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

Mohammad Sadeq Dousti and Rasool Jalili*

Network Security Center, Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran.
{dousti@ce., jalili@}sharif.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 try to prove that this requirement is too strict. In particular, assuming the existence of trapdoor one-way permutations, we provide a protocol which does not carry knowledge in the accepted sense, while there are efficient algorithms to distinguish a simulated view from the real view. To prove that the protocol is zero-knowledge, we show that if an efficient distinguisher succeeds in telling the simulated and real views apart, it should possess the knowledge of the prover. This new formalization is clarified through a detailed study of the protocol. We conclude the paper by providing a paradigm for proving the zero-knowledge property, called the *Simulation-Extraction Paradigm*, which can be exploited as a relaxed alternative to the strict *Simulation Paradigm*.

Key words: Zero-Knowledge Proof, Proof of Computational Ability, Proof of Knowledge, Trapdoor One-Way Permutations, Random Oracle, Simulation 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, assume that party A sends party B a random string r. This protocol carries information, but not knowledge (since party B could itself generate some random string). On the other hand, assume that party A sends party B the factorization of some number n. This protocol carries no information, but it *might* carry knowledge (since B might not be able to compute the factorization by itself).

* Part of this work was done while the author was visiting the Department of Computer Science, University of California, Davis.

The above sense of knowledge is intuitive, not formal. In [GMR85], the authors tried to formalize what it means for a protocol to carry no knowledge. Informally, they state that a protocol $\langle V \leftrightarrow P \rangle$ is zero-knowledge with respect to P if V can compute everything it sees on its tapes. A great deal of research was conducted on numerous aspects of this definition or its variants. Specifically, several papers (such as [FS87,TW87,FFS87,FFS88]) tried to formalize the concept of knowledge, but none of them suggested a satisfactory formalism. This was until the work of Bellare and Goldreich [BG93], which put forward a fully formalized definition of knowledge.

Common sense tells us that, a new formalism for the concept of *knowledge* must result in a new formalism for the concept of *zero-knowledge*. Unfortunately, the community has never considered a full revision of zero-knowledge based on [BG93].

In this paper, we show that the "classical" formalization of zero-knowledge—based on the existence of an efficient simulator—is too strict. We demonstrate a protocol that, according to the definition of knowledge in [BG93], is zero-knowledge; while it might be hard or even impossible to find the simulator required by the classical definition. Specifically, we show that every efficient machine which can distinguish the real and simulated views of the verifier of such protocol must be in possession of the knowledge of the prover (in the sense of [BG93]). We try to give intuitive examples and guide the reader in a step-by-step manner so that the final formalization does not seem perplexing.

The rest of this paper is organized as follows: 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 become allegedly zero-knowledge. Section 5 contains the main results of this paper: It suggests a new formalization of the concept of zero-knowledge, and proves that the protocol presented in Section 4 is zero-knowledge in the "new" sense. In addition, this section puts forward a paradigm for proving the zero-knowledge property of a protocol in the new sense. Finally, Section 6 concludes 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 $n \in \mathbb{N}$, $x \in \{0,1\}^n$, 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.[1] 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.[2] 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.

$\Pr[P_1, P_2, \ldots, P_n : E]$ 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 $\mathbb{E}[P_1, P_2, \ldots, P_n : E]$.

The worst-case running time of a Turing machine M is denoted as a function $t_M(\cdot)$ of its input. The term "efficient" is used identical to "probabilistic polynomial-time," which may be abbreviated as *PPT*. By polynomial-time, we mean an algorithm whose running time is bounded by some predetermined polynomial on its input. 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."

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

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

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

Let P and V be two interactive functions [BG93], which participate in an interactive proof.[3] Define:

– $\mathbf{state}\langle V, P\rangle(x)$: The state of V after interacting with P on common input x. The state is either *ACC* (for accept) or *REJ* (for reject).

[1] 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).

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

[3] 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.

– **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] and [GM84].)

