# Round-Optimal Zero-Knowledge Arguments Based on Any One-Way Function

Mihir Bellare[1] and Markus Jakobsson[2] and Moti Yung[3]

[1] Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir.

[2] Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-mail: markus@cs.ucsd.edu.

[3] CertCo, New York, NY, USA. E-mail: moti@certco.com.

**Abstract.** We fill a gap in the theory of zero-knowledge protocols by presenting NP-arguments that achieve negligible error probability and computational zero-knowledge in four rounds of interaction, assuming only the existence of a one-way function. This result is optimal in the sense that four rounds and a one-way function are each individually necessary to achieve a negligible error zero-knowledge argument for NP.

## 1 Introduction

In a zero-knowledge (ZK) protocol, a prover $P$ wants to "convince" a verifier $V$ that some claim is true, without "revealing" any extra information [GMR]. In the theory of ZK protocols, researchers have looked at the complexity assumptions based on which protocols can be constructed, and the resources necessary to do so. Here we fill a gap in this area. Let us begin by explaining the various dimensions of such protocols.

### 1.1 The big picture

The interaction between $P$ and $V$ takes place on some common input $x$, and $P$ is trying to convince $V$ that $x$ belongs to some underlying language $L$. The length of $x$ is denoted $n$ and one measures complexity in terms of $n$. The verifier is always a (probabilistic) polynomial time algorithm. Typically (and here) $L$ is in NP. The system has two dimensions: "conviction" and "zero-knowledge." Each can be formalized in one of two ways, a weak and a strong, depending on whether or not we restrict the adversary involved to polynomial time. To describe these dimensions, we use a terminology from [BCY] (which they credit to Chaum).

Degrees of conviction. Conviction is about "soundness." If $x \notin L$ we ask that no matter how the prover behaves, it cannot convince $V$ to accept, except with low probability (called the error probability, and denoted $\epsilon(\cdot)$). This has been formalized in two ways:

- **Statistical conviction**: This is the notion of [GMR]. Even a computationally unrestricted prover should be unable to make the verifier accept $x \notin L$, except with probability $\epsilon(n)$. Protocols providing this strong degree of conviction are usually called "proofs."
- **Computational conviction**: This is the notion of [BrCr, BCC]. A prover restricted to (randomized) polynomial time should be unable to make the verifier accept $x \notin L$, except with probability $\epsilon(n)$.[4] (But a more powerful prover might succeed in making the verifier accept with high probability.) Although weaker, this kind of soundness is good enough for cryptographic protocols. The soundness will typically depend on the assumed intractability of some computational problem, like factoring or computing discrete logarithms. Protocols meeting this condition are usually called "arguments."

DEGREES OF ZERO-KNOWLEDGE. Roughly, the zero-knowledge condition of [GMR] asks that when $x \in L$, the transcript of an interaction between the prover and a verifier yield no information (other than the fact that $x \in L$) to an adversary who gets to examine the transcript. Again, this adversary may be weak or strong:

- **Statistical ZK**: Even a computationally unrestricted adversary will not get useful information out of a transcript, except with low (negligible) probability. Protocols meeting this are usually called SZK.
- **Computational ZK**: A (randomized) polynomial time adversary will not get useful information out of a transcript. (But a computationally unrestricted adversary might.) This will be the case when the transcript contains encryptions of sensitive data, which are useless to a polynomial time adversary, but can be opened by an unrestricted one. This type of ZK is usually called CZK and, although weaker, is good enough for cryptographic protocols.

We clarify that this discussion is very informal. The definitions talk of the indistinguishability of ensembles. (See Section 2.4.) We also don't make perfect ZK a special case, considering it included as a sub-case of statistical.

A NOTE ON COMPLETENESS. In addition, a basic completeness condition is always required. It asks that if $x \in L$ then there is a strategy via which the prover can make $V$ accept. The definition of [BrCr, BCC] asks (as appropriate for a cryptographic protocol) that this be efficiently achievable: if $P$ is given a witness for the membership of $x$ in the NP language $L$ then it can make $V$ accept in polynomial time. The definition of [GMR] does not make such a requirement. However, all known proofs (statistically convincing) for NP languages do meet this efficient completeness requirement, so we won't discuss it further, assuming it always to be true.

A NOTE ON PROOFS OF KNOWLEDGE. One usually also wants that when $x \in L$, the ability of a prover to convince $V$ to accept should be indicative of "knowledge" of a witness. Like soundness, in proofs it holds for arbitrary provers and in arguments for polynomial time ones. (The notion was suggested in [GMR],

[4] This description masks some subtleties. See Definition 2 and the following discussion.

and an appropriate formalization has emerged in [BeGo]. See Section 2.3 for more.) Again, we will not discuss it further here, concentrating just on the two dimensions mentioned above.

FOUR KINDS OF PROTOCOLS. Since the dimensions discussed above are orthogonal, we get four kinds of protocols:

- CZK arguments: *Computationally convincing, computational ZK.* The weakest kind, but still adequate for cryptographic protocols. For example the arguments for all of NP in [BrCr, BCC] when a standard bit commitment is used.
- CZK proofs: *Statistically convincing, computational ZK.* For example the proofs for all of NP in [GMW].
- SZK arguments: *Computationally convincing, statistical ZK.* For example the arguments for all of NP in [BrCr, BCC] when a discrete logarithm based bit commitment is used; also [NOVY].
- SZK proofs: *Statistically convincing, statistical ZK.* The strongest kind, but not possible for all of NP unless the polynomial time hierarchy collapses [Fo]. But there are examples for special languages: quadratic residuosity and its complement [GMR]; graph isomorphism and its complement [GMW]; constant round SZK proofs for quadratic residuosity and graph isomorphism [BMO1].

## 1.2 Complexity measures and optimality

Recall that the error-probability is the probability $\epsilon(\cdot)$ in the soundness condition, whether in a proof or an argument. Most atomic ZK protocols have constant error. But one really wants low error. A standard goal is to make the error negligible. (That is, a function vanishing faster than the reciprocal of any polynomial.) We will have the same goal.

COMPLEXITIES TO MINIMIZE. Theoretical research in ZK proofs has focused on achieving this low error while trying to minimize other complexity measures. Two main ones are:

- Rounds: The round complexity is the number of messages exchanged, or rounds of interaction in the protocol.[5]
- Assumptions: The complexity assumption underlying the protocol, it underlies either the computational ZK or the computational conviction (or both). For example it may be an algebraic assumption like the hardness of factoring or discrete log computation, or a general assumption like the existence of claw-free pairs, trapdoor permutations, one-way permutations, or one-way functions.

[5] There may be some danger of confusion in terminology. We call each sending of a message by a party a round. Some works like [FeSh] call this a move, and say a round is two consecutive moves. In their terminology, our four round protocols would be four move or two round protocols.

| Rounds | Assumption | Reference | Type |
|---|---|---|---|
| poly($n$) | One-way function | Combine [GMW, HILL, Na] | CZK proof |
| $\omega(\log n)$ | Algebraic | [BrCr, BCC] | SZK argument |
| poly($n$) | One-way permutation | [NOVY] | SZK argument |
| 6 | Claw-free pairs | [BCY] | SZK argument |
| 6 | Claw-free pairs | [GoKa] | CZK proof |
| 5 | One-way function | [FeSh] | CZK argument |
| 4 | Algebraic | [FeSh] | CZK argument |
| 4 | Trapdoor perm. + Algebraic | Combine [Bl, FLS, BeYu] | CZK argument |
| 4 | One-way function | This paper | CZK argument |

**Fig. 1.** *Negligible error ZK protocols for NP.* We list round complexity, complexity assumption used, and type (CZK or SZK, proof or argument). Remember four rounds is optimal.

LOWER BOUNDS. We know that things can't go too low. Four rounds and a one-way function are each individually necessary to get low-error ZK:

- Four rounds needed: Goldreich and Krawczyk [GoKr] show that there do not exist *three round, negligible error* (whether proof or argument) ZK (whether computational or statistical) protocols for NP unless NP $\subseteq$ BPP. (There is a technical condition saying the ZK must be of a certain form called black-box. But all known ZK protocols are of this type. In this paper whenever we talk of ZK we always mean black box. See Definition 6.) Accordingly, four is the minimal number of rounds required to achieve ZK with low error. (The result also holds if the protocol is not sound but just a proof of knowledge, so that four rounds is also necessary for negligible knowledge error [IS1].)
- One-way function needed: ZK arguments can be used to implement many kinds of cryptographic schemes, whence by [ImLu] require a one-way function to implement. Even for the proof case with a computationally unbounded prover, it is known that for "hard" languages some kind of "one-way function" is necessary [OsWi]. Thus, a one-way function is a minimal assumption required to achieve ZK.

