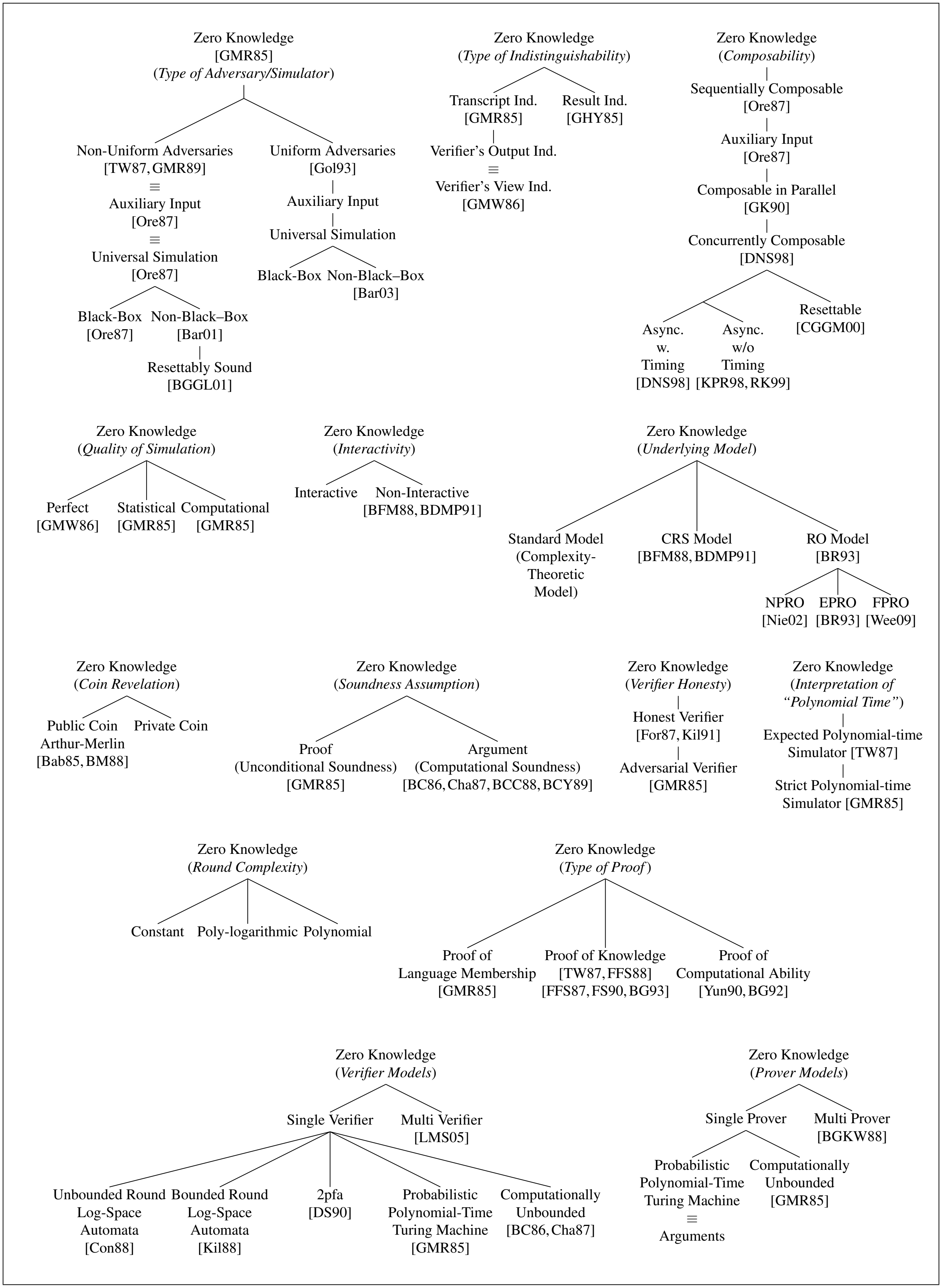

Fig. 1. Several Hierarchies of Definitions of Zero Knowledge

it over a single string. Similarly, while "computational indistinguishability" is defined over infinite distribution ensembles, the given protocol gives rise to a single probability distribution. That said, the informal exposition should be enough to familiarize the reader with the nature of the problem.

Let us begin with several informal definitions:

View of a Machine View is defined as everything a machine sees on its tapes, including the main input, the auxiliary input, the randomness, and every message sent or received by that machine.

Zero-Knowledge Protocol Consider a two-party protocol $\langle V \leftrightarrow P \rangle$, where V is probabilistic polynomial-time. The protocol is zero-knowledge[5] for P if there exists a probabilistic polynomial-time machine M, such that for every probabilistic polynomial-time machine V^*, M can construct the *view* of V^* such that the view is indistinguishable from the case where V^* is interacting with P. That is, for all probabilistic polynomial-time machine $\mathcal{A}$, the probability that $\mathcal{A}$ outputs 1 given the real view is almost equal to the probability that it outputs 1 given the simulated view. We assume that M has black-box access to V^*.

Random-Oracle Model In this model, every party participating in the protocol (including the adversary) has access to some randomly chosen function $\mathcal{O}$. For all string x, every party querying $\mathcal{O}$ at x will get the same response, regardless of the identity of that party. For instance, machines P, V^*, M, and $\mathcal{A}$ of the preceding definition have access to $\mathcal{O}$.

Now consider the following *identification* protocol: P and V have *a priori* shared an n-bit secret w (the security parameter is 1^n). We assume that w is on the auxiliary-input tape of P, but it is *hard-coded* into V's program. In addition, we assume that P has the same computational power as V (i.e. P is probabilistic polynomial-time).

P identifies itself by proving knowledge of w to V. To accomplish this, V chooses an n-bit random value r and sends it to P. P responses by sending $\mathcal{O}(w \oplus r)$ to V. Next, V verifies P's answer by querying $\mathcal{O}$ at $w \oplus r$. Finally, V informs P whether it accepted or rejected.

We first informally argue that the simulation paradigm does not allow the above protocol to be zero-knowledge. Next, we argue why the above protocol must be considered zero-knowledge.

To show that the above protocol is not zero-knowledge in the simulation paradigm, consider a (dishonest) verifier V^* which does not "know" w. Similar to V, V^* sends a random r, and receives $\mathcal{O}(w \oplus r)$. At this point, V^* aborts the protocol, and investigates the received message to gain more "knowledge."

No simulator M is able to create the view of such V^*, since M does not "know" w.[6] The old tricks, such as rewinding V^* or monitoring its queries to $\mathcal{O}$ are useless. Therefore, M's best bet is to send a random message instead of $\mathcal{O}(w \oplus r)$. However, a distinguisher $\mathcal{A}$ which "knows" w (hard-coded into its program[7]) can distinguish the two views. ■

Next, we argue that no V^* can gain knowledge in the above protocol. Let us consider two cases:

1. V^* does not know w prior to running the protocol.
2. V^* knows w prior to running the protocol.

[5] Note that this informal definition does not reference any language L.

[6] We may not assume an M into which w is hard-coded; otherwise, it will be meaningless to speak of zero-knowledge protocols. This fact cannot be justified in the informal discussion. In the formal definition (stated later), we see that the simulator is chosen universally, i.e. the order of quantifiers is such that M is determined before V^* and the inputs to $\langle V \leftrightarrow P \rangle$ are fixed, and thus M cannot depend on w.

[7] Hard-coding w into $\mathcal{A}$'s program is meaningful, since the honest verifier has this knowledge hard-coded. Therefore, $\mathcal{A}$ can be a modified version of V, suited for distinguishing tasks.

In the former case, we assume that, even if V^* is provided with a black-box (such as V) which only tells it whether some chosen w' equals w, V^* is still unable to recover w in polynomial time. Otherwise, we assume that V^* effectively knows w.

If the latter case holds, the protocol is trivially zero-knowledge; whether the simulator exists or not. After all, if we are assured that both P and V^* know w, then we can claim that no knowledge will leak from P to V^*; since P's knowledge and computational power are no more than V^*'s.

So, let us consider the former case; i.e. V^* does not know w prior to running the protocol. We claim that, knowing w is essential to distinguishing $\mathcal{O}(w \oplus r)$ from some random value. (A similar claim is formally proved in the body of the paper.) Therefore, if V^* does not know w, it cannot distinguish the real and simulated views.

However, what about a distinguisher $\mathcal{A}$ that knows w? As pointed out earlier, $\mathcal{A}$ should conceptually be regarded as the same entity as V^*. Therefore, if $\mathcal{A}$ knows something, so must do V^*. It is a problematic part of the definition of zero-knowledge which decouples $\mathcal{A}$ and V^*. We will sketch a solution in the next subsection.

1.2 Overview of the Solution

In this subsection, we informally state how the verifier V^* and the distinguisher $\mathcal{A}$ must be reunited. As pointed out in the previous subsection, $\mathcal{A}$ can only distinguish the real and simulated views of the given protocol if it "knows" w. To prove this, we use a *knowledge extractor* to extract this knowledge, as stated in the Informal Theorem 1 (a formal version of which is proven in the body of the paper):

Informal Theorem 1. For every $\mathcal{A}$, if $\mathcal{A}$ distinguishes the simulated and real views of the above protocol with *advantage*[8] q, there exists a probabilistic knowledge extractor K which, having black-box access to A, extracts w. In other words, by definition of *knowledge extraction* [BG93], $K^{\mathcal{A}}$ runs in *expected* time upper-bounded by some value proportional to $1/q$, and outputs w.

Utilizing Informal Theorem 1, we can take one further step by demanding that K simulate the real view. After all, if K succeeds in extraction of w from $\mathcal{A}$, it can easily output a view which $\mathcal{A}$ cannot distinguish from the real view. The new definition is advantageous in that it does not take into account what knowledge must be extracted from $\mathcal{A}$. In contrast, it requires K to simulate the view appropriately, regardless of the specific knowledge it extracts from $\mathcal{A}$.

Informal Definition 1. A protocol is zero-knowledge for P if there exists an efficient machine M and a probabilistic *knowledge extractor* K, such that for every efficient machine V^*, M can construct the *view* of V^*, so that for every efficient distinguisher $\mathcal{A}$, if $\mathcal{A}$ distinguishes the real and simulated views with *advantage* q, K runs in expected time (at most) proportional to $1/q$, such that $K^{V^*,\mathcal{A}}$ outputs a view which $\mathcal{A}$ cannot distinguish from the real view.

As mentioned in the previous section, $\mathcal{A}$ can only distinguish $\mathcal{O}(w \oplus r)$ from some random value if it knows w. In that case, $\mathcal{A}$ distinguishes the two views with probability almost 1, by querying $\mathcal{O}$ at $w \oplus r$.[9] Machine K can monitor $\mathcal{A}$'s queries, and learn w if $\mathcal{A}$ knows this value. Knowing w, K can easily output a view which is indistinguishable from the real view.

1.3 Road Map

In this paper, we show that the "classical" formalization of zero-knowledge—based on decoupled verifiers and distinguishers—is too strict. We revisit the definition of zero-knowledge *conceptually*, and put forward

[8] The absolute of *advantage* equals the absolute difference of two probabilities: The probability that $\mathcal{A}$ outputs 1 given the real view, and the probability that $\mathcal{A}$ outputs 1 given the simulated view.