Definition 1. (Collection of Trapdoor One-Way Permutations). *Let $X \subseteq \{0,1\}^*$ be an (efficiently samplable) 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***[4] *the following two conditions hold:*

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

$$\Pr\Big[\langle x, t_x\rangle \leftarrow G\left(1^n\right),\ z \leftarrow D(x)\ :\ x \in X \cap \{0,1\}^n \\ \wedge\ z \in D_x \\ \wedge\ F(x,z) = f_x(z) \\ \wedge\ I\left(f_x(z), t_x\right) = z\Big] = 1\ . \qquad (2)$$

   The probability is taken over the internal coin tosses of algorithms G, D, F, and I.
2. **Hard to invert without trapdoor:** *For any probabilistic polynomial-time adversary $\mathcal{A}$, the following condition should hold:*

$$\forall c \in \mathbb{N},\ \exists n_0 \in \mathbb{N}\ \text{ s.t. }\ \forall n \geq n_0 \\ \Pr\Big[\langle x, t_x\rangle \leftarrow G\left(1^n\right),\ z \leftarrow D(x)\ :\ \mathcal{A}\left(f_x(z)\right) = z\Big] < n^{-c}\ . \qquad (3)$$

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

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

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 the probability in (2) to be $1 - \mathrm{negl}(n)$. In this case, algorithms G, D, F, and I are permitted to fail with some probability negligible in 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 $R(x) \stackrel{\text{def}}{=} \{y | \langle x, y\rangle \in R\}$, and $L_R \equiv \mathrm{dom}(R) \stackrel{\text{def}}{=} \{x | R(x) \neq \varnothing\}$. 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,*

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

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 $p(x) \stackrel{\text{def}}{=} \Pr\left[\mathbf{state}\langle V, P\rangle(x) = ACC\right] > \kappa(x)$, 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* $\mathrm{dom}(R)$*. 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:*

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

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 $\kappa\colon \{0,1\}^* \to \{0,1\}$. We call an interactive function 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^*. In other words,*

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

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 $x \in L_R$, and all $\gamma \in ACC_V(x)$, machine $K^P(x,\gamma,\cdot)$ satisfies:*

   If $p_x \stackrel{\mathrm{def}}{=} \Pr\left[\mathbf{state}\langle V^{D_x}, P\rangle(x) = ACC\right] > \kappa(x)$, 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 (multi-party) protocol is considered ZK for party P if by the end of protocol, parties other than P cannot compute "something that they could not compute before the protocol begins."

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

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

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. 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[5]). *Let $\mathcal{V}_{x,\xi,r}(\overline{m})$ 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. We may replace each input with a "dot" to denote that it is not fixed a priori. 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, if there exists a PPT-OTM M such that for every PPT-ITM V^*, and all PPT-OTM $\mathcal{A}$ (the distinguisher):*

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

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 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

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

protocol, rather than the *view* of (possibly dishonest) verifier. In this case, the protocol is CZK for any passive observer eavesdropping the protocol messages. Any protocol, CZK in the sense of Definition 5 is also passive-observer CZK, but not necessarily vice versa. Therefore, Definition 7 provides a 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 [AH87, AH91]). *We call an interactive protocol $\langle V \leftrightarrow P \rangle$* **passive-observer CZK** *for P on L, if for every PPT-ITM V^*, there exists a PPT machine M_{V^*} such that for every PPT algorithm $\mathcal{A}$ (the distinguisher):*

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

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.