THE PROBLEM. There are many so-called "atomic" ZK protocols for NP that achieve constant error-probability in constant (three or four) rounds. Serial repetition lowers the error and preserves ZK [GoOr, ToWo], but at the cost of increasing the number of rounds to non-constant. So we would like to do parallel repetition. However, this is ruled out: first, we have the above mentioned results of [GoKr]; second, the latter also showed that in general parallel repetition does not preserve ZK. So one must build low error ZK protocols directly.

PREVIOUS WORK. A good deal of effort has gone into this, and a variety of ingenious constructions have been proposed. We summarize the known results in Figure 1. (One that may need elaboration is the protocol of [Bl, FLS, BeYu]. We discuss it briefly in Appendix A.)

Notice that prior to our work optimality had not been achieved in any protocol category. That is, neither for CZK arguments, SZK arguments or CZK proofs did we have four round, low error protocols based on any one-way function. In this paper we have filled the first of these gaps.

We also clarify that we are only tabulating ZK protocols for all of NP (ie. for NP-complete languages). There is also a lot of work on constant round ZK (especially statistical ZK) for special languages which we don't get into.

## 1.3 Our result

RESULT. We look at low error CZK arguments for all of NP. Figure 1 tells us that it is possible to do it in four rounds using an algebraic assumption (hardness of discrete log) [FeSh]; or in five rounds using a one-way function [FeSh]. This leaves a (small but noticeable) gap, which we fill: we provide an optimal protocol, that uses only four rounds and a one-way function.

**Theorem 1.** *Suppose there exists a one-way function. Then for any language in NP, there exists a protocol which has four rounds of interaction; is computationally convincing (ie. an argument) with negligible error probability; is computational zero-knowledge; and is a computational proof of knowledge (for the underlying NP-relation) with negligible knowledge-error.*

TECHNIQUES. Our protocol is for the NP-complete language $SAT$. Let $\varphi$ be the input formula. We use the idea of Feige and Shamir [FeSh] of ORing to $\varphi$ some formula $\Phi$ which represents some choices of the verifier, and then having the prover run a standard ZK proof on input $\Theta = \varphi \vee \Phi$. However, Feige and Shamir [FeSh] begin their protocol by having the verifier give a witness indistinguishable proof of knowledge of something underlying $\Phi$. Instead, we work directly with the one-way function, having the verifier give a cut-and-choose type proof that $\Phi$ meets some conditions. This is interleaved with a standard ZK proof run on $\Theta$. To implement the latter with a one-way function we use Naor's bit commitment scheme [Na] which can be based on a one-way function via [HILL].

The tricky part is getting the protocol to be ZK. When the protocol is finally designed, however, the ZK is not hard to see. It turns out the technically more challenging part is to prove computational soundness. We introduce what seems to be a new technique, proving the soundness by using proofs of knowledge, relying on the strong formulation of the latter given in [BeGo].

## 1.4 Open problems

We have filled the (small) existing gap between upper and lower bounds for CZK arguments. For other protocol categories, the existing gap is larger and still unfilled. For CZK proofs, it is not known whether constant error can be achieved with a one-way function (let alone with what value of the constant). For SZK arguments, it is not known whether it can be done at all (ie. in polynomially many rounds) with a one-way function.

## 2 Definitions

We provide definitions for zero-knowledge arguments and computational proofs of knowledge.

### 2.1 Preliminaries

NP-RELATIONS. Let $\rho(\cdot,\cdot)$ be a binary relation. We say that $\rho$ is an NP-relation if it is polynomial time computable and, moreover, there exists a polynomial $p$ such that $\rho(x,w) = 1$ implies $|w| \leq p(|x|)$. For any $x \in \{0,1\}^*$ we let $\rho(x) = \{ w \in \{0,1\}^* : \rho(x,w) = 1 \}$ denote the witness set of $x$. We let $L_\rho = \{ x \in \{0,1\}^* : \rho(x) \neq \emptyset \}$ denote the language defined by $\rho$. Note that a language $L$ is in NP iff there exists an NP-relation $\rho$ such that $L = L_\rho$. We say that $\rho$ is NP-complete if $L_\rho$ is NP-complete.

The example we will concentrate on is satisfiability. Let $\varphi$ be a boolean formula (circuit) and $T$ an assignment of 0/1 values to its variables. We let $\mathit{Satisfy}(\varphi,T) = 1$ if $T$ satisfies $\varphi$ (makes it true) and 0 otherwise. This is an NP-relation, and the corresponding language $L_{\mathit{Satisfy}}$ is of course just $SAT = \{ \varphi : \varphi \text{ is a satisfiable boolean formula} \}$.

NEGLIGIBILITY. Recall that a function $\delta\colon \mathsf{N} \to \mathsf{R}$ is *negligible* if for every polynomial $p(\cdot)$ there exists an integer $n_p$ such that $\delta(n) \leq 1/p(n)$ for every $n \geq n_p$.

INTERACTIVE ALGORITHMS. Parties in our protocols (provers and verifiers) are modeled as interactive functions. An interactive function $A$ takes input $x$ (the common input), the conversation $M_1 \ldots M_i$ so far, and coins $R$ to output $A(x, M_1 \ldots M_i, R)$, which is either the next message, or some indicator to stop, perhaps accepting or rejecting in the process. Probabilities pertaining to this function are over the choice of $R$. We let $A_x(\cdot,\cdot) = A(x,\cdot,\cdot)$ and $A_{x,R}(\cdot) = A(x,\cdot,R)$. Typically we will have fixed $x$ and will be talking about $A_x$; sometimes we will also have fixed $R$ and are talking about the deterministic function $A_{x,R}$. $A$ may also take an auxiliary input $w$ (when $A$ is the prover, this is a witness $w \in \rho(x)$) and we write $A^w$ for this algorithm. Thus we can have $A^w_x$ or $A^w_{x,R}$.

The transcript of a conversation between a pair of interactive functions is the entire sequence of messages exchanged between them until one of them halts. We let $\mathsf{Acc}(A_x, B_x)$ denote the probability (over the coins of both parties) that $B$ accepts when talking to $A$ on common input $x$. We let $\mathsf{Acc}(A_x, B_x, M_1 \ldots M_i)$ denote the conditional probability that $B$ accepts in talking to $A$ on common input $x$ when the conversation so far is $M_1 \ldots M_i$.

We refer to the sending of a message by one party as a round of interaction. So the number of rounds is the total number of messages sent.

## 2.2 Arguments, or computationally convincing proofs

The protocol must satisfy a standard completeness condition saying that a prover knowing a witness for $x \in L_\rho$ can convince the verifier to accept $x$. Soundness pertains to what happens when $x \notin L_\rho$. We want to say that it is unlikely that one can make the verifier accept, even if one is allowed to modify the strategy of the prover. The error-probability measures how unlikely. For the purpose of this paper we are interested in arguments of negligible error, but the definition that follows is for any error.

**Definition 2.** Let $P, V$ be polynomial time interactive algorithms and let $\rho$ be an NP-relation. We say that $(P, V)$ is a computationally convincing proof (or argument) for $\rho$, with error-probability $\epsilon(\cdot)$, if the following two conditions are met:

(1) Efficient completeness: For every $x \in L_\rho$ and every witness $w \in \rho(x)$ it is the case that $\mathsf{Acc}(P_x^w, V_x) = 1$.

(2) Computational soundness: For every polynomial time interactive algorithm $\widehat{P}$ there is a constant $N_{\widehat{P}}$ such $\mathsf{Acc}(\widehat{P}_x, V_x) \leq \epsilon(|x|)$ for all $x \notin L_\rho$ which have length at least $N_{\widehat{P}}$.

If $\epsilon$ is negligible then we say that the error-probability is negligible.

We highlight the case of negligible error: the system has negligible error as long as there is *some* negligible function $\epsilon(\cdot)$ such that the error is $\epsilon(\cdot)$.

Notice one difference with defining interactive proofs: we ask that the point at which the error goes down to $\epsilon(\cdot)$ depend on the prover $\widehat{P}$. This is necessary, as the discussion below explains.