[9] The word *almost* is used since it is possible (though with negligible probability) that the random value equals $\mathcal{O}(w \oplus r)$.

a less strict definition, replacing the *simulation paradigm* with a new paradigm termed *simulation-extraction paradigm*. The new paradigm enables us to construct interesting protocols. As an example, we construct a 1-round (i.e. 2-move) zero-knowledge proof of the computational ability to invert a trapdoor one-way permutation in the Random-Oracle Model. It is shown that the protocol is not zero-knowledge in the simulation paradigm. The elegance of the protocol is that it does work via reducing the trapdoor one-way permutation to another (e.g. an $\mathcal{NP}$-complete) problem. Therefore, any trapdoor one-way permutation will fit in the protocol. Put differently, it is not required to redesign the protocol for specific instances of trapdoor one-way permutation (e.g. RSA), or know the internal mechanisms or assumptions made by such constructs. This is in sharp contrast to previous works, such as [CEv87, CM99], where the protocols heavily depend on the nature of the construct being studied.

The rest of this paper is organized as follows: Section 2 reviews the basic notations and definitions. In Section 3, we provide a protocol and prove that it is a "proof of computational ability." This protocol is modified in Section 4 so as the resulting protocol becomes conceptually zero-knowledge. Section 5 contains the main results of this paper: It suggests a new formalization of the concept of zero-knowledge (the simulation-extraction paradigm), and proves that the protocol presented in Section 4 is zero-knowledge in this paradigm. Finally, Section 6 summarizes the paper, and suggests several open problems for further research.

2 Notations and Definitions

In this section, we briefly review some definitions. For a more thorough description of most of these concepts, please refer to [Gol01].

For all $n \in \mathbb{N}$, and all $x \in \{0,1\}^n$, define $|x| \stackrel{\text{def}}{=} n$. That is, $|x|$ denotes the length of x.

The assignment operator "$\leftarrow$" has several interpretations as follows. The specific interpretation should be clear from the context:

- Let x and y be variables. Then $x \leftarrow y$ means assigning the value of y to x.
- Let P be a (possibly random) process.[10] Then $x \leftarrow P$ means running P on its inputs (if any), and assigning the output to x.
- Let D be a probability distribution, and let $\mathrm{supp}(D)$ denote its support.[11] Then $x \leftarrow D$ means drawing x from $\mathrm{supp}(D)$ according to distribution D.

The assignment operator may be subscripted by an "R", which gives it a special meaning:

- Let S be a set. Then $x \leftarrow_R S$ means drawing an element from S uniformly and independently, and assigning it to x.

$\Pr[P_1, P_2, \ldots, P_n \colon E]$ denotes the probability of event E after ordered execution of (possibly random) processes $P_1, \ldots, P_n$. The same is true for the mathematical expectation $\mathbb{E}[P_1, P_2, \ldots, P_n \colon E]$.

Whenever we speak of the running time of a machine, we imply its running time in the length of the *security parameter* (or the *common input* to all parties). If no security parameter is available, the running time is measured in the length of the *first input* to the machine.

[10] In this paper, the terms "process" and "function" are used interchangeably. A process (function) might be either Turing-computable or Turing-uncomputable. In the former case, we may speak of algorithms or (Turing) machines instead of processes (functions).

[11] The *support* of a function f is the closure of (i.e. the smallest closed set containing) the set of arguments for which f is nonzero.

The term “efficient” is used identical to “probabilistic polynomial-time,” which may be abbreviated as *PPT*. In addition, we denote by *ITM* and *OTM* an “interactive Turing machine” and an “oracle Turing machine,” respectively. These terms can join together; thus PPT-ITM-OTM means a “probabilistic polynomial-time interactive oracle Turing machine.”

For any probabilistic algorithm $\mathcal{A}$, let $\mathcal{A}_r$ denote the algorithm $\mathcal{A}$, when its random input is r. (Thus, $\mathcal{A}_r$ is *deterministic* on its input.)

A function $\varepsilon\colon \mathbb{N} \to \mathbb{R}$ is called *negligible* in n if for every positive polynomial $p(n)$:

$$\exists n_0 \in \mathbb{N} \text{ s.t. } \forall n \geq n_0 \qquad \varepsilon(n) < \frac{1}{p(n)} \quad . \tag{1}$$

Let $\operatorname{negl}(n)$ denote the set of all functions negligible in n. Sometimes, we may abuse the notation and write $\varepsilon(n) = \operatorname{negl}(n)$ instead of $\varepsilon(n) \in \operatorname{negl}(n)$, as is the case with the Big-O notation.

Let P and V be two interactive functions [BG93], which participate in an interactive proof.[12] Define:

- **state**$\langle V,P\rangle(x)$: The state of V after interacting with P on common input x. The state is either *ACC* (for accept) or *REJ* (for reject).
- **tran**$\langle V,P\rangle(x)$: The transcript of the interaction of V and P on common input x. The transcript contains x, the interleaved sequence of messages sent by both parties during the interaction, and **state**$\langle V,P\rangle(x)$. Including the state enables us to speak of accepting and rejecting transcripts. Denote by $ACC_V(x)$ the set of transcripts of V's interaction with all possible interactive functions P, on which **state**$\langle V,P\rangle(x) = ACC$. Define $REJ_V(x)$ similarly.
- **view**$\langle V,P\rangle(x)$: The view of V after interacting with P on common input x. The view contains everything V sees on its (read-only) tapes (x, the random tape, any auxiliary input, and the messages it receives from P).

2.1 Trapdoor Permutations

Since our protocol is based on the concept of “collections of trapdoor one-way permutations,” we devote a subsection to it. One-way functions and trapdoor one-way permutations were first discussed by Diffie and Hellman [DH76]. (See also [Yao82], [GM84], and [BHSV98].)

Definition 1. (Collection of Trapdoor One-Way Permutations). *Let $X \subseteq \{0,1\}^*$ be an (efficiently samplable) infinite set, and consider a family of functions $\mathcal{F} = \{f_x\}_{x \in X}$. Assume that for all $x \in X$, f_x is a permutation over some (efficiently samplable) set $D_x \subseteq \{0,1\}^{|x|}$. The function family $\mathcal{F}$ is called a **collection of trapdoor one-way permutations**[13] if the following two conditions hold:*

1. **Easy to sample, compute, and invert using the trapdoor:** *There exist 4 efficient algorithms G, D, F and I, such that for all $n \in \mathbb{N}$:*

$$\begin{aligned}\Pr\Big[\langle x,t_x\rangle \leftarrow G(1^n)\,,\ z \leftarrow D(x) \ :\ & x \in X \cap \{0,1\}^n \\ & \wedge\ z \in D_x \\ & \wedge\ F(x,z) = f_x(z) \\ & \wedge\ I(f_x(z),t_x) = z\Big] = 1 \quad .\end{aligned} \tag{2}$$

[12] The proof can be a proof of language membership [GMR85, GMR89], a proof of knowledge [BG93], a proof of computational ability [BG92], and so on.

[13] For brevity, we may use the term “trapdoor permutation” to indicate a “collection of trapdoor one-way permutations.”

The probability is taken over the internal coin tosses of algorithms G, D, F, and I.

2. **Hard to invert without the trapdoor:** *For any PPT adversary $\mathcal{A}$, the following condition should hold:*

$$\forall c \in \mathbb{N},\ \exists n_0 \in \mathbb{N}\ \text{ s.t. } \forall n \geq n_0$$

$$\Pr\Big[\langle x, t_x\rangle \leftarrow G(1^n),\ z \leftarrow D(x)\ :\ \mathcal{A}(f_x(z)) = z\Big] < n^{-c}\ . \quad (3)$$

The probability is taken over the internal coin tosses of algorithms G and D, as well those of the adversary $\mathcal{A}$. □

Remark 1. Algorithms G, D, F, and I are called "Generator," "Domain Sampler," "Function Evaluator," and "Inverter," respectively.

Remark 2. If (2) holds, we can change the definition so as to allow algorithms F and I to be deterministic.

Remark 3. Definition 1 can be relaxed by allowing algorithms G, D, F, and I to fail with probability $\text{negl}(n)$.

2.2 Proof of Knowledge and Proof of Computational Ability

Proofs of knowledge and proofs of computational ability were fully formalized by Bellare and Goldreich [BG93, BG92]. The proofs are stated in terms of binary relations.

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. Define $R(x) \overset{\text{def}}{=} \{y | \langle x, y\rangle \in R\}$, and $L_R \equiv \text{dom}(R) \overset{\text{def}}{=} \{x | R(x) \neq \varnothing\}$. Definition 2 formalizes "proofs of knowledge."

Definition 2. (Proof of Knowledge [BG93]). *Let R be a binary relation, and let $\kappa\colon \{0,1\}^* \to [0,1]$. Let V be an efficiently computable interactive function. We call V a* ***knowledge verifier for** R **with error** κ if the following conditions hold:*

1. ***Non-triviality:*** *There exists an interactive function P^* such that V always accepts after interacting with P^* on all $x \in L_R$. In other words,*

$$\Pr\left[\mathbf{state}\langle V, P^*\rangle(x) = ACC\right] = 1\ . \quad (4)$$

2. ***Validity:*** *For some constant $c > 0$, there exists a probabilistic oracle machine K, so that for all interactive function P and every $x \in L_R$:*

   If $p(x) \overset{\text{def}}{=} \Pr\left[\mathbf{state}\langle V, P\rangle(x) = ACC\right] > \kappa(x)$, then $K^P(x)$ outputs an element of $R(x)$ in expected number of steps bounded by $\frac{|x|^c}{p(x) - \kappa(x)}$.

   We call K a ***universal knowledge extractor*** *for R.* □

A similar notion is the concept of "proofs of computational ability." Definitions 3 and 4 are devoted to this concept.

Definition 3. (Solving a Relation [BG92]). *Let R be a binary relation, and D be a distribution on* $\text{dom}(R)$. *Assume that $t \in \mathbb{N}$ and suppose that $M(\cdot)$ is a machine.*

- *We say that* ***machine** $M(\cdot)$ **solves** R **under** D **in expected** t **steps** if:*

$$\exists i, 1 \leq i \leq t\ \text{ s.t. }\ (x_i, y) \leftarrow M(\langle x_1, \ldots, x_t\rangle \leftarrow D^t) \text{ and } y \in R(x_i)\ . \quad (5)$$

  and M halts in expected t steps.

- *We say that machine $M(\cdot)$ **strongly** solves R under D in expected t steps if $M(x \leftarrow D) \in R(x)$, and M halts in expected t steps.*

In both cases, the expectation is taken over the random selection of the input, as well as the coin tosses of M. □

Let $\mathcal{R} = \{R_x\}_{x\in\{0,1\}^*}$ be a *family* of binary relations, where $R_x \subseteq \{0,1\}^{|x|} \times \{0,1\}^*$. $\mathcal{D} = \{D_x\}_{x\in\{0,1\}^*}$ is an input distribution for $\mathcal{R}$ if D_x is a distribution on $\mathrm{dom}(R_x)$ for all x.