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

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

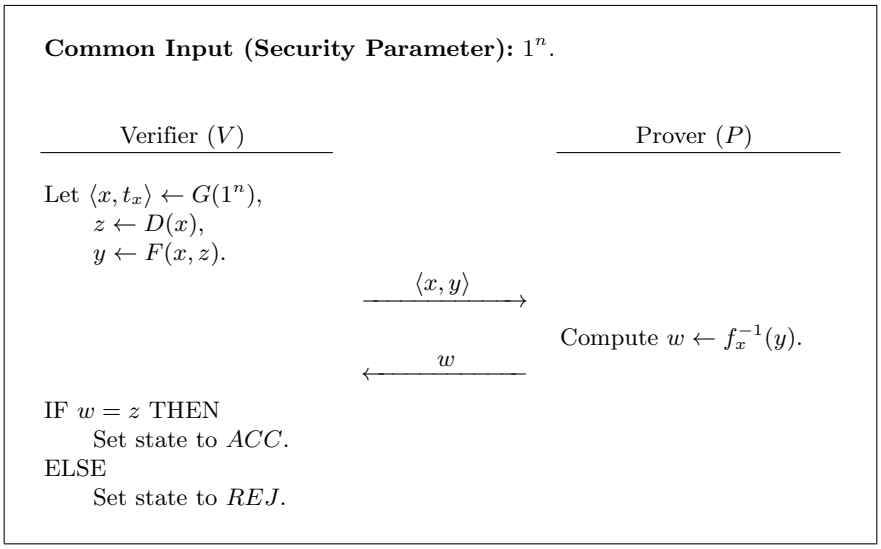
where the first probability is taken over the internal coin tosses of M and $\mathcal{A}$, and the second probability is taken over the internal coin tosses of V, P and $\mathcal{A}$. □

3 The Base Protocol

Let $\mathcal{F} \stackrel{\text{def}}{=} \{f_x\}_{x \in X}$ be a family 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 $R_x \stackrel{\text{def}}{=} \{\langle f_x(z), z\rangle\}_{z \in D_x}$ for all $x \in X$, and let $\mathcal{R}$ denote a family of binary relations: $\mathcal{R} \stackrel{\text{def}}{=} \{R_x\}_{x \in X}$. Note that $\mathcal{D} \stackrel{\text{def}}{=} \{D(x)\}_{x \in X}$ 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



Protocol 1. Base Protocol (Variant 1).

can solve $\mathcal{F}$ 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 honest prover and honest verifier's code. In the latter case, this assumption is justified by the fact that these algorithms are efficiently computable.

Another variant[6] 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.

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 $D(x)$ (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 $D(x)$ (with zero error), where $x \in X$ is chosen* a priori.

[6] Hereafter referred to as Variant 2.

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 $\{0,1\}^n$;
4. There is a minor error in the proof.[7]

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 7).

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 $p_x \stackrel{\text{def}}{=} \Pr[\mathbf{state}\langle V, P^*\rangle(x) = ACC]$, 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 $\mathcal{V}(x) = D(x)$ for all $x \in D_x$, 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 $D(x)$—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. We start by an intuitive analysis. The formal analysis is given in Remark 4.

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 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.[8]

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:

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

```
1:  ESTIMATE p_x.
2:  FOR i ← 1 TO n/p_x DO
3:      LET y ← V(x).
4:      LET w ← P*(x, y).
5:      IF y = F(x, w) THEN
6:          OUTPUT ⟨w, y⟩.
7:          HALT.
8:  PERFORM "Exhaustive Search" on y.
```

Program 1. The Ability Extractor

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

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

Remark 4. The above proof was informal in that we assumed that in each iteration, the probability that "some *specific* y is chosen by V, and P^* inverts this *specific* y" equals the "expected probability p_x." We are now ready to give a formal proof.

Let $t = n/p_x$, and for $1 \leq i \leq t$, let H_i be a random variable defined as:

$$H_i = \begin{cases} 1 & \text{If the extractor halts in the } i^{\text{th}} \text{ iteration,} \\ 0 & \text{Otherwise.} \end{cases} \tag{11}$$

Note that $H_1, H_2, \ldots, H_t$ are i.i.d samples of a random variable whose expectation is p_x, and let $q_i \stackrel{\text{def}}{=} \Pr[H_i = 0]$ for $i = 1, \ldots, t$.