Issues in computational soundness. In the interactive proof setting [GMR], the error-probability of a protocol $(P, V)$ is $\epsilon(\cdot)$ if for any $x \notin L$ and any interactive algorithm $\widehat{P}$ playing the role of the prover, $\mathsf{Acc}(\widehat{P}_x, V_x) \leq \epsilon(|x|)$. The question of what is the error-probability of a computationally sound proof (argument) is more subtle. The first thought is that we say the same thing, except restricting our attention to polynomial time prover algorithms. Namely, the error-probability is $\epsilon(\cdot)$ if $\mathsf{Acc}(\widehat{P}_x, V_x) \leq \epsilon(|x|)$ for any polynomial time interactive algorithm $\widehat{P}$ and any $x \notin L$. But this is not right. Underlying the argument is some computationally hard problem like inverting a one-way function. The size of this problem is proportional to $|x|$. So for any *fixed* $x$ there is *some* polynomial time prover who can convince the verifier with *high* probability, by solving the underlying computational problem. In other words, we cannot, for a fixed $x \notin L$, hope that the probability of convincing the verifier is at most $\epsilon(|x|)$ for *all* polynomial time provers. (Unless the argument is in fact a proof.) However, for any fixed polynomial time prover, as $|x|$ grows, the probability of convincing the verifier decreases, because the size of the underlying hard computational problem is increasing. In particular it is reasonable to ask that for each $\widehat{P}$ the error eventually goes below the desired error-probability $\epsilon(n)$, which is what we did above.

In particular, the probability of convincing the verifier to accept $x \notin L$ in a computationally convincing proof cannot be reasonably expected to be exponentially small. It is restricted by the probability of solving the underlying computational problem. Since the typical assumption is that the latter is negligible (not but less), the error of the argument too is negligible but not less. In particular, independent repetition will not lower the error to exponentially small.

Another way to resolve the issue is to have a security parameter $k$ that is separate from the input $x$ and measures the size of the underlying hard problem. For any fixed $x$, the error-probability still goes down as we increase $k$. This formulation is probably better for protocol design, but in the current theoretical setting, we stick, for simplicity, to just one input, and adopt the definition above.

## 2.3 Computational proofs of knowledge

We want to say that if an interactive algorithm can convince $V$ to accept $x \in L$ then it must actually "know" a witness $w \in \rho(x)$. This notion of a "proof of knowledge" was suggested in [GMR]. It was formalized in [BeGo] both for the standard interactive proof setting and the argument, or computationally convincing setting. (They discuss the latter in [BeGo, Section 4.7].) We adopt their notion. It comes in two equivalent forms. We present both.

Recall an oracle algorithm $E$ is an algorithm that can be equipped with an oracle. An invocation of the oracle counts as one step. We will talk of an "extractor" $E$ which will be given an oracle for $\widehat{P}_x$, a prover algorithm on input $x$, and will then try to find a witness $w$ to the membership of $x$ in $L_\rho$. The first definition below is what [BeGo] refer to as the "alternative form of validity."

**Definition 3.** [BeGo] We say that verifier $V$ defines a computational proof of knowledge for NP-relation $\rho$, with knowledge-error $\kappa(\cdot)$, if there is a an expected polynomial time oracle algorithm $E$ (the extractor) such that for every polynomial time interactive algorithm $\widehat{P}$ there is a constant $N_{\widehat{P}}$ such that if $x \in L_\rho$ has length at least $N_{\widehat{P}}$ then

$$\Pr\left[ E^{\widehat{P}_x}(x) \in \rho(x) \right] \geq \mathsf{Acc}(\widehat{P}_x, V_x) - \kappa(|x|) .$$

If $\kappa(\cdot)$ is negligible then we say the proof has negligible knowledge-error.

In other words, if $E$ has oracle access to $\widehat{P}$ then it can output a witness for membership of $x$ in $L_\rho$ with a probability only slightly less than the probability that $\widehat{P}$ would convince $V$ to accept $x$. Again, note negligible knowledge error means the above is true for *some* negligible function $\kappa(\cdot)$.

In the next formulation (the main one of [BeGo]) the extractor must find a witness with probability one. It is not limited to (expected) polynomial time, but must run in time inversely proportional to the excess of the accepting probability over the knowledge error.

**Definition 4.** [BeGo] We say that verifier $V$ defines a computational proof of knowledge for NP-relation $\rho$, with knowledge-error $\kappa(\cdot)$, if there is a an oracle algorithm $E$ (the extractor) and a constant $c$ such that for every polynomial time interactive algorithm $\widehat{P}$ there is a constant $N_{\widehat{P}}$ such that if $x \in L_\rho$ has length at least $N_{\widehat{P}}$ and satisfies $\mathsf{Acc}(\widehat{P}_x, V_x) > \kappa(x)$, then $E^{\widehat{P}_x}(x) \in \rho(x)$, and moreover this computation halts in an expected number of steps bounded by

$$\frac{|x|^c}{\mathsf{Acc}(\widehat{P}_x, V_x) - \kappa(x)} .$$

If $\kappa(\cdot)$ is negligible then we say the proof has negligible knowledge-error.

See [BeGo] for the proof that these two notions are equivalent. Sometimes it is convenient to use one, sometimes the other.

### 2.4 Zero-knowledge

ENSEMBLES AND COMPUTATIONAL INDISTINGUISHABILITY. We recall these notions of [GoMi, GMR]. An *ensemble* indexed by $L \subseteq \{0,1\}^*$ is a collection $\{E(x)\}_{x \in L}$ of probability spaces (of finite support), one for each $x \in L$. Let $\mathcal{E}_1 = \{E_1(x)\}_{x \in L}$ and $\mathcal{E}_2 = \{E_2(x)\}_{x \in L}$ be ensembles over a common index set $L$. A ***distinguisher*** is a polynomial sized family of circuits $D = \{D_x\}_{x \in L}$, with one circuit for each $x \in L$. We say that $\mathcal{E}_1, \mathcal{E}_2$ are *(computationally) indistinguishable* if there is a negligible function $\delta(\cdot)$ such that for every distinguisher $D$ there is a constant $N_D$ such that if $x \in L$ has length at least $N_D$ then

$$\left| \Pr\left[ D_x(v) = 1 \ : \ v \xleftarrow{R} E_1(x) \right] - \Pr\left[ D_x(v) = 1 \ : \ v \xleftarrow{R} E_2(x) \right] \right| \leq \delta(|x|) .$$

ZERO-KNOWLEDGE. Let $P, V$ be interactive algorithms. The definition of a zero-knowledge interactive proof [GMR] refers to a language $L$. It begins by defining a probability space, the view of a cheating verifier $\widehat{V}$ in talking to $P$ on input $x \in L$. (And then says there is a simulator that on input $x$ produces an "indistinguishable" view.) The basic idea is the same in the argument setting, but one must be careful about a couple of things. Recall prover $P$ begins with a witness $w$ to $x$. The view generated by $P$ and $V$ depends not just on $P$ but on $w$. An elegant way to bring this into the picture is via the notion of a witness selector [BeYu].

**Definition 5.** [BeYu] A *witness selector* for an NP-relation $\rho$ is a map $W\colon\ L_\rho \to \{0,1\}^*$ with the property that $W(x) \in \rho(x)$ for each $x \in L_\rho$.

That is, a witness selector is just a way of fixing an association of a particular witness to each input. When $\rho = \mathit{Satisfy}$ and $L_\rho = \mathit{SAT}$ this just means associating to any formula $x = \varphi \in \mathit{SAT}$ a particular satisfying assignment to it, out of all the possible satisfying assignments.

Now we can define the view. Let $P, V$ be interactive algorithms, $\rho$ an NP-relation, and $W$ a witness selector for $\rho$. We let $\text{VIEW}(P, W, V, x)$ be the probability space whose points are of the form $(R, \tau)$, where $R$ is a random tape for $V_x$ and $\tau$ is a transcript of an interaction between $P_x^{W(x)}$ and $V_{x,R}$. The associated probability is that over the choice of $R$ and the coins of $P_x^{W(x)}$. The collection $\{\text{VIEW}(P, W, \widehat{V}, x)\}_{x \in L_\rho}$ becomes an ensemble.