Definition 4. (Proof of Computational Ability [BG92]). *Let $\mathcal{R}$ and $\mathcal{D}$ be as defined above, and let $\kappa\colon \{0,1\}^* \to [0,1]$. Let V be an efficiently computable interactive function. We call V a **verifier of the ability to (strongly) solve $\mathcal{R}$ under $\mathcal{D}$ with error κ** if the following conditions hold:*

1. ***Non-triviality:** There exists an interactive function P^* such that V always accepts after interacting with P^* on all $x \in L_{\mathcal{R}}$. In other words,*

$$\Pr\left[\mathbf{state}\langle V^{D_x}, P^*\rangle(x) = ACC\right] = 1 \quad . \tag{6}$$

2. ***Validity:** For some constant $c > 0$, there exists a probabilistic oracle machine $K(\cdot,\cdot,\cdot)$, so that for all interactive function P, every $x \in L_{\mathcal{R}}$, and all $\gamma \in ACC_V(x)$, machine $K^P(x,\gamma,\cdot)$ satisfies:*

   If $p_x \overset{\text{def}}{=} \Pr\left[\mathbf{state}\langle V^{D_x}, P\rangle(x) = ACC\right] > \kappa(x)$, then $K^P(x,\gamma,\cdot)$ (strongly) solves R_x under D_x in expected number of steps bounded by $\frac{|x|^c}{p_x - \kappa(x)}$.

   *We call K a **(strongly) ability extractor** for $\mathcal{R}$ under $\mathcal{D}$.* □

The verifier might be able to draw elements from $\mathcal{D}$ independently. In such cases, one can omit $\mathcal{D}$ from Definition 4, and speak of *distribution-free verifiers of ability*.

2.3 Definitions of Zero Knowledge

It is a desired property for security protocols to be *zero knowledge* (*ZK*, for short). Informally, a (two-party) protocol $\langle V \leftrightarrow P\rangle$ is considered ZK for party P if by the end of protocol, V cannot compute something that it could not compute before the protocol began.

A more practical definition requires the existence of an efficient program, called the *simulator*, which approximates the execution of protocol. Based on the "accuracy" of the approximation, we have *Perfect ZK* (PZK), *Statistical ZK* (SZK), and *Computational ZK* (CZK). In this paper, we only deal with CZK. As such, other concepts are not defined here.

> Zero-knowledge is a term applied normally to interactive proofs, although it can be used independently. To emphasize the distinction, we use the term "interactive protocols" instead of "interactive proofs." A *ZK interactive proof* is an interactive protocol which is also ZK.

Definition 5 (CZK [GMR85]). *Let $\langle V \leftrightarrow P\rangle$ be interactive protocol. We say that this protocol is* **CZK** *for P on $L = L_R$, if for every PPT-ITM V^*, there exists a PPT machine M_{V^*} such that for every PPT algorithm $\mathcal{A}$ (the distinguisher):*

$$\forall c > 0,\ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0,\ \forall x \in L \cap \{0,1\}^n$$
$$|\Pr[\mathcal{A}(M_{V^*}(x)) = 1] - \Pr[\mathcal{A}(\mathbf{view}\langle V^*, P\rangle(x)) = 1]| < n^{-c} \quad . \tag{7}$$

where the first probability is taken over the internal coin tosses of M_{V^*} and $\mathcal{A}$, and the second probability is taken over the internal coin tosses of V^*, P, and $\mathcal{A}$. □

Definition 5 has a major drawback: It does not consider any possible *auxiliary input* in possession of *dishonest* verifiers. Several works (e.g. [Ore87, TW87, GMR89, GO94]) tried to overcome this drawback, resulting in the definition of *auxiliary-input ZK*. Another definition, called *universal-simulation ZK*, results from swapping the "universal quantifier over V^*" and the "existential quantifier over M_{V^*}" in Definition 5. Such definition guarantees a universal simulator for every verifier. Oren [Ore87] claimed that the class of languages possessing auxiliary-input ZK is equivalent to the class of languages possessing universal-simulation ZK.[14] We do not formalize these concepts here, since we make use of an even more strict definition of ZK: *Black-box ZK*.

Definition 6 (Black-Box CZK[15]). *Let $\mathcal{V}_{x,r,\xi}(\overline{m})$ be a function that on input $\overline{m}$, a sequence of messages sent by P, returns the* `next-message` *of V on common input x, auxiliary input ξ, and random input r. In addition, let $V_r(\xi)$ denote machine V with auxiliary input ξ and random input r.*

We call an interactive protocol $\langle V \leftrightarrow P\rangle$ **black-box CZK** *for P on $L = L_R$, if there exists a PPT-OTM M such that for every PPT-ITM V^*, and all PPT-OTM $\mathcal{A}$ (the distinguisher):*

$$\begin{aligned}&\forall c > 0,\ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0,\ \forall x \in L \cap \{0,1\}^n,\ \forall r, \xi \in \{0,1\}^* \\ &\qquad \left|\Pr\left[\mathcal{A}^{\mathcal{V}^*_{(x,r,\xi)}}\left(M^{\mathcal{V}^*_{(x,r,\xi)}}(x,r,\xi)\right) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{V}^*_{(x,r,\xi)}}\left(\mathbf{view}\langle V^*_r(\xi), P\rangle(x)\right) = 1\right]\right| < n^{-c} \ .\end{aligned} \tag{8}$$

The first probability is taken over the internal coin tosses of M and $\mathcal{A}$, and the second probability is taken over the internal coin tosses of P and $\mathcal{A}$. (Note that V^* is a deterministic machine with hard-coded randomness r.) □

It is obvious from the definition that, black-box CZK is both auxiliary-input CZK and universal-simulation CZK.

We will use another variant of CZK, which we prefer to call "passive-observer CZK." In this variant, the simulator should approximate the *transcript* of the protocol, rather than the *view* of (possibly dishonest) verifier. In this case, the protocol is CZK for any passive observer eavesdropping the protocol messages. Definition 7 provides a somehow weaker form of CZK than Definition 5. We will exploit this weaker form when trying to formulate a revised definition of ZK.

Definition 7 (Passive-Observer CZK [GMR85]). *We call an interactive protocol $\langle V \leftrightarrow P\rangle$* **passive-observer CZK** *for P on $L = L_R$, if for every PPT-ITM V^*, there exists a PPT machine M_{V^*} such that for every PPT algorithm $\mathcal{A}$ (the distinguisher):*

$$\begin{aligned}&\forall c > 0,\ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0,\ \forall x \in L \cap \{0,1\}^n \\ &\qquad\qquad \left|\Pr\left[\mathcal{A}\left(M_{V^*}(x)\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathbf{tran}\langle V^*, P\rangle(x)\right) = 1\right]\right| < n^{-c} \ .\end{aligned} \tag{9}$$

where the first probability is taken over the internal coin tosses of M_{V^*} and $\mathcal{A}$, and the second probability is taken over the internal coin tosses of V^*, P and $\mathcal{A}$. □

Another definition considers the ZK property only with respect to the *honest verifier*; i.e. the verifier which sends messages according to the prescribed program (but may store the received messages for further investigations). Definition 8 formalizes this concept.

[14] The claim is with respect to *non-uniform* adversaries. It is conjectured that, with respect to *uniform* adversaries, the class of languages possessing universal simulation ZK is a proper subclass of the class of languages possessing auxiliary-input ZK. (See for example [Bar03].)

[15] The definition first appeared in [Ore87, GO94], but we adopted it from [Gol01].

Common Input (Security Parameter): 1^n.

Verifier (V)		Prover (P)
Let $\langle x, t_x \rangle \leftarrow G(1^n)$, $z \leftarrow D(x)$, $y \leftarrow F(x,z)$.		
	$\xrightarrow{\langle x,y \rangle}$	
		Compute $w \leftarrow f_x^{-1}(y)$.
	$\xleftarrow{w}$	

IF $w = z$ THEN
Set state to ACC.
ELSE
Set state to REJ.		

Protocol 1. Base Protocol (Variant 1).

Definition 8 (Honest-Verifier CZK (HVCZK)). *We call an interactive protocol $\langle V \leftrightarrow P \rangle$* **HVCZK** *for P on $L = L_R$, if for the honest verifier V, there exists a PPT machine M such that for every PPT algorithm $\mathcal{A}$ (the distinguisher):*

$$\forall c > 0,\ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0,\ \forall x \in L \cap \{0,1\}^n$$
$$\left|\Pr\left[\mathcal{A}\left(M(x)\right) = 1\right] - \Pr\left[\mathcal{A}\left(\mathbf{view}\langle V, P \rangle(x)\right) = 1\right]\right| < n^{-c} \quad . \qquad (10)$$

where the first probability is taken over the internal coin tosses of M and $\mathcal{A}$, and the second probability is taken over the internal coin tosses of V, P and $\mathcal{A}$. □

3 The Base Protocol

Let $\mathcal{F} \overset{\text{def}}{=} \{f_x\}_{x \in X}$ be a collection of trapdoor one-way permutations, and let X and D_x be its index set and domain set, and G, D, F and I be the corresponding efficient algorithms, as per Definition 1.

Define the binary relation $R_x \overset{\text{def}}{=} \{\langle f_x(z), z \rangle\}_{z \in D_x}$ for all $x \in X$, and let $\mathcal{R}$ denote a family of binary relations: $\mathcal{R} \overset{\text{def}}{=} \{R_x\}_{x \in X}$. Note that $\mathcal{D} \overset{\text{def}}{=} \{F(x, D(x))\}_{x \in X}$ is an input distribution for $\mathcal{R}$.

Proving membership in $\mathcal{R}$ is easy (given x and a pair $\langle y, z \rangle$) due to the "easy-to-compute" property of $\mathcal{F}$. In contrast, solving $\mathcal{R}$ under $\mathcal{D}$ (see Definition 3) is hard, since $\mathcal{F}$ is "hard to invert." Therefore, we intuitively believe that, if a prover can solve $\mathcal{R}$ under $\mathcal{D}$, it has some computational advantage over an efficient verifier, and such interaction constitutes a "proof of computational ability" (see Definition 4).

Protocol 1 is a formal presentation of the interaction described above. It is called the "Base Protocol" since it forms the basis for proofs and protocols presented in this paper. To simplify the exposition, we assume that algorithms G, D, F and I are incorporated into the code of the honest prover and the honest verifier. In the latter case, this assumption is justified by the fact that these algorithms are efficiently computable.

Another variant[16] of Protocol 1, presented in [BG92], assumes that $x \in X$ is chosen *a priori* to the start of the protocol, and given to both parties as the common input. In this variant, P has to show the ability to invert f_x for some fixed x. While the two variants are quite different, the choice of which is immaterial to the objective of this paper. Theorem 1 formalizes what we expect from the Base Protocol.

[16] Hereafter referred to as Variant 2.

```
1:  ESTIMATE $p_x$.
2:  FOR $i \leftarrow 1$ TO $n/p_x$ DO
3:      LET $y \leftarrow \mathcal{V}(x)$.
4:      LET $w \leftarrow \mathcal{P}(x,y)$.
5:      IF $y = F(x,w)$ THEN
6:          OUTPUT $\langle w,y \rangle$.
7:          HALT.
8:  PERFORM "Exhaustive Search" on $y$.
```

Program 1. The Ability Extractor

Theorem 1. *Both variants of Protocol 1 constitute proofs of computational ability. In particular:*

> **Claim 1.1** *In Variant 1 of the Base Protocol, V is an ability verifier for solving R_x under $F(x,D(x))$ (with zero error), where x is chosen according to the distribution induced by the (first component of) $G(1^n)$ output over $X \cap \{0,1\}^n$.*
>
> **Claim 1.2** *In Variant 2 of the Base Protocol, V is an ability verifier for solving R_x under $F(x,D(x))$ (with zero error), where $x \in X$ is chosen* a priori.