We want to compute the probability P_8, that Program 1 halts in none of t iterations, and thus Step 8 gets to run: $P_8 \stackrel{\text{def}}{=} \Pr\left[\bigwedge_{i=1}^{t}(H_i = 0)\right] = \prod_{i=1}^{t} q_i$. Using the *inequality of arithmetic and geometric means*, we have: $P_8^{1/t} \leq \frac{1}{t}\sum_{i=1}^{t} q_i$. Due to the *law of large numbers*, the RHS tends to $1 - p_x$ as $t \to \infty$. Therefore, for large enough t's, it is safe to assume that $P_8 \leq (1-p_x)^t = (1-p_x)^{n/p_x} < e^{-n}$, where the last inequality is deduced as above.

The astute reader notices that the above analysis might not work for small n's. However, the existential quantifier in Definition 4 guarantees the existence of some K which is able to invert y for all small n's. ■

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 warn that Goldreich and Oren [Ore87, GO94] proved a negative result:

Theorem 2. *Let L be a language for which there exists a* ***two-step*** auxiliary-input ($\equiv$ universal-simulation) *ZK proof of membership with negligible error. Then $L \in \mathcal{BPP}$.*[9]

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 (reps. proofs of computational ability) as well. That is, Let $\langle V \leftrightarrow P \rangle$ be a **two-step** *auxiliary-input* ($\equiv$ *universal-simulation*) ZK proof of knowledge (resp. proof of computational ability) with negligible error on some binary relation 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). But, 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].[10] 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.

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

[10] 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 adapted 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 security parameter of the system (e.g. the length of the common input). Symbolically, let $\mathbb{F}_\ell$ be the collection mentioned above: $\mathbb{F}_\ell \stackrel{\text{def}}{=} \left\{ f\colon \{0,1\}^* \to \{0,1\}^{\ell(n)} \right\}$. 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). We describe the three variants briefly, and justify the variant we adapt 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.

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 adapted 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.[11]

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

$$\begin{aligned}\forall c>0,\ \exists n_0\in\mathbb{N}\quad s.t.\quad \forall n\geq n_0,\ \forall x\in L\cap\{0,1\}^n,\ \forall r,\xi\in\{0,1\}^{t_{V^*}(n)}\\ \Big|\mathbb{E}\Big[\mathcal{O}\leftarrow_R\mathbb{F}_\ell\colon \Pr\left[\mathcal{A}^{\mathcal{O}}\left(\mathbf{view}\langle {V_r^*}^{\mathcal{O}}(\xi),P^{\mathcal{O}}\rangle(x)\right)=1\right]-\\ \Pr\left[\mathcal{A}^{\mathcal{O}}\left(M_{V^*}^{\mathcal{O}}(x,\xi,r)=1\right)\right]\Big]\Big|<n^{-c}\ .\end{aligned}\tag{12}$$

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 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

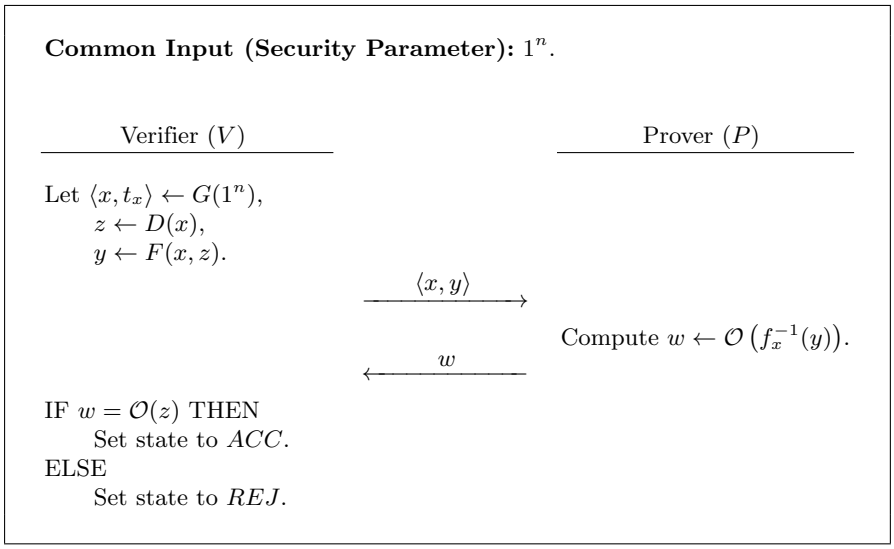
We are now prepared to give a ZK implementation of the Base Protocol. We adapted Variant 1 for some technical reasons.