We define zero-knowledge in a strong "black-box" simulation form. The simulator $S$ is an oracle algorithm given input $x$ and oracle access to $\widehat{V}_{x,R}$ where $R$ has been chosen at random. (The simulator does not have to pick $R$. It is done automatically and the simulator only sees the interface to the oracle $\widehat{V}_{x,R}$.) It will output a transcript $\tau$ of a conversation between $P_x$ and $\widehat{V}_{x,R}$. We let $\overline{S}^{\widehat{V}_*}(x)$ denote the probability space of pairs $(R, \tau)$ where $R$ was chosen at random and $\tau \leftarrow S^{\widehat{V}_{*,R}}(x)$.

**Definition 6.** We say that $(P, V)$ is a (computational) zero-knowledge protocol for NP-relation $\rho$ if there exists an expected polynomial time oracle algorithm $S$ (the simulator) such that for every polynomial time interactive algorithm $\widehat{V}$ (the cheating verifier) and every witness selector $W$ for $\rho$, the ensembles $\{\overline{S}^{\widehat{V}_*}(x)\}_{x \in L_\rho}$ and $\{\text{VIEW}(P, W, \widehat{V}, x)\}_{x \in L_\rho}$ are computationally indistinguishable.

Note formally, zero-knowledge is no longer a property of the language $L_\rho$ but of the relation $\rho$ itself.

Under this definition of zero-knowledge, we know that any negligible error probability zero-knowledge argument for an NP-complete relation $\rho$ must have at least four rounds, assuming NP is not in BPP [GoKr]. We want to meet this bound given only a one-way function.

REMARK. The above notion of black-box simulation zero-knowledge is stronger than those of [GoOr, GoKr, BMO2] in the following sense. In our notion, the simulator has no control over the coins $R$ of $\widehat{V}_x$: they are automatically chosen (at random) and then fixed. The simulator does not even have direct access to them: it just gets an oracle for $\widehat{V}_{x,R}$. In the notions of [GoOr, GoKr], the simulator could choose these coins as it liked, even try running $\widehat{V}_x$ on many different random tapes. In the notion of [BMO2] it could not choose them, but did have direct access to them, and could try several random tapes. However, since our results are positive, making a more stringent definition only strengthens them. Also, all known zero-knowledge protocols do meet our definition.

For simplicity we do not talk of non-uniform verifiers, but of course the above definition could be extended to include them.

## 3 Building blocks for our protocol

Our protocol uses one-way functions, satisfiability, and a standard bit commitment based atomic ZK protocol for satisfiability.

### 3.1 One-way functions

Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be some length-preserving function. An *inverter* for $f$ is a family $I = \{I_n\}_{n\geq 1}$ where each $I_n$ is a circuit, taking $n$ bit inputs and yielding $n$ bit outputs, and having size at most $p(n)$ for some polynomial $p(\cdot)$. We let

$$\mathsf{Inv}_f^I(n) \;=\; \Pr\left[\, f(x') = y \;:\; x \xleftarrow{R} \{0,1\}^n \,;\; y \leftarrow f(x) \,;\; x' \leftarrow I_n(y) \,\right]$$

denote the probability that $I_n$ successfully inverts $f$ at the point $y = f(x)$, taken over a random choice of $x \in \{0,1\}^n$.

**Definition 7.** Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a polynomial time computable, length-preserving function. We say $f$ is one-way if there is a negligible function $\delta(\cdot)$ such that for every inverter $I$ there is an integer $N_I$ such that $\mathsf{Inv}_f^I(n) \leq \delta(n)$ for all $n \geq N_I$.

Hereafter we fix a one-way function $f$, and the notation $f$ will always refer to this fixed function.

### 3.2 Formulas and satisfiability

We will present ZK arguments for the NP-complete language $SAT$. More precisely let *Satisfy* be the NP-relation defined by *Satisfy*$(\varphi, T) = 1$ if assignment $T$ satisfies formula $\varphi$. The corresponding language $L_{Satisfy}$ is of course $SAT = \{\, \varphi \;:\; \varphi \text{ is a satisfiable boolean formula} \,\}$. We will present ZK arguments for the NP-relation *Satisfy* meeting the definitions in Section 2. (In terms of those definitions, the NP-relation here is $\rho =$ *Satisfy*, the common input is $x = \varphi$, a boolean formula, and the witness $w$ is a satisfying assignment $T$ to $\varphi$.)

We will be encoding statements about the one-way function $f$ as formulas, and need some standard features of the Cook-Levin theorem. The NP-completeness of $SAT$ as proved in this theorem implies the following. There is a polynomial time computable transformation $\textsc{Formula}_f(\cdot)$ such that for any $y \in \{0,1\}^*$ it is the case that $\textsc{Formula}_f(y)$ is a boolean formula which is satisfiable iff there exists an $x \in \{0,1\}^*$ such that $f(x) = y$. More important, there are polynomial time computable maps $t_{f,1}, t_{f,2}$ (called witness transformations) with the following properties. Given $x$, map $t_{f,1}$ outputs a satisfying assignment $T = t_{f,1}(x)$ to $\textsc{Formula}_f(f(x))$. Conversely, given a satisfying assignment $T$ to $\textsc{Formula}_f(y)$, map $t_{f,2}$ outputs a point $x = t_{f,2}(T)$ such that $f(x) = y$. We will refer to both the transformation $\textsc{Formula}_f$ and to the accompanying witness transformations in what follows. What is important to remember is that knowledge of a satisfying assignment $T$ to $\textsc{Formula}_f(y)$ is tantamount to knowledge of a pre-image $x$ of $y$ under $f$.

## 3.3 Naor's commitment scheme

We will use Naor's commitment scheme [Na] which can be based on any one-way function via [HILL]. Some special properties of the scheme are important for us.

It work like this. Suppose $A$ has some data $d \in \{0,1\}^m$ that she wants to commit to $B$. First, $B$ must send $A$ a random string $R$, which we call the *commitment setup* string, and which has length polynomial in the security parameter $n$ and the data length $m$. Then, $A$ picks at random some string $s$ to use as coins, and computes a function $\alpha = \text{COMMIT}_f(R, d, s)$. (This function depends on a pseudorandom bit generator [BlMi, Ya], constructed out of $f$ via [HILL], but we don't need to know that.) This $\alpha$ is $A$'s commitment to $d$ and is sent to $B$. At a later stage, $B$ can ask $A$ to "open" the commitment, at which point $A$ sends $d$ and $s$, and $B$ checks that $\alpha = \text{COMMIT}_f(R, d, s)$.

The protocol must have two properties. First is *privacy:* $\alpha$ gives $B$ no information about $d$. Second is *soundness:* $A$ can't create commitments which she can open in more than one way.

In Naor's scheme [Na], the privacy is true in a computational sense. That is, as long as $B$ cannot invert the underlying one-way function $f$, it gets no partial information about $d$. Soundness however is true in a strong, unconditional sense, and since this is important for us, we need to discuss it further.

A *de-committal* of $\alpha$ is a pair $(d, s)$ such that $\alpha = \text{COMMIT}_f(R, d, s)$. We say that $A$ opens $\alpha$ as $d$ if she provides a de-committal $(d, s)$ of $\alpha$. We say that a commitment setup string $R$ is *bad* if there exists a pair $(d_1, s_1), (d_2, s_2)$ of de-committals of $\alpha$ such that $d_1 \neq d_2$. We say $R$ is *good* if it is not bad. Naor's scheme has the property that a randomly chosen commitment setup string is bad with probability exponentially small in $n$ [Na, Claim 3.1]. For our purposes we set the parameters of the scheme so that this probability is $2^{-2n}$. (The length of $R$ required to make this true depends not only on $n$ but also on the data length $m$. In what follows, we assume $R$ is of the right length to make this true with respect to whatever data length we have.) It follows that the probability that even one out of $n$ random commitment setup strings $R_1, \ldots, R_n$ is bad is at most $n \cdot 2^{-2n} \leq 2^{-n}$. This will be used repeatedly in what follows.

## 3.4 The atomic protocol

We use as a primitive a atomic four round ZK argument achieving error $1/2$. We now specify the properties we want of it and the notation used to describe it. To avoid depending on the details of any specific protocol, it is described via generic components and steps.

THE PROTOCOL. In the literature there are several commitment-based three round ZK arguments with error $1/2$. For concreteness, take the one of Brassard, Crépeau and Chaum [BCC], or the one based on general commitment in [ImYu]. To set it up using one-way function based commitment, we first have the verifier send a commitment setup string, and then run a protocol such as the ones in [BCC, ImYu], so that we have four rounds.