A proof for Claim 1.2 is given in [BG92]. However, their proof suffers from four drawbacks:

1. It is very complicated;
2. The proof is only sketched, and is not thoroughly discussed;
3. The proof is for the special case where $D(x)$ is uniform over $\{0,1\}^{|x|}$;
4. There is a minor error in the proof.[17]

The following proof overcomes these shortcomings. In addition, it is general enough to cover both Claims 1.1 and 1.2 (but we only present it for Claim 1.2, since it is simpler). Moreover, the proof ideas are used to prove the main result of this paper (Theorem 8).

Proof. The "non-triviality" property of Definition 4 follows directly from the existence of the "Function Evaluator" F for the family $\mathcal{F}$. We therefore focus on the validity property.

Let $p_x \overset{\text{def}}{=} \Pr[\mathbf{state}\langle V,P\rangle(x) = ACC]$, where the probability is taken over the random coin tosses of both P and V. The ability extractor can accurately estimate p_x in time $\text{poly}(n)/p_x$. A good survey of estimating the average of some function $g\colon \{0,1\}^n \to [0,1]$ is presented in [Gol97].

Let $\mathcal{V}(\cdot)$ and $\mathcal{P}(\cdot,\cdot)$ denote the `next-message` function of V and P, respectively. Note that to evaluate the `next-message` function, having oracle access to the corresponding party suffices. We know that $\mathcal{V}(x) = F(x,D(x))$ for all input x's (assuming $x \in \text{dom}(\mathcal{R})$), since V is supposed to be honest.

We are now ready to provide the ability extractor K. The program of K is given in Program 1. We assume that V uses fresh randomness every time execution reaches Step 3. Therefore, a new challenge—drawn according to $F(x,D(x))$—is applied to each round. In addition, we assume that the extractor *resets* P in Step 4, and provides it with fresh randomness. This way, P cannot keep a history of previous challenges, and thus it returns an *independent* response.

We now prove that Program 1 satisfies the validity property. Since the probability that P can invert y chosen according to $\mathcal{V}(x)$ is p_x, each iteration halts with expected probability p_x. Therefore, the probability that Step

[17] Specifically, inequality (1) on page 6 of [BG92] states that $\frac{|Y_x(i)|}{2^{|x|}} > \frac{p_x \cdot 2^i}{n}$. We confirmed that this is wrong, and found several counterexamples to it. The correct version is: $\frac{|Y_x(i)|}{2^{|x|}} \geq \frac{p_x \cdot 2^{i-1}}{n}$.

8 runs (i.e. the extractor is unsuccessful in all n/p_x iterations) is $(1-p_x)^{(n/p_x)} < e^{-n}$, which is negligible in n.[18]

Thus, the ability extractor is able to invert at least one of y's provided to it by the verifier with overwhelmingly high probability. The negligible probability can be eliminated by running an exhaustive search in Step 8.

The expected running time of Program 1 can be decomposed into the following:

- **Step 1:** Runs in time $\mathrm{poly}(n)/p_x$;
- **Steps 2–7:** Runs in time $(n/p_x)\mathrm{poly}(n)$;
- **Step 8:** Runs in time $O(2^n)\mathrm{poly}(n)$ with expected probability at most e^{-n}.

Consequently, the expected running time of Program 1 is $\mathrm{poly}(n)/p_x$, as required by Definition 4. ■

It is intuitively obvious that the Base Protocol is a "knowledge carrying" proof. In particular, the prover can be used as an "inversion oracle"—That is, a dishonest verifier can exploit the prover to invert any string y under $f_x(\cdot)$, whose inverse is not "known" to it.

Before trying to give a ZK variant of the Base Protocol, we take note of the following negative result:

Theorem 2 (Goldreich and Oren [Ore87, GO94]). *Let L be a language for which there exists a* ***two-step*** auxiliary-input *ZK proof of membership with negligible error. Then $L \in \mathcal{BPP}$.*[19]

Note that Theorem 2 holds for proofs of language membership, while the Base Protocol is a proof of computational ability. We do not know whether it holds for such proofs. Thus, we state the modified version as a conjecture.

Conjecture 1. Theorem 2 holds for proofs of knowledge (respectively proofs of computational ability) as well. That is, Let $\langle V \leftrightarrow P\rangle$ be a **two-step** *auxiliary-input* ZK proof of knowledge (respectively proof of computational ability) with negligible error on some language $L = L_R$. Then, R is trivial, i.e. it can be (strongly) solved by a $\mathcal{BPP}$ algorithm.

Proving or refuting Conjecture 1 is of independent interest. If it is proven, a two-step ZK version of the Base Protocol is out of question (in the auxiliary-input sense). However, even if Conjecture 1 is refuted, it may be impossible to give a two-step ZK version of the Base Protocol.

Anyway, we could not conceive of the ZK version of our two-step protocol in the *Standard Model*. Thus, we resorted to a more powerful model of computation, discussed in the next section.

4 Exploiting the Random Oracle Model

Despite what Conjecture 1 states, it might be possible to give a two-step ZK version of the Base Protocol in a *different* computational model than the Standard Model. A similar approach has been adopted by Aiello and Håstad [AH91]. In particular, they proved that *relative* to some oracle A, there exists a two-step passive-observer (Definition 7) PZK proof for some language $L \notin \mathcal{BPP}^A$.

While at first it may seem that we are proving a similar proposition, our approach is unique in that we perform relativization with respect to a "random oracle" [BR93].[20] In addition, [AH91] assumes a very loose definition of ZK (in which the adversary is merely a *passive observer*, and may not interfere with the execution of the protocol). Thus, our proposition is stronger in that the adversary (which tries to gain knowledge) is the malicious verifier itself, and can send messages, in addition to observing them.

[18] The inequality is due to the fact that $g(\alpha) \overset{\text{def}}{=} (1-\alpha)^{\frac{1}{\alpha}}$ is a decreasing function of α. Hence $g(\alpha) < \lim_{t\to 0} g(t) = e^{-1}$ for all $\alpha \in (0,1]$.

[19] Goldreich and Krawczyk [GK90, GK96] proved a similar result for three-step *black-box* ZK proofs of membership.

[20] It might be helpful to draw an analogy: Our work is to [AH91], as [BG81] is to [BGS75]. However, this analogy is too informal to be taken serious. It is solely given to provide the reader with some intuition. In addition, note that the main objective of this paper is to revise the concept of ZK, not to provide some specific protocol.

4.1 The Random Oracle Model

Intuitively, the *Random Oracle Model* (*RO Model*, for short) is the Standard Model augmented with a "random function" $\mathcal{O}$. Every party (including the adversary) can query $\mathcal{O}$ at the point of its choice, and receive the answer regardless of its identity.

There are several definitions of the random oracle, based on its length of the input or output. We adopted a definition in which the length of input is arbitrary, while the length of output is bounded by some predetermined function:

Definition 9. *A* random oracle *is a function $\mathcal{O}$, chosen uniformly from the collection of all functions whose length of output is $\ell(n)$, where $\ell\colon \mathbb{N} \to \mathbb{N}$ is a function, and n is the length of the security parameter of the system (e.g. the length of the common input). Symbolically, let $\mathbb{F}_\ell$ be the collection mentioned above: $\mathbb{F}_\ell \overset{\text{def}}{=} \{f\colon \{0,1\}^* \to \{0,1\}^{\ell(n)}\}$. Then $\mathcal{O} \leftarrow_R \mathbb{F}_\ell$.*
We assume that $\ell(\cdot)$ is a polynomial. □

There are at least three variants of ZK in the RO Model. The definitions differ in what the *simulator* can perform in order to simulate the view (resp. transcript) of the verifier (resp. the protocol), and whether the distinguisher can access $\mathcal{O}$. We describe the three variants briefly, and justify the variant we adopt for the rest of this paper.

In the original definition of ZK in the RO Model (due to Bellare and Rogaway [BR93]), the "RO-Model Simulator" has two advantages over a "Standard-Model Simulator":

1. It can see the queries made by parties to $\mathcal{O}$, and
2. It can choose the answer to every query.

Nielsen [Nie02] coined the term "programmability" for the second property, and introduced a new model in which the RO-Model Simulator was not able to program the random oracle. Such model was called the *Non-Programmable Random Oracle* (NPRO) model. One year later, Pass [Pas03] showed that a remarkable feature of the ZK in the Standard Model, namely *deniability*, will be lost if we allow the simulator to program the random oracle. Deniability is informally defined as the ability of the prover to deny its *interaction* with the verifier.

Later, in 2009, Wee [Wee09] coined the term *Explicitly Programmable Random Oracle* (EPRO) model for the original definition of Bellare and Rogaway [BR93], and defined another model he termed the *Fully Programmable Random Oracle* (FPRO) model. In FPRO model, the distinguisher may not query $\mathcal{O}$ at all.

Note that the NPRO model is more restrictive than the EPRO model, which in turn is more restrictive than the FPRO model. That is, if a protocol is ZK in the NPRO model, it is ZK in both the EPRO and FPRO models. For this reason, and because we believe that deniability is a natural expectation from any ZK protocol, we adopted the NPRO model for the rest of this paper. The interested reader may consult [BR93, Nie02, Pas03, Wee09] for a more thorough review of other models.

We now define what it means for a protocol to be auxiliary-input ZK in the NPRO Model.[21]

Definition 10 (Auxiliary-Input CZK in the NPRO Model). *Let $\langle V \leftrightarrow P\rangle$ be an interactive protocol. We say that this protocol is* **auxiliary-input CZK in the NPRO Model** *for P on $L = L_R$, if for all PPT-ITM-OTM*

[21] Unruh [Unr07] put forward a new definition for auxiliary-input in the RO model, in which the auxiliary-input is dependent on the choice of $\mathcal{O}$. Later, Wee [Wee09] proved that the ZK property is not closed under sequential composition, unless the protocol at hand is RO-dependent auxiliary-input ZK. However, as we will point out at the end of Section 5, Wee's proof invalidates under our revised definition of ZK. Thus, the validity of his negative result remains open (under the new definition of ZK).

V^*, there exists a PPT-OTM M_{V^*}, such that for every PPT-OTM $\mathcal{A}$ (the distinguisher):

$$\forall c > 0,\ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0,\ \forall x \in L \cap \{0,1\}^n,\ \forall r, \xi \in \{0,1\}^*$$
$$\Big| \mathbb{E}\Big[\mathcal{O} \leftarrow_R \mathbb{F}_\ell \colon \Pr\Big[\mathcal{A}^{\mathcal{O}}\Big(\mathbf{view}\langle V_r^{*\mathcal{O}}(\xi), P^{\mathcal{O}}\rangle(x)\Big) = 1\Big] - \Pr\big[\mathcal{A}^{\mathcal{O}}\big(M_{V^*}^{\mathcal{O}}(x,r,\xi) = 1\big]\Big]\Big| < n^{-c} \ . \tag{11}$$

The expectation is taken over the random selection of $\mathcal{O}$, the first probability is taken over the internal coin tosses of P and $\mathcal{A}$ (note that V^* is a deterministic machine with hard-coded randomness r), and the second probability is taken over the internal coin tosses of M_{V^*} and $\mathcal{A}$. □

4.2 A ZK Implementation of the Base Protocol

We are now prepared to give a ZK implementation of the Base Protocol. We adopted Variant 1 for some technical reasons.