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 5. 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 state of V.

[11] 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).



Protocol 2. Alleged ZK Implementation of the Base Protocol.

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

$$\chi_R\left(\langle a_1, \ldots, a_k \rangle\right) = \begin{cases} 1 & \text{if } \langle a_1, \ldots, a_k \rangle \in R, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Relation R is **hard to decide** *if for every efficient algorithm S:*

$$\Pr\Big[\forall \langle a_1, \ldots, a_k \rangle \in (\{0,1\}^*)^k \\ S\left(\langle a_1, \ldots, a_k \rangle\right) = \chi_R\left(\langle a_1, \ldots, a_k \rangle\right)\Big] = 0 \ . \tag{14}$$

where the probability is taken over the internal coin tosses of S.

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

$$\forall c > 0, \ \exists n_0 \in \mathbb{N} \quad s.t. \quad \forall n \geq n_0 \\ \Pr\left[\langle a_1, \ldots, a_k \rangle \leftarrow S(1^n) \colon \mathcal{A}\left(\langle a_1, \ldots, a_k \rangle\right) = \chi_R\left(\langle a_1, \ldots, a_k \rangle\right)\right] \\ < \frac{1}{2} + n^{-c} \ . \tag{15}$$

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 probabilities in (14) *and* (15) *are taken over the random selection of $\mathcal{O}$ as well.* □

```
0:    LET ⟨x, t_x⟩ ← G(1^n).
1:    LET O ←_R F_ℓ.
2:    LET z ← D(x).
3:    LET y ← F(x, z).
4:    LET ρ ←_R {0,1}.
5:    IF ρ = 0 THEN
6:        LET r ← O(z).
7:    ELSE
8:        LET r ←_R {0,1}^{ℓ(|x|)}.
9:    LET d ← A^O(x, ⟨y, r⟩).
```

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

We are now ready to state and prove Theorem 3.

Theorem 3. *For every x drawn according to $G(1^n)$, the relation $\hat{R}_x$ (defined at the beginning of this section) is*:

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

We assume that x is given as an auxiliary input to algorithms S and $\mathcal{A}$ of Definition 11.

Proof. **Part (i)**— Let $\mathcal{A}$ be any efficient program, and consider the experiment given in Program 2, where Steps 1–8 represent the program of algorithm S. 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.

Lemma 1. *For large enough x's,* $\Pr[E_1] = \text{negl}(|x|)$.[12]

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 $t_{\mathcal{A}}(|x|)$ 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.

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

```
1:   LET Ψ ← ∅.
2:   FOR i ← 1 TO t_A(|x|) DO
3:       SINGLE-STEP A(x, ⟨y, r⟩).
4:       IF A halts THEN
5:           HALT.
6:       ELSE IF A queries O THEN
7:           LET q be the query.
8:           IF y = F(x, q) THEN
9:               OUTPUT q.
10:              HALT.
11:          ELSE IF q ∈ dom(Ψ) THEN
12:              ANSWER query with Ψ(q).
13:          ELSE
14:              LET ψ ←_R {0,1}^{ℓ(|x|)}.
15:              LET Ψ ← Ψ ∪ {(q, ψ)}.
16:              ANSWER query with ψ.
```

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

Remark 6. 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 is in contradiction to 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 $\mathcal{A}$, Program 2 is indistinguishable from 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, $\Pr[E_2 \mid {E_1}'] = \frac{1}{2}$. ■

```
0:    LET ⟨x, t_x⟩ ← G(1^n).
1:    LET 𝒪 ←_R 𝔽_ℓ.
2:    LET z ← D(x).
3:    LET y ← F(x, z).
4:    LET ρ ←_R {0, 1}.
5:    IF ρ = 0 THEN
6:        LET r ← U_ℓ.
7:    ELSE
8:        LET r ← U'_ℓ.
9:    LET d ← 𝒜^𝒪 (x, ⟨y, r⟩).
```

Program 4. A modified version of Program 2.