To avoid depending on the details of any specific underlying protocol, we describe the protocol via generic components and steps. Let $\Theta$ denote the boolean formula which is the common input. The prover is assumed to have a satisfying assignment $T$ for $\Theta$. We now specify the instructions for the parties, with the nomenclature to be explained later:

(1) Verifier picks at random a commitment setup string $R$ and sends it to the prover.

(2) Prover picks a random string $\rho$ and computes an encapsulated circuit $C = \textsc{EncCirc}_f(\Theta, T, R, \rho)$. This is sent to the verifier.

(3) Verifier picks a random challenge bit $c$ and sends it to the prover.

(4) Prover computes an answer $D = \textsc{Answer}_f(\Theta, T, R, \rho, c)$ and sends it to the verifier.

(5) Verifier checks that $\textsc{Check}_f(\Theta, R, C, c, D) = 1$. If this is true it accepts, else rejects.

Now let us explain the components. In the second step, the prover computes an object $C$ we call an "encapsulated circuit." This step will involve a number of bit commitments which is proportional to the size of $\Theta$, and they are performed, here, using the scheme of Section 3.3, which can be implemented given $f$. The commitment setup string used (for all the commitments) is $R$, and $\rho$ represents some random choices that underly the encapsulation. (Roughly, the prover will first create a randomized version of $\Theta$ that is annotated with the values given by the truth assignment $T$. This annotated circuit, call it $d$, would reveal $T$, but the prover does not send it directly. Instead, he commits to it, sending $\textsc{Commit}_f(R, d, s)$ where $s$ is part of $\rho$. But the details, such as what is $d$, will not matter: later we will summarize all the properties we need.) As in a typical cut-and-choose protocol, the verifier then poses a random challenge question, which is the bit $c$, and prover must "open" the encapsulated circuit in one of two ways. This "answer" of the prover, denoted $D$, is computed as a function of the truth assignment, the challenge, and the random choices underlying the original encapsulation. It consists of de-committing certain parts of $C$. The answer being sent to the verifier, the latter checks that it is correct. The check is a function of the encapsulated circuit, the commitment setup string, the challenge, and the answer provided.

Properties. We assume certain properties of this protocol. The standard example protocols (eg. [BCC]) do have these properties.

We assume that if an encapsulated circuit $C$ is successfully "opened" in both ways, ie. for both a 0-challenge and a 1-challenge, then one can obtain the truth assignment underlying $\Theta$. This is true no matter how $C$ was constructed, and is the technical fact underlying the protocol being a (computational) proof of knowledge with knowledge error $1/2$.

More precisely, there is a polynomial time algorithm $\textsc{Extract}_f$ such that the following is true. Suppose $R$ is a good commitment setup string. Let $C$ be some string sent by the prover in the first step. (It purports to be a correctly computed encapsulated circuit.) Let $D_0, D_1$ be strings such that $\textsc{Check}_f(\Theta, R, C, 0, D_0) =$

$\text{CHECK}_f(\Theta, R, C, 1, D_1) = 1$. Then $\text{EXTRACT}_f(\Theta, R, C, D_0, D_1) = T'$ is a truth assignment that satisfies $\Theta$.

We stress that this requires the commitment setup string $R$ to be good as defined in Section 3.3. We are using the fact that when this happens, it is impossible (not just computationally infeasible) for the commiter (here the prover) to open a commitment in two different ways.

We will need (to show our protocol is ZK) that one can compute $\text{ENCCIRC}_f(\Theta, T, R, \rho)$ for any $T$, not just a $T$ that satisfies $\rho$. The underlying annotated circuit $d$ will be non-sensical in this case, but the verifier will not know, because the annotated circuit is provided in committed form. (Of course, a prover providing such an encapsulated circuit will be hard put to answer the challenges, but that will not matter for us.)

Finally, of course, we also need that the protocol is ZK. (Actually, all we will use is that it is witness indistinguishable in the sense of [FeSh], something which follows from its being ZK.)

## 4 Protocol 4R-ZK and its properties

We now describe our protocol and its properties. We call the protocol 4R-ZK for "four round ZK."

### 4.1 Protocol description

We give instructions for the prover $P$ and the verifier $V$ to execute protocol 4R-ZK. The common input is a formula $\varphi$ of size $n$, and the prover is assumed in possession of a satisfying assignment $T$ to $\varphi$. Refer to Section 3 for the notation and components referred to below.

**(1)** The verifier's message $M_1 = M_{1,1}M_{1,2}$ consists of two parts computed as we now describe.

- **(1.1)** For $i = 1, \ldots, n$ and $j = 0, 1$ the verifier chooses $x_{i,j} \overset{R}{\leftarrow} \{0,1\}^n$ and sets $y_{i,j} = f(x_{i,j})$. These points are hereafter called the "$Y$-values." It lets $M_{1,1}$ consist of these $2n$ strings.
- **(1.2)** The verifier picks at random commitment setup strings $R_1, \ldots, R_n$. It is thereby initiating $n$ parallel runs of the atomic protocol: $R_i$ will play the role of the commitment setup string for the $i$-th run. (But the input formula $\Theta$ for these runs has however not yet been defined! That will appear later.) It sets $M_{1,2} = (R_1, \ldots, R_n)$.

The verifier **sends** $M_1 = M_{1,1}M_{1,2}$ to the prover. Now for $i = 1, \ldots, n$ and $j = 0, 1$ we let $\Phi_{i,j} = \text{FORMULA}_f(y_{i,j})$ as per Section 3.2. This is a formula both parties can now compute.

**(2)** The prover receives $M_1$. Its reply $M_2 = M_{2,1}M_{2,2}$ consists of two parts computed as we now describe.

- **(2.1)** The prover picks bits $b_1, \ldots, b_n \overset{R}{\leftarrow} \{0,1\}$ and sets $M_{2,1} = (b_1, \ldots, b_n)$. The bit $b_i$ is viewed as selecting the $Y$-value $y_{i,b_i}$, and the verifier is being asked to reveal the pre-image of this value, which he will do in the next step.

**(2.2)** We now set $\Phi = \Phi_{1,1-b_1} \vee \ldots \vee \Phi_{n,1-b_n}$. (This is the OR of all formulas corresponding to $Y$-values which the prover has *not* asked be revealed. As long as $f$ is one-way, the prover has very little chance of knowing a satisfying assignment to $\Phi$.) We then set $\Theta = \Phi \vee \varphi$. Notice that $T$ (the satisfying assignment to $\varphi$ that the prover has) is also a satisfying assignment to $\Theta$, so the prover has a satisfying assignment to $\Theta$ (even though he does not have one for $\Phi$). Viewing $R_1, \ldots, R_n$ as commitment setup strings initiating $n$ parallel runs of the atomic protocol on common input $\Theta$, the prover will now perform the second step for each of these executions of the atomic protocol. Namely, for $i = 1, \ldots, n$ it picks at random a string $\rho_i$ to be used as coins in the encapsulated circuit computation, and computes $C_i = \text{EncCirc}_f(\Theta, T, R_i, \rho_i)$ for $i = 1, \ldots, n$. He now sets $M_{2,2} = (C_1, \ldots, C_n)$.

The prover **sends** $M_2 = M_{2,1}M_{2,2}$ to the verifier.

**(3)** The verifier receives $M_2 = M_{2,1}M_{2,2}$. Its reply $M_3 = M_{3,1}M_{3,2}$ consists of two parts computed as we now describe:

**(3.1)** It sets $M_{3,1} = (x_{1,b_1}, \ldots, x_{n,b_n})$, meaning it returns the pre-images for the $Y$-values selected by the bits $b_1, \ldots, b_n$ that the prover sent in $M_{2,1} = (b_1, \ldots, b_n)$.

**(3.2)** Having $b_1, \ldots, b_n$, the verifier knows $\Phi$ and hence $\Theta$, these formulas being as defined above. It now picks challenges $c_1, \ldots, c_n \overset{R}{\leftarrow} \{0,1\}$, one for each run of the atomic protocol on input $\Theta$, and sets $M_{3,2} = (c_1, \ldots, c_n)$.

The verifier **sends** $M_3 = M_{3,1}M_{3,2}$ to the prover.

**(4)** The prover receives $M_3 = M_{3,1}M_{3,2}$.

**(4.1)** Say $M_{3,1} = (x_1, \ldots, x_n)$. The prover checks that $f(x_i) = y_{i,b_i}$ for $i = 1, \ldots, n$, and if this check fails then it aborts the protocol. Else it goes on to the next step.