Let $\mathcal{F} = \{f_x\}_{x \in X}$ be a collection of trapdoor one-way permutations, and $\mathcal{O}$ be a random oracle. Define the binary relation $\hat{R}_x \stackrel{\text{def}}{=} \{\langle f_x(z), \mathcal{O}(z)\rangle\}_{z \in D_x}$ for all $x \in X$, and let $\hat{\mathcal{R}}$ denote a family of binary relations: $\hat{\mathcal{R}} \stackrel{\text{def}}{=} \{\hat{R}_x\}_{x \in X}$. Note that $\mathcal{D} \stackrel{\text{def}}{=} \{F(x, D(x))\}_{x \in X}$ is an input distribution for $\hat{\mathcal{R}}$.

Protocol 2 is an interactive proof of computational ability—That is, one can prove that V is an ability verifier for solving $\hat{R}_x$ under $F(x, D(x))$ (with zero error). The proof is similar to that of Theorem 1.

Remark 4. In fact, there is one clever difference: While membership in relation R_x (defined in Section 3) can be decided efficiently, this is not the case with relation $\hat{R}_x$ (see Theorem 3, Part (ii)). Hence, if we try to give an ability extractor similar to Program 1, the extractor will probably get stuck at Step 5. A simple workaround is to change Definition 4, so as to let the ability extractor have oracle access to the final state of V.

Definition 11. *Let χ_R denote the* **characteristic function** *of the k-ary relation R. That is, for every $\langle a_1, \ldots, a_k\rangle \in (\{0,1\}^*)^k$:*

$$\chi_R(\langle a_1, \ldots, a_k\rangle) = \begin{cases} 1 & \text{if } \langle a_1, \ldots, a_k\rangle \in R, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Relation R is **hard to decide** *if for every efficient algorithm $\mathcal{A}$, there exists an infinite sequence $\{s_i\}_{i \in \mathbb{N}}$, where for all $i \in \mathbb{N}$, we have $s_i \in (\{0,1\}^*)^k$ and $\Pr[\mathcal{A}(s_i) = \chi_R(s_i)] < \frac{1}{2} + \mathrm{negl}(|s_i|)$. The probability is taken over the internal coin tosses of $\mathcal{A}$.*

Relation R is **hard to approximate** *if there exists an efficient algorithm S, such that for every efficient algorithm $\mathcal{A}$:*

$$\forall c > 0,\ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0$$
$$\Pr[\langle a_1, \ldots, a_k\rangle \leftarrow S(1^n) \colon \mathcal{A}(\langle a_1, \ldots, a_k\rangle) = \chi_R(\langle a_1, \ldots, a_k\rangle)] < \frac{1}{2} + n^{-c} \ . \tag{13}$$

where the probability is taken over the internal coin tosses of S and $\mathcal{A}$.

In the RO Model, algorithms S and $\mathcal{A}$ have oracle access to $\mathcal{O}$. In addition, the corresponding probabilities are taken over the random selection of $\mathcal{O}$ as well. □

Common Input (Security Parameter): 1^n.

Verifier (V)		Prover (P)
Let $\langle x, t_x \rangle \leftarrow G(1^n)$, $z \leftarrow D(x)$, $y \leftarrow F(x, z)$.		
	$\xrightarrow{\langle x, y \rangle}$	
		Compute $w \leftarrow \mathcal{O}\left(f_x^{-1}(y)\right)$.
	$\xleftarrow{w}$	

IF $w = \mathcal{O}(z)$ THEN
Set state to ACC.
ELSE
Set state to REJ.		

Protocol 2. Alleged ZK Implementation of the Base Protocol.

```
LET $\langle x, t_x \rangle \leftarrow G(1^n)$.
LET $\mathcal{O} \leftarrow_R \mathbb{F}_\ell$.
LET $z \leftarrow D(x)$.
LET $y \leftarrow F(x, z)$.
LET $\rho \leftarrow_R \{0, 1\}$.
IF $\rho = 0$ THEN
    LET $r \leftarrow \mathcal{O}(z)$.
ELSE
    LET $r \leftarrow_R \{0, 1\}^{\ell(|x|)}$.
OUTPUT $(x, \langle y, r \rangle)$.
```

Program 2. The algorithm used to prove that $\hat{\mathcal{R}}$ is hard to approximate.

We are now ready to state and prove Theorem 3.

Theorem 3. *Relation $\hat{\mathcal{R}}$ (defined at the beginning of this section) is*:

(i) *hard to approximate; and*
(ii) *hard to decide.*

Proof. **Part (i)—** Let S be the efficient algorithm whose code is described in Program 2. Note that on input 1^n, S outputs a pair $\langle y, r \rangle$, as well as a string x. The latter provides extra information about the former. Outputting such extra information strengthens Theorem 3, since we will indeed prove that no efficient adversary can approximate or decide $\hat{\mathcal{R}}$, even if it is provided with such extra information.

Let $\mathcal{A}$ be any efficient program, and define $d \leftarrow \mathcal{A}(S(1^n)) = \mathcal{A}(x, \langle y, r \rangle)$, characterizing $\mathcal{A}$'s verdict as to whether or not the output of S belongs to $\hat{\mathcal{R}}$. Let E_1 denote the event that $\mathcal{O}$ is queried at point $z = f_x^{-1}(y)$, and E_2 denote the event that $d = \rho$. The proof is based on lemmas 1 and 2.[22]

[22] The idea is based on the work of Bellare *et al.* [BHSV98].

```
1:  LET $\Psi \leftarrow \varnothing$.
2:  FOR $i \leftarrow 1$ TO $t_{\mathcal{A}}(|x|)$ DO
3:      SINGLE-STEP $\mathcal{A}(x, \langle y, r \rangle)$.
4:      IF $\mathcal{A}$ halts THEN
5:          HALT.
6:      ELSE IF $\mathcal{A}$ queries $\mathcal{O}$ THEN
7:          LET $q$ be the query.
8:          IF $y = F(x, q)$ THEN
9:              OUTPUT $q$.
10:             HALT.
11:         ELSE IF $q \in \mathrm{dom}(\Psi)$ THEN
12:             ANSWER query with $\Psi(q)$.
13:         ELSE
14:             LET $\psi \leftarrow_R \{0,1\}^{\ell(|x|)}$.
15:             LET $\Psi \leftarrow \Psi \cup \{(q, \psi)\}$.
16:             ANSWER query with $\psi$.
```

Program 3. Algorithm of $\mathcal{A}'$.

Lemma 1. *For large enough x's,* $\Pr[E_1] = \mathrm{negl}(|x|)$.

Proof. Assume to the contrary that, there exists some $c \in \mathbb{N}$, such that the probability that $\mathcal{O}$ is queried at point z is greater than $|x|^{-c}$ for infinitely many x's. We show that, there exists an efficient algorithm $\mathcal{A}'$, which uses $\mathcal{A}$ as a subroutine, and inverts infinitely many y's with non-negligible probability.

Let $t_{\mathcal{A}}(\cdot)$ be a polynomial upper-bounding the running time of $\mathcal{A}$. Program 3 describes the algorithm of $\mathcal{A}'$ on input $(x, \langle y, r \rangle)$ generated by Program 2.

Let us describe Program 3. Function Ψ, defined in Step 1, is to simulate the random oracle $\mathcal{O}$. Queries made by $\mathcal{A}$ should be answered consistently; that is, if $\mathcal{A}$ has asked them before, it should get the same answer. This consistency measure is enforced in Steps 11–16.

Remark 5. Instead of function Ψ, we could have used *pseudorandom function generators* [GGM86] for more efficiency (in terms of *storage*). However, that would make the analysis more complex, which is undesirable.

In Step 3, `SINGLE-STEP` $\mathcal{A}$ means "to proceed one step in the computation of $\mathcal{A}$." If the next computation step of $\mathcal{A}$ is `HALT`, algorithm $\mathcal{A}'$ halts as well (Steps 4–5), while if it is a query to $\mathcal{O}$, algorithm $\mathcal{A}'$ first checks whether it is $f_x^{-1}(y)$ (Step 8), and if so, outputs it and halts. Otherwise, $\mathcal{A}'$ tries to simulate the behavior of a random oracle as described above.

Obviously, algorithm $\mathcal{A}'$ outputs $f_x^{-1}(y)$ with the same probability that $\mathcal{A}$ queries $\mathcal{O}$ at point $f_x^{-1}(y)$ (on average). Since we assumed that the latter probability was non-negligible for infinitely many x's, so would be the former probability. But this contradicts the non-invertibility of $\mathcal{F}$ by efficient machines. ■

Lemma 2. $\Pr[E_2 \mid E_1{}'] = \frac{1}{2}$.

Proof. Assuming $E_1{}'$ (that is, $\mathcal{O}$ is *not* queried at point $z = f_x^{-1}(y)$), from the viewpoint of $\mathcal{A}$, the output of Program 2 is indistinguishable from that of Program 4, where U_ℓ and U'_ℓ are two independent uniform distributions over $\{0,1\}^{\ell(|x|)}$. In Program 4, conditioned on $E_1{}'$, $\mathcal{A}$ cannot distinguish the case where $\rho = 0$ from the case that $\rho = 1$, even in the information-theoretic sense. Therefore, $\Pr[E_2 \mid E_1{}'] = \frac{1}{2}$. ■

```
LET $\langle x, t_x \rangle \leftarrow G(1^n)$.
LET $\mathcal{O} \leftarrow_R \mathbb{F}_\ell$.
LET $z \leftarrow D(x)$.
LET $y \leftarrow F(x,z)$.
LET $\rho \leftarrow_R \{0,1\}$.
IF $\rho = 0$ THEN
    LET $r \leftarrow U_\ell$.
ELSE
    LET $r \leftarrow U'_\ell$.
OUTPUT $(x, \langle y, r \rangle)$.
```

Program 4. A modified version of Program 2.

Given lemmas 1 and 2, inequality (14) holds.

$$\begin{aligned} \frac{1}{2} = \Pr[E_2 \mid E_1{}'] &= \frac{\Pr[E_2 \cap E_1{}']}{\Pr[E_1{}']} \\ = \frac{\Pr[E_2 - E_1]}{\Pr[E_1{}']} &= \frac{\Pr[E_2] - \Pr[E_1 \cap E_2]}{\Pr[E_1{}']} \\ \geq \frac{\Pr[E_2] - \Pr[E_1]}{\Pr[E_1{}']} &= \frac{\Pr[E_2] - \text{negl}(|x|)}{1 - \text{negl}(|x|)} \quad . \end{aligned} \tag{14}$$

Thus $\Pr[E_2] \leq \frac{1}{2} + \frac{1}{2}\text{negl}(|x|)$, and Part (i) follows. ■

Part (ii)— Any hard-to-approximate relation is trivially hard to decide, since for infinitely many n, $S(1^n)$ produces tuples which no efficient algorithm $\mathcal{A}$ can decide their membership with probability better than $\frac{1}{2} + \text{negl}(n)$. ■

Part (i) of Theorem 3 provides us with an invaluable tool to prove the main results of this paper: That is, Protocol 2 is a ZK proof. This proposition is studied in the next section.

5 Main Results

In the previous section, we proved that Protocol 2 is a proof of computational ability. In this section, we will concentrate on the ZK property of this protocol. We start by proving that Protocol 2 is ZK in the sense of Definition 7.