Given lemmas 1 and 2, inequality (16) holds.

$$\begin{aligned}\frac{1}{2} = \Pr[E_2 \mid {E_1}'] &= \frac{\Pr[E_2 \cap {E_1}']}{\Pr[{E_1}']} \\ = \frac{\Pr[E_2 - E_1]}{\Pr[{E_1}']} &= \frac{\Pr[E_2] - \Pr[E_1 \cap E_2]}{\Pr[{E_1}']} \\ \geq \frac{\Pr[E_2] - \Pr[E_1]}{\Pr[{E_1}']} &= \frac{\Pr[E_2] - \mathrm{negl}(|x|)}{1 - \mathrm{negl}(|x|)} \ . \end{aligned} \tag{16}$$

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

Part (ii)—Since we proved in Part (i) that no efficient algorithm $\mathcal{A}$ can decide the output of S with probability better than $\frac{1}{2} + \mathrm{negl}(|x|)$, the probability that any efficient algorithm $\mathcal{A}$ can decide *all* tuples correctly is:

$$\lim_{\alpha \to \infty} \left(\frac{1}{2} + \mathrm{negl}(|x|) \right)^{\alpha} = 0 \ . \tag{17}$$

and Part (ii) follows. ■

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.

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

1: LET $y \leftarrow \mathcal{V}^*(x)$.
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 $\langle x, y, \Psi(y) \rangle$.

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

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.

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*[13] $\hat{R} \notin \mathcal{BPP}^{\mathcal{O}}$.

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

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?

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

1: $\langle x, t_x \rangle \leftarrow G(1^n)$, $z \leftarrow D(x)$, $y \leftarrow F(x, z)$.
2: OUTPUT $\langle 1^n, r, x, y, \mathcal{O}(z) \rangle$.

Program 6. The HVCZK Simulator.

To answer the above question, we first note that Protocol 2 is HVCZK. The corresponding HVCZK simulator is shown in Program 6. The simulated 'r' is a prefix of the simulator's random tape, used by efficient algorithms G, D, and (possibly) F. The intersected reader can easily prove that no efficient distinguisher can tell the real and simulated views apart.[14]

In contrast to the honest-verifier case, we were not able to prove the ZK property for general verifiers using "classical" ZK definitions. In particular:

- Old techniques such as *rewinding* the verifier does not help the simulator out of the situation. This is because, in contrast to many proposed ZK protocols, Protocol 2 is not a repetition or composition of some *primitive protocol*.
- The simulator cannot simulate the view by merely *monitoring* the verifier's queries. For instance, consider a dishonest verifier which refuses to make any query to $\mathcal{O}$.[15]

As a side note, if we assume that Protocol 2 is ZK in the *classical* auxiliary-input sense, then Theorem 5 holds.

Theorem 5. *If Conjecture 1 holds, Protocol 2 will be a* non-contrived[16] *example of a protocol secure* ($\equiv ZK$) *in the RO Model which, for no implementation of the random oracle in the Standard Model will remain secure* ($\equiv ZK$).

Proof. Conjecture 1 states that two-step auxiliary input ZK proofs of computational ability do not exist in the Standard Model. Since we assumed that Protocol 2 is ZK in the *classical* auxiliary-input sense, and since it is two-step, any implementation of the random oracle using a single function or a function ensemble will not be ZK in the Standard Model. ■

In the rest, we assume that Protocol 2 cannot be proven ZK in the *classical* auxiliary-input sense. In other words, we assume that no efficient simulator satisfies the requirement of Definition 10. Please note that even if this assumption is wrong, our proof techniques do not invalidate. In fact, we will provide a

[14] In fact, a stricter argument can be proven: The two views are identically distributed. Therefore, the protocol is honest-verifier PZK (HVPZK).