**(4.2)** Say $M_{3,2} = (c_1, \ldots, c_n)$. The prover computes the answers to these challenges. Namely for $i = 1, \ldots, n$ it sets $D_i = \text{Answer}_f(\Theta, T, R_i, \rho_i, c_i)$. (Recall $\rho_i$ was the coins used to produce the encapsulated circuit $C_i$, so that here the prover is opening this encapsulated circuit according to challenge $c_i$.)

The prover **sends** $M_4 = (D_1, \ldots, D_n)$ to the verifier.

**(5)** The verifier receives $M_4$ and makes its final check. For $i = 1, \ldots, n$ it checks that $\text{Check}_f(\Theta, R_i, C_i, c_i, D_i) = 1$. (Recall the verifier received the encapsulated circuit $C_i$ in $M_{3,2}$ and the opening $D_i$ in $M_4$.) If this is true it accepts, else it rejects.

Notice that the protocol is indeed of four rounds. Next we address its properties.

## 4.2 Result

Our claims about the above protocol are summarized in the following theorem. Refer to Section 2 for definitions of the various notions.

**Theorem 8.** *Assume $f$ is a one-way function. Then protocol 4R-ZK is:*

(1) *A computationally convincing proof (ie. an argument) with negligible error probability,*

(2) *A computational proof of knowledge with negligible knowledge error, and*

(3) *A (computational) zero-knowledge protocol,*

*all for the NP-relation Satisfy corresponding to the NP-complete language SAT.*

We will prove these items in turn. As one might imagine, the difficulty in the protocol design was making sure it was ZK. Having done the design to make this work out, however, it will be relatively easy to show. The other claims turn out to be more non-trivial. In particular the soundness is shown via a novel use of proofs of knowledge. We begin with a technical lemma that underlies the first two claims above.

## 4.3 The $\Theta$-Extraction Lemma

The first two claims about the protocol are that it is computationally convincing and a computational proof of knowledge. The first says that if $\varphi$ is unsatisfiable then a polynomial time prover has little chance of convincing the verifier to accept, and the second says that if $\varphi$ is satisfiable then any prover convincing the verifier to accept actually "knows" a satisfying assignment to $\varphi$. Both these claims pertain to the input formula $\varphi$. Yet our main technical lemma is a claim not about $\varphi$ but about the formula $\Theta$ constructed in the protocol. Remember this formula (a random variable depending on other choices in the protocol) is the one on which the atomic protocol is actually run. The crucial property of this formula is that (as long as the verifier is honest, namely is $V$) it is *always satisfiable*: whether or not $\varphi$ is satisfiable, $\Theta$ is, because $\Phi$ is always satisfiable.

We claim that if a prover $A$ convinces $V$ to accept $\varphi$ then we can extract a satisfying assignment for $\Theta$, regardless of whether or not $\varphi$ is satisfiable. Furthermore, this extraction can be done to meet the kinds of conditions asked in the definition of [BeGo]. This will help prove both the above mentioned claims, and, as motivation, it may help to say why. Roughly, an assignment to $\Phi$ corresponds to knowledge of inverses of $f$ on random points. But remember $\Theta = \varphi \vee \Phi$. So if $\varphi$ is unsatisfiable, then an assignment to $\Theta$ must be an assignment to $\Phi$, and this will enable us to say in Lemma 10 that significant success in making the verifier accept when $\varphi$ is unsatisfiable translates to inverting the one-way function $f$. On the other hand, if $\varphi$ is satisfiable then an assignment to $\Theta$ will with high probability be one to $\varphi$ since otherwise someone is inverting $f$. Now let us state and prove the lemma.

**Lemma 9.** *There is an expected polynomial time oracle algorithm $E$ (the extractor) such that for any prover $A$ and formula $\varphi$ the following is true. Let $R$ be a random tape for $A_\varphi$ and $M_1M_2M_{3,1}$ a partial transcript of an interaction between $A_{\varphi,R}$ and $V_\varphi$. (The transcript includes the first two messages of the protocol and the first part of $V$'s third message). Assume the commitment setup strings in $M_1$ are good. Let $n = |\varphi|$. Let $p = \mathsf{Acc}(A_{\varphi,R}, V_\varphi, M_1M_2M_{3,1})$ be the probability that $V$ accepts given the current partial transcript. Then on input $\varphi, M_1M_2M_{3,1}$ and with oracle access to $A_{\varphi,R}$, algorithm $E$ outputs a satisfying assignment to the formula $\Theta$ defined by the above partial transcript as in the description of our protocol, and this with probability at least $p - 2^{-n}$.*

*Proof.* Let $\mathbf{R} = (R_1, \ldots, R_n)$ be the sequence of commitment setup strings in $M_1$. We know that $M_2 = (\mathbf{b}, \mathbf{C})$ where $\mathbf{C} = (C_1, \ldots, C_n)$ and $C_i$ is (supposed to be) an encapsulated circuit as per an execution of the atomic protocol on input $\Theta$. Say $\mathbf{c} = (c_1, \ldots, c_n)$ is a challenge vector playing the role of message $M_{3,2}$ in the protocol, and $\mathbf{D} = (D_1, \ldots, D_n) = M_4$ is some response. It is useful to let

$$\textsc{Check}^n_f(\Theta, \mathbf{R}, \mathbf{C}, \mathbf{c}, \mathbf{D}) \;=\; \textstyle\bigwedge_{i=1}^n \textsc{Check}_f(\Theta, R_i, C_i, c_i, D_i)$$

be the final evaluation predicate of our verifier. We first describe a different oracle algorithm $E_1$. It takes the same inputs as $E$ should. It always returns a satisfying assignment to $\Theta$, and this within an expected number of steps bounded by $\mathrm{poly}(n)/(p-2^{-n})$. (We can assume $p > 2^{-n}$ since otherwise there is nothing to show.) Algorithm $E_1$ will sample responses of $A_{\varphi,R}$ for different random challenge vectors $\mathbf{c}$, keeping other information fixed, until it finds a pair of challenge vectors that are accepted by $V$ but are different in at least one component. Namely, repeat the following steps:

(1) Pick $\mathbf{c}_t = (c_{t,1}, \ldots, c_{t,n}) \overset{R}{\leftarrow} \{0,1\}^n$ and let $M^*_{t,3} = M_{3,1} \,.\, \mathbf{c}_t$
(2) Let $\mathbf{D}_t = (D_{t,1}, \ldots, D_{t,n}) \leftarrow A_{\varphi,R}(M_1M_2M^*_{t,3})$
until $\exists\, l, m \in [t]$ such that $\mathbf{c}_l \neq \mathbf{c}_m$ but

$$\textsc{Check}^n_f(\Theta, \mathbf{R}, \mathbf{C}, \mathbf{c}_l, \mathbf{D}_l) \;=\; \textsc{Check}^n_f(\Theta, \mathbf{R}, \mathbf{C}, \mathbf{c}_m, \mathbf{D}_m) \;=\; 1\,.$$

Now let $l, m$ satisfy the halting condition. Let $i \in [n]$ be such that $c_{l,i} \neq c_{m,i}$. By definition of $\textsc{Check}^n_f$ it must be that $\textsc{Check}_f(\Theta, R_i, C_i, c_{l,i}, D_{l,i}) = \textsc{Check}_f(\Theta, R_i, C_i, c_{m,i}, D_{m,i}) = 1$, meaning encapsulated circuit $C_i$ of the atomic protocol has been successfully opened both for a 0-challenge and 1-challenge. But then, we know from the properties of the atomic protocol described in Section 3.4, that we can compute a satisfying assignment for $\Theta$ via $\textsc{Extract}_f(\Theta, R_i, C_i, D_{l,i}, D_{m,i})$. (We use here the assumption, made in the lemma statement, that the commitment setup strings in $M_1$ are good. See Sections 3.3 and 3.4.)

Now we need to analyze the running time of $E_1$. Say $\mathbf{c}$ is good if $\textsc{Check}^n_f(\Theta, \mathbf{R}, \mathbf{C}, \mathbf{c}, \mathbf{D}) = 1$ where $\mathbf{D} = A_{\varphi,R}(M_1M_2M_{3,1} \,.\, \mathbf{c})$. The probability that a random $\mathbf{c}$ is good is $p$ so one is found in expected $1/p$ tries. Another different one is then found in expected $1/(p-2^{-n})$ tries. So the pair is found within $2/(p-2^{-n})$ tries. Each try being $\mathrm{poly}(n)$ time, we have the claimed time bound on the expected running time of $E_1$.