Theorem 4. *Protocol 2 is passive-observer CZK.*

Proof. Consider the simulator described by Program 5. The simulator simply lets V^* send its challenge, and then answers the challenge with a random string. A more detailed description follows.

Since the algorithm of the simulator is efficient, algorithm $\mathcal{A}$ of Definition 7 can easily run it over and over, and the simulator should always give the same answer to the same challenge. This is because $\mathcal{A}$ can easily force V^* to ask the same challenge y, since due to the universal quantifier over V^* and $\mathcal{A}$ in Definition 7, $\mathcal{A}$ might ask for a deterministic V^*. The simulator uses function $\Psi(\cdot)$ to "memorize" queries already asked (Steps 0 and 2–4), similar to Program 3. Note that Step 0 of M_{V^*} runs only once, during the initialization phase.

```
0:  LET $\Psi \leftarrow \varnothing$.
    (This step is taken only while initializing).

1:  LET $\langle x, y \rangle \leftarrow \mathcal{V}^*(1^n)$.
2:  IF $y \notin \mathrm{dom}(\Psi)$ THEN
3:      LET $\psi \leftarrow_R \{0,1\}^{\ell(|x|)}$.
4:      LET $\Psi \leftarrow \Psi \cup \{(y, \psi)\}$.
5:  OUTPUT $\langle 1^n, x, y, \Psi(y) \rangle$.
```

Program 5. The Simulator M_{V^*} of Theorem 4.

Step 1 uses V^*'s `next-message` function $\mathcal{V}^*(\cdot)$ to get the query. Here, we assume that $\mathcal{V}^*(\cdot)$ uses the internal coin tosses of V^*, and didn't mention the randomness explicitly.

Finally, Step 5 outputs the proof transcript. According to Part (i) of Theorem 3, no efficient algorithm can distinguish such "simulated" output from the real output with probability better than $\frac{1}{2} + \mathrm{negl}(|x|)$, when x is large enough. This completes the proof. ■

Corollary 1 (cf. [AH87, AH91]). *Relative to a random oracle $\mathcal{O}$, there exists a two-step passive-observer CZK proof for some relation*[23] $\hat{\mathcal{R}} \notin \mathcal{BPP}^{\mathcal{O}}$.

Proof. Corollary 1 follows from Theorem 4, and Part (i) of Theorem 3. (The latter is used to show that $\hat{\mathcal{R}} \notin \mathcal{BPP}^{\mathcal{O}}$.) ■

Theorem 4 shows that no passive observer can gain knowledge by eavesdropping on the transcript of Protocol 2. Can we extend this result from passive observers to dishonest verifiers? That is, can we migrate from the above simulator—which constructed the transcript of the protocol—towards some simulator that constructs the view of any verifier?

To answer the above question, we first note that Protocol 2 is HVCZK: On input 1^n, and for every $r \in \{0,1\}^*$, the simulator M first runs $\mathcal{V}_r(1^n)$ to get the challenge (i.e. $y = F(x,z)$). It then replies with some random value, and monitors the query V makes to $\mathcal{O}$. Due to the honesty of V, it will query $\mathcal{O}$ at z. Having z at hand, M will output $\langle 1^n, r, \perp, x, y, \mathcal{O}(z) \rangle$. The symbol $\perp$ denotes that the auxiliary input is empty, since the honest verifier does not use such input. Note that the simulated view is *identical* to the real view. Therefore, not only Protocol 2 is HVCZK, but it is honest-verifier PZK (HVPZK).

Another possible way of outputting a view identical to the real view is as follow: On input $\langle 1^n, r, \perp \rangle$, M runs algorithms G, D, and F to get x, y, and z, and outputs $\langle 1^n, r, \perp, x, y, \mathcal{O}(z) \rangle$.

In contrast to the passive-observer and honest-verifier cases, it is impossible to prove the ZK property for general verifiers using "classical" ZK definitions. In particular, old techniques such as *rewinding* the verifier or *monitoring* the verifier's queries to the random oracle will not help the simulator. Theorem 5 states this formally.

Theorem 5. *Protocol 2 is not ZK in the classical* **black-box** *CZK sense.*

Proof (Sketch). Consider the family $\mathbf{V}^*$ of dishonest verifiers, each of which has a polynomial-time computable permutation, hard-coded into its program. That is, to every verifier $V^* \in \mathbf{V}^*$, a polynomial-time computable permutation $\pi_{V^*} : 1^* \times \{0,1\}^* \to \{0,1\}^*$ is associated, where the first input determines the running time, while the second input specifies the domain over which the permutation is applied. (Note that

[23] We implicitly generalized $\mathcal{BPP}$ to cover relations, in addition to languages.

```
LET $r' \leftarrow \pi_{V^*}(1^n, r)$, $\langle x, t_x \rangle \leftarrow G_{r'}(1^n)$, $z \leftarrow D_{r'}(x)$, $y \leftarrow F_{r'}(x, z)$.
SEND $\langle x, y \rangle$ TO $P$.
RECEIVE $w$ FROM $P$.
ABORT interaction.
INVESTIGATE $w$ to gain knowledge.
```

Program 6. The Program of $V^* \in \mathbf{V}^*$.

only a polynomial prefix of the second input is used.) Since there are infinitely many such permutations, $\mathbf{V}^*$ will be an infinite family. The verifiers in $\mathbf{V}^*$ are dishonest since they do not use their random input r directly; instead, they first compute $r' = \pi_{V^*}(1^n, r)$, and then use r' instead of r. Since π_{V^*} is a permutation and r is a uniformly random string, r' will be a uniformly random string as well.

The program of $V^* \in \mathbf{V}^*$ is described in Program 6. M cannot acquire z by monitoring V^*'s queries to $\mathcal{O}$, since it does not make any queries at all. Therefore, it should try another trick.

The black-box CZK requires the existence a *universal* simulator M for all verifiers. Since there are infinitely many permutations π_{V^*}, M cannot incorporate the description of all such permutations into its program. In contrast, since the distinguisher $\mathcal{A}$ is chosen after V^*, we can assume that π_{V^*} is hard-coded into $\mathcal{A}$'s program. Fixing r, $\mathcal{A}$ can deterministically verify whether the output of the simulator is *identically* distributed to the real view: Assume that, on input $\langle 1^n, r, \xi \rangle$, M outputs $\langle 1^n, r, \xi, x', y', w \rangle$. Then, $\mathcal{A}$ computes $r' = \pi_{V^*}(1^n, r)$, and outputs 1 if and only if:

- x' equals the first component of $G_{r'}(1^n)$; AND
- y' equals $F_{r'}(x', D_{r'}(x'))$; AND
- w equals $\mathcal{O}(D_{r'}(x'))$.

Therefore, given the input $\langle 1^n, r, \xi \rangle$, the values of x', y', and w are uniquely determined. This implies that M has one of the following two approaches to satisfy $\mathcal{A}$:

1. Find π_{V^*} so that it can compute r', and therefore the second component of $G_{r'}(1^n)$ (that is, the trapdoor).
2. Get $\langle x, y \rangle$ using the `next-message` function of V^*, and try to find z' such that $\mathcal{O}(z') = \mathcal{O}(f_x^{-1}(y))$.

The first approach is infeasible due to the universality of the simulator (described earlier). The second approach is infeasible since it involves inverting the one-way permutation f_x.

We conclude that a black-box simulator for Protocol 2 does not exist. ■

As a side note, if we assume that Protocol 2 is ZK in the *classical* **auxiliary-input** sense (which is less strict than the black-box sense), then Theorem 6 holds.

Theorem 6. *If Conjecture 1 holds, and Protocol 2 is ZK in the* classical ***auxiliary-input*** *sense, it will be a* non-contrived[24] *example of a protocol secure (≡ZK) in the RO Model which, for no implementation of the random oracle in the Standard Model will remain secure (≡ZK).*

Proof. Conjecture 1 states that nontrivial two-step auxiliary input ZK proofs of computational ability do not exist in the Standard Model. if Protocol 2 is ZK in the *classical* auxiliary-input sense, and because it is two-step, any implementation of the random oracle (using a single function or a function ensemble) will not preserve the ZK property in the Standard Model. ■

[24] Canetti, Goldreich, and Halevi [CGH98, CGH04, CGH08] proved this theorem for *contrived* encryption and signature schemes.

```
LET $\rho \leftarrow_R \{0,1\}$.
LET $\langle x,y \rangle \leftarrow \mathcal{V}_{r,\xi}(1^n)$.
LET $\mathcal{O} \leftarrow \mathbb{F}_\ell$.
IF $\rho = 1$ THEN
    LET $w \leftarrow \mathcal{P}(1^n,x,y)$.
ELSE
    LET $w \leftarrow M(1^n,r,\xi,x,y)$.
LET $d \leftarrow \mathcal{A}^{\mathcal{O}}(1^n,r,\xi,x,y,w)$.
```

Program 7. An experiment used to prove Theorem 8.

Now that we proved that Protocol 2 does not possess a black-box simulator, let us intuitively justify why failure to show the existence of such simulator does not rule out the ZK property for this protocol. Consider a spectrum of verifiers, from the honest one (which exactly follows the prescribed rules of protocol), to the most adversarial ones (which deviate from the rules of protocol as much as they can). We proved earlier that the protocol is HVPZK. Consequently, the protocol is ZK at one extreme of the spectrum.

In the middle of the spectrum, we have verifiers with "some knowledge" about z. As an example, consider verifiers with "full knowledge" about z, such as those of Theorem 5. Such verifiers "know" z, but do not let the simulator learn it too. However, the protocol is *conceptually* ZK for such verifiers, since the message sent by P (i.e. $\mathcal{O}(z)$) will not carry more knowledge than z.

We now look at the other extreme of the spectrum, where the (dishonest) verifier V^* "knows" nothing about z (beyond what can be efficiently computed from y). For instance, consider a verifier with empty auxiliary input, which sets $y \leftarrow_R \{0,1\}^{|x|}$ and sends it to P. The protocol is ZK for such V^* as well, since it has no advantage over a passive observer; and by Theorem 4, we know that the protocol is CZK for passive observers.

We intuitively and *conceptually* argued that Protocol 2 is ZK for special classes of verifiers in different parts of the spectrum. However, is it true for other types of verifiers? (That is, those dishonest verifiers possessing some *partial* knowledge about z.)

To answer the above question, let us present several notations. For all $n \in \mathbb{N}$, and every efficient (possibly dishonest) verifier V (of Protocol 2), and for every $r, \xi \in \{0,1\}^*$, let $\mathcal{V}_{r,\xi}(1^n)$ denote the `next-message` function of V on common input 1^n, random input r, and auxiliary input ξ. Similarly, let $\mathcal{P}(1^n,x,y)$ be the `next-message` function of the (honest) prover P on common input 1^n and on $\langle x,y \rangle \leftarrow \mathcal{V}_{r,\xi}(1^n)$. (The random input, if any, is implicit in this function.) Let M be the simulator in Program 5, with the difference that Step 1 is no longer needed.

Now consider the experiment in Program 7, which is similar to Program 2. Define $\Delta_{\mathcal{A}} \stackrel{\text{def}}{=} \Delta_{\mathcal{A}}(x;r,\xi) \stackrel{\text{def}}{=} \Pr[d = \rho]$, where the probability is taken over the random selection of $\mathcal{O}$ and the internal coin tosses of $\mathcal{A}$ and M. We did not mentioned the internal coin tosses of P since there exists a deterministic algorithm, which can mimic the behavior of P on y. That is, $f_x^{-1}(y)$ can be computed deterministically.