[15] Obviously, such verifier is unable to verify provers response. But the ZK property has nothing to do with that.

[16] Canetti, Goldreich, and Halevi [CGH98, CGH04, CGH08] proved this theorem for *contrived* protocols.

paradigm which will ease proving the ZK property when finding proper simulator seems hard or even impossible.

Let us begin with an intuitive analysis. 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). Obviously, Protocol 2 is ZK for the honest verifier V, since V can compute $w = \mathcal{O}(z)$ all by itself. This is due to the fact that V "knows" z when sending y to P.

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 its has no advantage over a passive observer; and by Theorem 4, we know that the protocol is CZK for passive observers.

We intuitively argued that Protocol 2 is ZK for the extreme ends of the spectrum of verifiers. However, is it true for verifiers in the middle of the spectrum? That is, what if a (dishonest) verifier has some *partial* knowledge about z?

To answer the above question, we have to formalize the concept. For any efficient algorithm $\mathcal{E}$, assume that $\mathcal{E}_\rho$ denotes the algorithms $\mathcal{E}$, when its random input is ρ. (Thus, $\mathcal{E}_\rho$ is *deterministic* on its input.)

For all $x \in X$, and every efficient (possibly dishonest) verifier V (of Protocol 2), and for every $r, \xi \in \{0,1\}^{t_V(|x|)}$, let $\mathcal{V}_{r,\xi}(x)$ denote the `next-message` function of V on common input x, random input r, and auxiliary input ξ. Similarly, let $\mathcal{P}(x,y)$ be the `next-message` function of the (honest) prover P on common input x and on $y \leftarrow \mathcal{V}_{r,\xi}(x)$. (The random input, if any, is implicit in this function.)

Let M be the simulator in Program 5, with two differences (x, r, ξ, and y are the inputs to M):

1. Step 1 is no longer needed.
2. In Step 5, the output of M includes function T of Definition 10. M sets this function to $\varnothing$, since as we will see, M is not required to set this function at any specific point. This way, the ROC(T) generates some truly random oracle $\mathcal{O}$.

Now consider the experiment in Program 7. This experiment 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.[17]

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 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 *forth* input is $\mathcal{O}(z)$, and 0 otherwise. Thus, for the honest verifier and such $\mathcal{A}$, $\Delta_{\mathcal{A}} = 1$. On the other

[17] 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.

```
1:    LET ρ ←_R {0,1}.
2:    LET y ← V_{r,ξ}(x).
3:    LET O ← F_ℓ.
4:    IF ρ = 1 THEN
5:        LET w ← P(x, y).
6:    ELSE
7:        LET w ← M(x; r, ξ, y).
8:    LET d ← A^O (x, r, ξ, w).
```

Program 7. An experiment used to prove Theorem 7.

hand, if V is the dishonest verifier described in the informal analysis above, we have $\Delta_{\mathcal{A}} \leq \frac{1}{2} + \text{negl}(|x|)$ for every efficient algorithm $\mathcal{A}$ and large enough x'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 $\delta_{\mathcal{A}} \stackrel{\text{def}}{=} 2\Delta_{\mathcal{A}} - 1$.