Finally, we need to specify the extractor $E$ claimed in the lemma. We apply a trick used in [BeGo] to prove the equivalence of Definitions 3 and 4. On input $\varphi, M_1 M_2 M_{3,1}$ and with oracle access to $A_{\varphi,R}$, algorithm $E$ produces $M_{3,2}$ as $V$ would (this consists of just picking $n$ random challenges), sets $M_3 = M_{3,1}M_{3,2}$, and runs $A_{\varphi,R}$ to get the response $M_4 = A_{\varphi,R}(M_1 M_2 M_3)$. If the resulting transcript is rejecting (as can be determined by running the verifier's check) then $E$ just aborts. If not, it nonetheless aborts with probability exactly $2^{-n}$. If neither of these aborts happens, it runs $E_1$. Since it runs $E_1$ with probability $p - 2^{-n}$, it finds the satisfying assignment with this probability, and moreover its expected running time is $\text{poly}(n) + (p - 2^{-n}) \cdot \text{poly}(n)/(p - 2^{-n})$ which is $\text{poly}(n)$. ∎

## 4.4 Protocol 4R-ZK is computationally convincing

We will justify the first claim of Theorem 8 by proving the following:

**Lemma 10.** *Assume $f$ is a one-way function. Then protocol 4R-ZK is a computationally sound proof for the NP-relation Satisfy, achieving negligible error-probability.*

We first remark and explain that there is indeed something (non-trivial) to be proven here. Typically, error-reduction is done by (serial or parallel) repetition. Firstly, that's not what we are doing; there is some repetition in the protocol, but the protocol itself does not consist of independently repeating some atomic protocol. Moreover, even when the input $\varphi$ is unsatisfiable, the atomic subprotocols are actually being run on a *satisfiable* formula (namely $\Theta$). So we are not counting on the soundness of the atomic protocol to prove the soundness of our protocol!

As mentioned earlier, our approach is to use proofs of knowledge, and in particular Lemma 9. Let us now provide the proof.

*Proof of Lemma 10.* It is easy to see that the specified polynomial time prover strategy $P$ in 4R-ZK will meet the efficient completeness condition of Definition 2. The issue is to show that computational soundness is achieved, and with the claimed negligible error.

Let us assume protocol 4R-ZK does not have negligible error-probability. As per Definition 2 this means there is no negligible function $\epsilon$ such that 4R-ZK meets the computational soundness condition of Definition 2 with error set to $\epsilon$. We will show this contradicts the assumption that $f$ is one-way.

So we want to show that $f$ is not one-way. As per Definition 7, this means we are given an arbitrary negligible function $\delta$ and must show that there is an inverter $I$ and an infinite set $K$ of integers such that $\mathsf{Inv}_f^I(n) > \delta(n)$ for all $n \in K$. Let us set $\epsilon(n) = \delta(n) \cdot 64n$. This is still a negligible function. So by the above assumption, 4R-ZK does not achieve error-probability $\epsilon$. Hence there exists a polynomial time prover $\widehat{P}$ and an infinite set $F$ of unsatisfiable boolean formulae such that $\text{Acc}(\widehat{P}_\varphi, V_\varphi) \geq \epsilon(|\varphi|)$ for all $\varphi \in F$. Let $K$ be the set of all integers $n$ for which $F$ contains a formula $\varphi$ of length $n$. For each $n \in K$ we fix (arbitrarily) some formula $\varphi_n \in F$. Before describing the inverter $I$ for $f$ we need to isolate certain executions of the interaction between $\widehat{P}_\varphi$ and $V_\varphi$, where $\varphi = \varphi_n$.

GOOD EXECUTIONS. Let $n \in K$ and let $\varphi = \varphi_n$. Let $R$ be a random tape for $\widehat{P}_\varphi$ and $M_1M_2M_{3,1}$ a partial transcript of an interaction between $P_{\varphi,R}$ and $V_\varphi$. (The transcript includes the first two messages of the protocol and the first part of $V$'s third message.) We say that $R, M_1M_2M_{3,1}$ is *good* if the commitment setup string in $M_1$ is good (as defined in Section 3.3) and also $\mathsf{Acc}(P_{\varphi,R}, V_\varphi, M_1 M_2M_{3,1}) \geq \epsilon(n)/2$ (the probability here is only over the choice of the verifier's challenge vector **c**, since all other quantities are fixed). Since $\mathsf{Acc}(\widehat{P}_\varphi, V_\varphi) \geq \epsilon(n)$ it must be that the probability (over $R$ and the coins of $V$ leading to $M_1M_2M_{3,1}$) that $\mathsf{Acc}(P_{\varphi,R}, V_\varphi, M_1M_2M_{3,1}) \geq \epsilon(n)/2$ is at least 1/2. On the other hand the probability that the commitment setup string in $M_1$ is bad is $2^{-n}$ (cf. Section 3.3). So the probability that $R, M_1M_2M_{3,1}$ is good is at least, say, $\epsilon(n)/4$. (This is because we can assume wlog that $\delta(n) = \epsilon(n)/(64n)$ is, say, at least $2^{-n/2}$, whence $2^{-n} \leq \epsilon(n)/2$.) In the sequel we will focus on these good transcript prefixes.

STRUCTURE OF INVERTER. We now describe an inverter $I$ for $f$. The inverter $I$ is a polynomial sized collection of circuits $\{ I_n \ : \ n \geq 1 \}$ as described in Section 3.1. (Meaning there is a polynomial $p_2(\cdot)$ such that the size of $I_n$ is a most $p_2(n)$ for all $n \geq 1$.) We will show that that for all $n \in K$ we have $\mathsf{Inv}_f^I(n) > \delta(n) = \epsilon(n)/(64n)$. $I_n$ has embedded into it the formula $\varphi_n$ (which by assumption is unsatisfiable). The input to $I_n$ is a $n$-bit string $y = f(x)$ where $x$ was chosen at random from $\{0,1\}^n$. $I_n$ wants to output a pre-image of $y$ under $f$. We describe $I_n$ as a randomized algorithm. (The coins can always be later eliminated by using the non-uniformity). Think if $I_n$ as having oracle access to $\widehat{P}_\varphi$ where $\varphi = \varphi_n$. (Meaning it will feed it messages and run it, sometimes "backing it up" and so forth. It implements this by running $\widehat{P}$ as a subroutine with the common input fixed to $\varphi$. It is important here that $\widehat{P}$ is polynomial time). It begins by picking a random string $R$ for $\widehat{P}_\varphi$ and initializing the latter with that.

FIRST MOVE. $I_n$ will mimic the first move of $V$, with a slight twist. It picks $\alpha \xleftarrow{R} [n]$ and $\beta \xleftarrow{R} \{0,1\}$. Then for $i = 1, \ldots, n$ and $j = 0,1$ it does the following: If $(i,j) = (\alpha, \beta)$ then set $y_{i,j} = y$, else pick $x_{i,j} \xleftarrow{R} \{0,1\}^n$ and set $y_{i,j} = f(x_{i,j})$. We let $\Phi_{i,j} = \text{FORMULA}_f(y_{i,j})$ be the boolean formula resulting from applying Cook's theorem to the "$f(\cdot) = \cdot$" relation on input $y_{i,j}$, as described in Section 3.2. Now $I_n$ also picks random strings $R_1, \ldots, R_n$, of appropriate length, as setup strings for the bit commitment to be used in the atomic protocol. It lets $M_1$ consist of the strings $y_{i,j}$ for $i = 1, \ldots, n$ and $j = 0,1$, together with $R_1, \ldots, R_n$. This, thought of as the first message of $V$ to $\widehat{P}_\varphi$, is then "sent" to $\widehat{P}_\varphi$.

SECOND MOVE. $I_n$ runs $\widehat{P}_\varphi$ to get its response $M_2 = \widehat{P}_\varphi(M_1 \ ; \ R)$ to the verifier message $M_1$. This response has the form $M_2 = M_{2,1}M_{2,2}$ where $M_{2,1} = (b_1, \ldots, b_n)$ and $M_{2,2} = (C_1, \ldots, C_n)$. Here $C_i$ is (supposed to be) a committal for a run of the atomic protocol on input $\Theta = \varphi \vee \Phi$, where $\Phi = \Phi_{1,1-b_1} \vee \ldots \vee \Phi_{n,1-b_n}$.