As stated in the informal analysis above, if V is honest, there exists an efficient machine $\mathcal{A}$ which can easily distinguish the real proof from the simulated proof. This is due to the fact that since the honest V "knows" z, there exists some efficient machine $\mathcal{A}$ which, given the same common and random inputs as V, can regenerate z. Then, $\mathcal{A}$ queries $\mathcal{O}$ at z, and outputs 1 if its *sixth* input (i.e. w) is $\mathcal{O}(z)$, and 0 otherwise. Thus, for the honest verifier and such $\mathcal{A}$, $\Delta_{\mathcal{A}} \geq 1 - \text{negl}(n)$. (There's a negligible probability that the simulator outputs the correct response.) On the other hand, if V is the dishonest verifier described in the

```
1:  ESTIMATE $\delta_{\mathcal{A}}$.
2:  FOR $i \leftarrow 1$ TO $\frac{|x|}{\delta_{\mathcal{A}}}$ DO
3:      LET $\rho_1 \leftarrow_R \{0,1\}^*$.
4:      LET $\rho_2 \leftarrow_R \{0,1\}^*$.
5:          PROCEED $\mathcal{A}_{\rho_1}\left(M_{\rho_2}(1^n,r,\xi,x,y)\right)$.
6:          IF $\mathcal{A}_{\rho_1}$ queries $\mathcal{O}$ THEN
7:              OUTPUT the query.
8:              HALT.
9:  PERFORM “Exhaustive Search” on $y$.
```

Program 8. The Universal Knowledge Extractor K of Theorem 8.

informal analysis above (where V^* knew nothing about z), we have $\Delta_{\mathcal{A}} \leq \frac{1}{2} + \mathrm{negl}(n)$ for every efficient algorithm $\mathcal{A}$ and large enough n's. (The proof is similar to that of Theorem 4.)

Define the advantage of $\mathcal{A}$ in guessing ρ (and therefore distinguishing real and simulated views) as $\delta_{\mathcal{A}} \overset{\text{def}}{=} 2\Delta_{\mathcal{A}} - 1$. By definition, the $|\delta_{\mathcal{A}}|$ equals the absolute difference of two probabilities: The probability that $\mathcal{A}$ outputs 1 given the real view, and the probability that $\mathcal{A}$ outputs 1 given the simulated view.

Theorem 7. *For every efficient $\mathcal{A}$, the probability that $\mathcal{A}$ queries $\mathcal{O}$ at z is at least $\delta_{\mathcal{A}}$.*

Proof. Let events E_1 and E_2 be defined as in the proof of Theorem 3. That is, let E_1 denote the event that $\mathcal{O}$ is queried at point z, and E_2 denote the event that $d = \rho$. Note that $\Pr[E_2] = \Delta_{\mathcal{A}}$, and $\Pr[E_2|E_1{}'] = \frac{1}{2}$. (The latter can be proven using Lemma 2.) By inequality (14): $\Pr[E_1] \geq 2\Delta_{\mathcal{A}} - 1 = \delta_{\mathcal{A}}$. ■

Let $\mathcal{A}$ be an efficient machine which queries $\mathcal{O}$ at z with advantage $\delta_{\mathcal{A}}$. Then, there exists an efficient machine $\mathcal{A}'$ which queries $\mathcal{O}$ at z with the same advantage $\delta_{\mathcal{A}}$, while $\mathcal{A}'$ makes at most 1 query to $\mathcal{O}$. This is because $\mathcal{A}'$ can use the $\mathcal{A}$ as a subroutine, which, on an oracle query by $\mathcal{A}$:

1. returns $\mathcal{O}(z)$ if the query is at point z;
2. returns a random but consistent answer otherwise.

We call $\mathcal{A}'$ a *single-query* machine. Intuitively, the knowledge of $\mathcal{A}'$ is no more than that of $\mathcal{A}$. Thus, statements about the knowledge of $\mathcal{A}$ can be "lower-bounded" by the knowledge of $\mathcal{A}'$, and in the rest, we only give propositions about single-query machines.

We now deal with verifiers with *partial* knowledge about z: Informally, For every (single-query) efficient algorithm $\mathcal{A}$, if $\mathcal{A}$ has advantage $\delta_{\mathcal{A}}$, then $\mathcal{A}$ "knows" z with probability $\max\{0, \delta_{\mathcal{A}}\}$. Using the formalized concept of "a machine knowing something" in Definition 2, we state the main result of this paper: Theorem 8.

Theorem 8 (Main Theorem). *There exists a constant $c > 0$ and a probabilistic oracle machine K (the Universal Knowledge Extractor) such that for every efficient single-query algorithm $\mathcal{A}$, and for all strings 1^n, r, ξ, x, and y as described above, if $\delta_{\mathcal{A}} > 0$, then $K^{\mathcal{A}}(1^n, r, \xi, x, y)$ outputs $z = f_x^{-1}(y)$ in expected time upper-bounded by $\frac{n^c}{\delta_{\mathcal{A}}}$.*

Proof. Note that since M is an efficient algorithm, K can incorporate its code within itself. Now, consider the algorithm of the universal knowledge extractor K, described by Program 8.

Program 8 resembles Program 1. One difference is the command `PROCEED` in Step 5. This command allows $\mathcal{A}$ to run, until it "pauses." The pause is due to $\mathcal{A}$ entering either the *halt* state (in which its execution terminates), or the *query* state (in which $\mathcal{A}$ writes some query on its query tape and waits for response from the random oracle). Since we assumed that $\mathcal{A}$ is a single-query machine, if the latter case happens, the query is indeed the string z, and we output it at Step 7, and finish the search at Step 8. Otherwise, a new round of search begins.

At each iteration of the loop, $\mathcal{A}$ either makes the required query or halts. Since, according to Theorem 7, $\mathcal{A}$ queries $\mathcal{O}$ at point z with probability at least $\delta_{\mathcal{A}}$ (assuming $\delta_{\mathcal{A}} > 0$), we expect that, at each iteration, Step 7 gets to run with probability at least $\delta_{\mathcal{A}}$. An analysis similar to the proof of Theorem 1 shows that, for large enough x's, the probability that Step 9 gets to run is no more than $(1-\delta_{\mathcal{A}})^{\frac{n}{\delta_{\mathcal{A}}}} < e^{-n}$.

Consequently, since Step 9 gets to run with negligible probability, the expected running time of Program 8 is at most $\frac{n}{\delta_{\mathcal{A}}} O(1)$, as required by Theorem 8. ■

5.1 The Simulation-Extraction Paradigm

Before proposing the Simulation-Extraction Paradigm, let us study the properties of Protocol 2, which we exploited to prove its ZK property:

(a) The honest verifier of Protocol 2 "knows" the right answer when sending its challenge. In addition, the protocol is *HVZK*.
(b) Protocol 2 is not a repetition/composition of some primitive protocol.
(c) The verifier of Protocol 2 uses private coins.
(d) Protocol 2 uses the RO Model.

We "believe" that Property (a) is crucial for our paradigm, while we do not know whether other properties are required or not.

Below, we formalize what we mean by the term *Simulation-Extraction Paradigm* (abbreviated as *SE Paradigm*):

Definition 12 (Black-Box ZK in the SE Paradigm). *Let $\mathcal{V}_{x,r,\xi}(\overline{m})$ and $V_r(\xi)$ be defined as in Definition 6. We call an interactive protocol $\langle V \leftrightarrow P\rangle$* **black-box SE-CZK** *for P on $L = L_R$, if there exists an (expected) PPT-OTM M, a probabilistic OTM K, and some constant $c_1 > 0$, such that for every PPT-ITM V^*, every PPT-OTM $\mathcal{A}$ (the distinguisher), all constants $c_2 > 0$, every $x \in L$ and all $r, \xi \in \{0,1\}^*$, the following property holds:*

> *Let $\delta_{\mathcal{A}}$ denote the* advantage[25] *of $\mathcal{A}$ in distinguishing the real and simulated views. If $\delta_{\mathcal{A}} > 0$, then $K^{V^*,\mathcal{A}}(x,r,\xi)$ can output a view in expected running time upper-bounded by $\frac{|x|^{c_1}}{\delta_{\mathcal{A}}}$, such that $\mathcal{A}$'s absolute advantage of distinguishing K's output from the real view is less than $|x|^{-c_2}$.*

The definition in the RO Model follows easily. □

Several observations are in order:

- All existing black-box CZK protocols are black-box SE-CZK. This is due to the fact that for all existing black-box CZK protocols, and for all PPT-OTM $\mathcal{A}$, $\delta_{\mathcal{A}}$ is a negligible quantity in $|x|$. Therefore, K's expected running time is upper-bounded by some super-polynomial. However, M already simulates the view in polynomial-time; and therefore we can take K to be identical to M.
- K is not too powerful. In other words, consider a protocol which is not black-box CZK; in the sense that for every PPT-OTM simulator M, there exists a PPT-OTM $\mathcal{A}$ which distinguishes the simulated and real views with noticeable advantage. Then, the expected running time of K will be bounded by some polynomial, much the same as M. Therefore, the only advantage that K has over M is that K has oracle access to $\mathcal{A}$.

[25] That is, $|\delta_{\mathcal{A}}| = \left|\Pr\left[\mathcal{A}^{\mathcal{V}^*_{(x,r,\xi)}}\left(M^{\mathcal{V}^*_{(x,r,\xi)}}(x,r,\xi)\right) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{V}^*_{(x,r,\xi)}}(\mathbf{view}\langle V_r^*(\xi), P\rangle(x)) = 1\right]\right|$

- The above discussion demonstrates that the SE Paradigm simplifies proving positive results (i.e. a protocol *is* ZK), while it complicates proving negative results (i.e. a protocol is *not* ZK). In particular, it *invalidates* some proofs which were totally valid in the classical sense (for example, see the proof of Theorem 3 in [Wee09][26]).
- We already proved, in the context of Protocol 2, how K can simulate a view indistinguishable from the real view. Therefore, by definition, Protocol 2 is black-box SE-CZK.

6 Conclusion and Open Problems

The concept of "zero-knowledge" was proposed before "knowledge" was formalized, and it was not revised thereafter. The current formalism is too strict to include all the protocols which are *conceptually* zero-knowledge. As an example, we came up with one such protocol, and formally proved that it was zero-knowledge. The formal proof provided us with a relaxed paradigm for proving the zero-knowledge property.