Theorem 6. *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 (16): $\Pr[E_1] \geq 2\Delta_{\mathcal{A}} - 1 = \delta_{\mathcal{A}}$. ■

Let $\mathcal{A}$ be an efficient machine which queries $\mathcal{O}$ at z with advantage $\delta_{\mathcal{A}}$. Assume that $\mathcal{A}$ makes $k \geq 0$ queries to $\mathcal{O}$. 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 no more than 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 $\mathcal{A}'$. Thus, in the rest, we 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 7.

```
1:  ESTIMATE δ_A.
2:  FOR i ← 1 TO |x|/δ_A DO
3:      LET ρ_1 ← {0,1}^{t_A(|x|)}.
4:      LET ρ_2 ← {0,1}^{t_M(|x|)}.
5:          PROCEED A_{ρ_1}(x, r, ξ, M_{ρ_2}(x; r, ξ, y)).
6:          IF A_{ρ_1} queries O THEN
7:              OUTPUT the query.
8:              HALT.
9:  PERFORM "Exhaustive Search" on y.
```

Program 8. The Universal Knowledge Extractor K of Theorem 7.

Theorem 7 (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 x, r, ξ, and y as described above, if $\delta_{\mathcal{A}} > 0$, then $K^{\mathcal{A}}(x; r, \xi, y)$ outputs $z = f_x^{-1}(y)$ in expected time upper-bounded by $\frac{|x|^c}{\delta_{\mathcal{A}}}$.*

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 (which is analyzed in Appendix **??**). 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 6, $\mathcal{A}$ queries $\mathcal{O}$ at point z with probability at least $\delta_{\mathcal{A}}$, 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 $(1 - \delta_{\mathcal{A}})^{\frac{|x|}{\delta_{\mathcal{A}}}} < e^{-|x|}$.

Consequently, since Step 9 gets to run with negligible probability, the expected running time of Program 8 is at most $\frac{|x|}{\delta_{\mathcal{A}}} O(1)$, as required by Theorem 7. ■

Note that, since we only used the `next-message` function of the verifier (to get $y = \mathcal{V}_{r,\xi}(x)$ for M), we have effectively showed that Protocol 2 is a *black-box* CZK proof.

5.1 The Simulation-Extraction Paradigm for Proving the ZK Property

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.
(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.

It is now trivial to formulate a relaxed paradigm for proving the ZK property:

1. Formally state the knowledge that may be carried from the prover to the verifier.[18]
2. Describe an efficient algorithm for the simulator M.
3. If no efficient algorithm $\mathcal{A}$ can distinguish the real and simulated views with non-negligible probability, the protocol is zero-knowledge in the "classical" sense.
4. Otherwise, prove that for every efficient algorithm $\mathcal{A}$, if $\mathcal{A}$ can distinguish the real and simulated views with advantage $\delta_{\mathcal{A}}$, then, there exists a universal knowledge extractor which uses $\mathcal{A}$ as an oracle and, on input the *simulated view*, can extract the knowledge stated in clause §1 in expected time $\frac{\mathrm{poly}(n)}{\delta_{\mathcal{A}}}$ (n is the security parameter).

The above 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][19]).

6 Conclusions 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:

[18] For Protocol 2, the knowledge is the string z.
[19] Note that we are stating nothing about the validity of the theorem itself. We are merely pointing that the proof becomes invalid.

1. 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 relaxed paradigm?
2. We assumed that Protocol 2 does not have a classical zero-knowledge simulator. Is this true? If not, does it give rise to Conjecture **??**?
   In addition, can we conceive of protocols other than Protocol 2, which are not zero-knowledge in the classical sense but are zero-knowledge in our paradigm?
3. 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, Mic01].

We encourage the reader to engage in research to solve these problems.

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.

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

[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*, pages 106–115. IEEE, 2001.

[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.

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ 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 283–298. Springer-Verlag, 1998. Extended version available from `http://cseweb.ucsd.edu/~mihir/papers/tf.html`.

[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.

[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`.

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

[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.

[FS87] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology—CRYPTO '86*, pages 186–194, Santa Barbara, California, United States, 1987. Springer-Verlag.

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

[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 of 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 of Computing*, 18(1):186–208, 1989.

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

[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.

[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.

[Mic01] Silvio Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2001.

[Nie02] Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *Advances in Cryptology—CRYPTO '02*, 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 '03*, pages 316–337. Springer-Verlag, 2003.

[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.