The new formalization of zero-knowledge led us to many open problems, a few of which are listed below:

1. Does Conjecture 1 hold? That is, is it true that 1-round proofs of knowledge (or proofs of computational ability) exist only for trivial relations?
2. At the beginning of Section 5.1, we discussed several properties, labeled (a)–(d). Is it true that all of these properties are required? Specially, is it possible to use the paradigm in models other than the RO Model, such as the *Common Reference String* (CRS) Model or the *Standard* Model?
   In addition, are there any other required/sufficient properties which, if satisfied by a protocol, make it a candidate for the SE Paradigm?
3. We proved that Protocol 2 does not have a classical black-box CZK simulator. Can we conceive of other useful protocols, which are not zero-knowledge in the classical sense but are zero-knowledge in our paradigm?
4. Does revising the concept of zero-knowledge result in the revision of any concept/theorem in the foundations of cryptography? We are specially interested in revising the concept of zero-knowledge proofs in the *non-black–box* sense [Bar01], and in the *CS Proofs* model [Mic94, Mic00]. Moreover, we are interested in theorems about the composition of ZK protocols: Is SE-ZK closed under various compositions?
5. In Definition 12, the knowledge extractor K tries to extract $\mathcal{A}$'s knowledge as if $\mathcal{A}$ is a **prover** of knowledge. This might be inappropriate in some cases, since $\mathcal{A}$ is a "judge," not a prover. Therefore, its one-bit output may be inadequate for K to extract $\mathcal{A}$'s knowledge within a limited number of steps. A workaround is to give K non-black–box access to $\mathcal{A}$, similar to the approach [BGGL01]. Yet a better solution is to formalize the concept of knowledge in such a way that it is not limited to parties who engage in proving, but includes verifiers and distinguishers as well.

Acknowledgements

We would like to thank Mohammad Mahmoody for his fruitful and valuable discussions.

References

[AH87] William Aiello and Johan Håstad. Perfect Zero-Knowledge Languages can be Recognized in Two Rounds. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS'87*), pages 439–448, 1987.

[26] Note that we are stating nothing about the validity of the theorem itself. We are merely pointing that the proof becomes invalid.

[AH91] William Aiello and Johan Håstad. Relativized Perfect Zero Knowledge is not $\mathcal{BPP}$. *Journal of Information and Computation*, 93(2):223–240, 1991.

[Bab85] László Babai. Trading Group Theory for Randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (*STOC '85*), pages 421–429, New York, NY, USA, 1985. ACM.

[Bar01] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '01*), pages 106–115, Las Vegas, Nevada, USA, 2001. IEEE Computer Society.

[Bar03] Boaz Barak. How to Go Beyond the Black-Box Simulation Barrier, 2003. Available from: `http://www.cs.princeton.edu/~boaz/Papers/nonbb.ps`. See [Bar01] for the proceedings version.

[BC86] Gilles Brassard and Claude Crépeau. Non-Transitive Transfer of Confidence: A Perfect Zero-knowledge Interactive Protocol for SAT and Beyond. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '86*), pages 188–195. IEEE Computer Society, 1986.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences* (*JCSS*), 37(2):156–189, 1988.

[BCY89] Gilles Brassard, Claude Crépeau, and Moti Yung. Everything in NP can be Argued in Perfect Zero-Knowledge in a Bounded Number of Rounds. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming* (*ICALP '89*), pages 123–136, London, UK, 1989. Springer-Verlag.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive Zero-Knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and its Applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (*STOC '88*), pages 103–112, Chicago, Illinois, United States, 1988. ACM.

[BG81] Charles H. Bennett and John Gill. Relative to a Random Oracle A, $\mathcal{P}^A \neq \mathcal{NP}^A \neq$ co-$\mathcal{NP}^A$ with Probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

[BG92] Mihir Bellare and Oded Goldreich. Proving Computational Ability. Unpublished manuscript. Available from: `http://cseweb.ucsd.edu/~mihir/papers/poa.ps` or `http://www.wisdom.weizmann.ac.il/~oded/PS/poa.ps`, 1992.

[BG93] Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In *Advances in Cryptology—CRYPTO '92*, pages 390–420, London, UK, 1993. Springer-Verlag.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-Sound Zero-Knowledge and Its Applications. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science* (*FOCS '01*), pages 116–125. IEEE, 2001.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (*STOC '88*), pages 113–131, New York, NY, USA, 1988. ACM.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P} \overset{?}{=} \mathcal{NP}$ Question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil Vadhan. Many-to-One Trapdoor Functions and their Relation to Public-key Cryptosystem. In *Advances in Cryptology—CRYPTO '98*, pages 283–299. Springer-Verlag, 1998. Extended version available from `http://cseweb.ucsd.edu/~mihir/papers/tf.html`.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Class. *Journal of Computer and System Sciences* (*JCSS*), 36(2):254–276, 1988.

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st Annual ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[CEv87] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *Advances in Cryptology—EUROCRYPT '87*, pages 127–141, London, UK, 1987. Springer-Verlag.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable Zero-Knowledge (extended abstract). In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 235–244, Portland, Oregon, United States, 2000. ACM.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (preliminary version). In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (*STOC '98*), pages 209–218, New York, NY, USA, 1998. ACM.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. *Journal of the ACM* (*JACM*), 51(4):557–594, 2004.

[CGH08] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited. Technical Report arXiv:cs/0010019v1, arXiv.org, February 1, 2008. Available from: `http://arxiv.org/abs/cs/0010019`.

[Cha87] David Chaum. Demonstrating that a Public Predicate Can Be Satisfied without Revealing any Information about How. In *Advances in Cryptology—CRYPTO '86*, pages 195–199, London, UK, 1987. Springer-Verlag.

[CM99] Jan Camenisch and Markus Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In *Advances in Cryptology—EUROCRYPT '99*, pages 107–122. Springer-Verlag, 1999.

[Con88] Anne Condon. Space-Bounded Probabilistic Game Automata. In *Proceedings of the 3rd Annual IEEE Structure in Complexity Theory Conference* (*SCT '88*), pages 162–174, Washington, DC, USA, 1988. IEEE Computer Society. See [Con91] for journal version.

[Con91] Anne Condon. Space-Bounded Probabilistic Game Automata. *Journal of the ACM* (*JACM*), 38(2):472–494, 1991. See [Con88] for conference version.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent Zero-Knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (*STOC '98*), pages 409–418, 1998.

[DS89] C. Dwork and L. Stockmeyer. On The Power of 2-Way Probabilistic Finite State Automata (Extended Abstract). In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '89*), pages 480–485, Washington, DC, USA, 1989. IEEE Computer Society.

[DS90] Cynthia Dwork and Larry J. Stockmeyer. Zero-Knowledge with Finite State Verifiers (Extended Abstract). In *Advances in Cryptology—CRYPTO '88*, pages 71–75, London, UK, 1990. Springer-Verlag. See also [DS89,DS92a, DS92b].

[DS92a] Cynthia Dwork and Larry Stockmeyer. Finite State Verifiers I: The Power of Interaction. *Journal of the ACM* (*JACM*), 39(4):800–828, 1992.

[DS92b] Cynthia Dwork and Larry Stockmeyer. Finite State Verifiers II: Zero Knowledge. *Journal of the ACM* (*JACM*), 39(4):829–858, 1992.

[FFS87] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-Knowledge Proofs of Identity (extended abstract). In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (*STOC '87*), pages 210–217, New York, NY, USA, 1987.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[For87] Lance Fortnow. The Complexity of Perfect Zero-Knowledge. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (*STOC '87*), pages 204–209, New York, NY, USA, 1987. ACM. See the full version in [For89].

[For89] Lance Fortnow. The Complexity of Perfect Zero-Knowledge. In Franco P. Preparata and Silvio Micali, editors, *Advances in Computing Research*, volume 5: Randomness & Computation, pages 327–343. JAI Press, Inc., 1989.

[FS90] Uriel Feige and Adi Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing* (*STOC '90*), pages 416–426, New York, NY, USA, 1990. ACM.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *Journal of the ACM* (*JACM*), 33(4):792–807, 1986.

[GHY85] Zvi Galil, Stuart Haber, and Moti Yung. A Private Interactive Test of a Boolean Predicate and a Minimum-Knowledge Public-Key Cryptosystems. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '85*), pages 360–371, Washington, DC, USA, 1985. IEEE Computer Society. See [GHY89] for journal version.

[GHY89] Zvi Galil, Stuart Haber, and Moti Yung. Minimum-Knowledge Interactive Proofs for Decision Problems. *SIAM Journal on Computing*, 18(4):711–739, 1989.

[GK90] Oded Goldreich and Hugo Krawczyk. On the Composition of Zero-Knowledge Proof Systems. In Mike Paterson, editor, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming* (*ICALP '90*), volume 443 of *Lecture Notes in Computer Science*, pages 268–282, Warwick University, England, 1990. Springer.

[GK96] Oded Goldreich and Hugo Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 291–304, 1985.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract). In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '86*), pages 174–187, 1986. See [GMW91] for the journal version.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM* (*JACM*), 38(3):690–728, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, 7:1–32, 1994.

[Gol93] Oded Goldreich. A Uniform-Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, 6(1):21–53, 1993.

[Gol97] Oded Goldreich. A Sample of Samplers: A Computational Perspective on Sampling (survey). Technical Report TR97-020, Electronic Colloquium on Computational Complexity (ECCC), 1997. `http://eccc.hpi-web.de/report/1997/020/`.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.

[HM98] Shai Halevi and Silvio Micali. More on Proofs of Knowledge. Cryptology ePrint Archive, Report 1998/015, 1998. `http://eprint.iacr.org/`.

[Kil88] Joe Kilian. Zero-knowledge with Log-Space Verifiers. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '88*), pages 25–35. IEEE Computer Society, 1988.

[Kil91] Joe Kilian. Interactive Proofs with Provable Security Against Honest Verifiers. In *Advances in Cryptology—CRYPTO '90*, pages 378–392, London, UK, 1991. Springer-Verlag.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower Bounds for Zero Knowledge on the Internet. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '98*), pages 484–492, Washington, DC, USA, 1998. IEEE Computer Society.

[LMS05] Matt Lepinski, Silvio Micali, and Abhi Shelat. Fair-Zero Knowledge. In *Theory of Cryptography—*(*TCC '05*), pages 245–263. Springer-Verlag, 2005.

[Mic94] Silvio Micali. CS Proofs. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '94*), pages 436–453, Santa Fe, New Mexico, USA, 1994. IEEE Computer Society.

[Mic00] Silvio Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[Nie02] Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *Advances in Cryptology—CRYPTO 2002*, pages 111–126. Springer-Verlag, 2002.

[Ore87] Yair Oren. On the Cunning Power of Cheating Verifiers: Some Observations about Zero Knowledge Proofs (Extended Abstract). In Ashok K. Chandra, editor, *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '87*), pages 462–471, Los Angeles, California, USA, 1987. IEEE Computer Society Press.

[Pas03] Rafael Pass. On Deniability in the Common Reference String and Random Oracle Model. In *Advances in Cryptology—CRYPTO 2003*, pages 316–337. Springer-Verlag, 2003.

[RK99] Ransom Richardson and Joe Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *Advances in Cryptology—EUROCRYPT '99*, pages 415–431. Springer-Verlag, 1999.

[TW87] Martin Tompa and Heather Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '87*), pages 472–482. IEEE, 1987.

[Unr07] Dominique Unruh. Random Oracles and Auxiliary Input. In *Advances in Cryptology—CRYPTO 2007*, pages 205–223. Springer-Verlag, 2007.

[Wee09] Hoeteck Wee. Zero Knowledge in the Random Oracle Model, Revisited. In *Advances in Cryptology—ASIACRYPT 2009*, pages 417–434. Springer-Verlag, 2009.

[Yao82] Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (extended abstract). In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science* (*FOCS '82*), pages 80–91. IEEE, 1982.

[Yun90] Moti Yung. Zero-Knowledge Proofs of Computational Power. In *Advances in Cryptology—EUROCRYPT '89*, pages 196–207. Springer-Verlag, 1990.