OPENING. Recall that $V_\varphi$ is supposed to return $x_{i,b_i}$ to $\widehat{P}_\varphi$ for all $i = 1, \ldots, n$. $I_n$ would like to do the same. But if $b_\alpha = \beta$ then this means it must return a pre-image of $y_{\alpha,\beta}$ under $f$, and it does not know such a pre-image. (Indeed, the goal of $I_n$ is to find one). So in this case $I_n$ aborts. But this can only happen with probability 1/2 since $\beta$ was a random bit. In case $b_\alpha \neq \beta$, our $I_n$ sets $M_{3,1} = \mathbf{x} = (x_{1,b_1}, \ldots, x_{n,b_n})$. This is the first part of a verifier message $M_3$ to be sent to $\widehat{P}_\varphi$.

FINDING A WITNESS FOR $\Phi$. Now comes the important step. $I_n$ will run an "extractor" for the protocol which consists of $n$ parallel runs of the atomic protocol on input $\Theta$ and find a satisfying assignment for $\Phi$. Specifically, we apply Lemma 9. Let $E$ be as in that lemma and let $p_1(\cdot)$ be the polynomial which is its expected running time. $I_n$ runs $E$ on input $\varphi, M_1M_2M_{3,1}$, giving it oracle access to $\widehat{P}_{\varphi,R}$. However, this execution is halted in $2p_1(n)$ steps. (Recall $E$ has an expected polynomial running time, but $I_n$ needs to halt within a fixed polynomial amount of time.) If $E$ finds, within this time, a satisfying assignment $T$ to $\Theta = \varphi \vee \Phi$, then $I_n$ will be able to find what it wants, namely a point $x$ satisfying $f(x) = y$. The crucial observation is that since $\varphi$ is unsatisfiable, the assignment *$T$ must satisfy $\Phi$*. Hence it must satisfy $\Phi_{i,1-b_i}$ for some $i \in [n]$. Since $\alpha$ was chosen at random from $[n]$ it will be the case that $i = \alpha$ with probability at least $1/n$. We know $b_\alpha \neq \beta$ (since otherwise we aborted above) meaning $b_\alpha = 1 - \beta$. So we have an assignment to $\Phi_{\alpha,\beta}$. Now recall that $\Phi_{\alpha,\beta} = \text{FORMULA}_f(y)$. Applying the witness transformation $t_{f,2}$ discussed in Section 3.2, we can compute a string $x$ such that $f(x) = y$. $I_n$ does this and outputs $x$.

ANALYSIS. The running time of $I_n$ is clearly poly$(n)$. We must analyze its success probability. We assume $R, M_1M_2M_{3,1}$ is good in the sense defined above: we saw this happens with probability at least 1/4. This means the commitment setup strings in $M_1$ are good and $p = \text{Acc}(P_{\varphi,R}, V_\varphi, M_1M_2M_{3,1}) \geq \epsilon(n)/2$. Now Lemma 9 says that $E$ would find a satisfying assignment to $\Theta$ with probability at least $p - 2^{-n} \geq \epsilon(n)/2 - 2^{-n} > \epsilon(n)/4$. (Recall we assumed wlog that $\delta(n) = \epsilon(n)/(64n)$ is at least $2^{-n/2}$, whence the last inequality.) Since we halt $E$ within twice its expected running time, Markov's inequality says we find the assignment with at least half the original probability. So $I_n$ finds $x$ with probability at least $\epsilon(n)/8$. Putting this together with the other probability losses, all together, $I_n$ succeeds with probability at least $\epsilon(n)/(64n) = \delta(n)$, as desired. ∎

## 4.5 Protocol 4R-ZK is a computational proof of knowledge

The second claim of Theorem 8 is justified by the following lemma.

**Lemma 11.** *Assume $f$ is a one-way function. Then protocol 4R-ZK is a computational proof of knowledge (with negligible knowledge error) for the NP-relation Satisfy.*

Before proving it let us discuss the issues. Given a satisfiable formula $\varphi$ and oracle access to a polynomial time prover $\widehat{P}$, the goal is to extract a satisfying assignment to $\varphi$, with a success probability only marginally less than the probability that $\widehat{P}_\varphi$ convinces $V_\varphi$ to accept. We can easily run the extractor of Lemma 9 to find a satisfying assignment $T$, but for $\Theta$, not $\varphi$. But $\Theta = \varphi \vee \Phi$. Our worry is that $T$ satisfies $\Phi$, not $\varphi$. However, intuitively not, because a satisfying assignment to $\Phi$ corresponds to the ability to invert $f$, and thus should appear only with negligible probability. To capture this intuition we must show that were $T$ to satisfy $\Phi$ too often then there would be a way to invert $f$. We can do this similarly to the proof of Lemma 10.

*Proof of Lemma 11.* We will exhibit an extractor $E_1$ such that the conditions of Definition 3 are met for some negligible function $\kappa(\cdot)$. (Recall Definition 3 and Definition 4 are equivalent.) $E_1$ has input satisfiable formula $\varphi$, and has oracle access to $\widehat{P}_{\varphi,R}$ where $R$ is some (randomly chosen and then fixed) random tape for prover $\widehat{P}$. $E_1$ first picks a random a tape $R'$ for $V$. It now plays the role of $V$, invoking $\widehat{P}$ for the role of the prover, and generates a partial transcript $M_1M_2M_{3,1}$ of the interaction between $\widehat{P}_\varphi$ and $V_{\varphi,R'}$. If the commitment setup strings in $M_1$ are not good then $E_1$ aborts. Else it runs the knowledge extractor $E$ of Lemma 9 on input $\varphi, M_1M_2M_{3,1}$, giving it oracle access to $\widehat{P}_{\varphi,R}$. Whatever the latter outputs (hopefully an assignment $T$ to $\Theta$) is what $E_1$ outputs.

Since $E$ runs in expected polynomial time, it is easy to see that $E_1$ does too. Similarly, given Lemma 9, it is easy to see that with probability at least $p-2^{-n+1}$, algorithm $E_1$ outputs a satisfying assignment $T$ to $\Theta$ (not $\varphi$!), where $p = \mathsf{Acc}(\widehat{P}_\varphi, V_\varphi)$. (We loose the additional $2^{-n}$ over the success probability of $E$ because the commitment setup strings are bad with probability at most $2^{-n}$ (cf. Section 3.3) and $E_1$ aborts in this case.)

But our goal is to find a satisfying assignment to $\varphi$. Remember $\Theta = \Phi \vee \varphi$. Our worry is that $T$ satisfies $\Phi$ rather than $\varphi$. Intuitively, however, not, because we know that the ability to find an assignment to $\Phi$ corresponds to the ability to invert $f$. Thus it might happen, but only negligibly often. We must now capture this.

We must show there exists a negligible function $\kappa(\cdot)$ such that $T$ is a satisfying assignment to $\varphi$ with probability $p - \kappa(n)$, for all $\varphi$ of size at least $N_{\widehat{P}}$, where $N_{\widehat{P}}$ is an integer depending on $\widehat{P}$. Assume towards a contradiction that there is no such $\kappa$. So given any negligible function $\kappa$ there is a polynomial time prover $\widehat{P}$ and an infinite set $F$ of formulas such that when $\varphi \in F$, the assignment $T$ output by $E_1$ satisfies $\Phi$ (rather than $\varphi$) with probability at least $(p-2^{-n})-(p-\kappa(n)) = \kappa(n) - 2^{-n}$. We must show that this implies $f$ is not one-way.

We will not give the construction and proof for this last statement in full because the idea is essentially the same as in the proof of Lemma 10. We use the composite of $E_1^{\widehat{P}}$ as an algorithm to construct an inverter for $f$. Like in the proof of Lemma 10, we are given a value $y$ and want to find a pre-image of $y$ under $f$. We put $y$ into the first message of the verifier in the same way as before. Eventually when $E_1^{\widehat{P}}$ gives us an assignment $T$ to $\Phi$, it has some probability of satisfying $\textsc{Formula}_f(y)$ and then we get a pre-image of $y$ under $f$, just as before. The details can be filled in by looking at the proof of Lemma 10. ∎

## 4.6 Protocol 4R-ZK is zero-knowledge

The third claim of Theorem 8 is justified by the following lemma.

**Lemma 12.** *Assume $f$ is a one-way function. Then protocol 4R-ZK is a (computational) zero-knowledge protocol.*

*Proof.* We must specify a simulator $S$ for which Definition 6 is met. $S$ has input $\varphi$ and oracle access to $\widehat{V}_{\varphi,R}$ where $\widehat{V}$ is any (possibly cheating) polynomial time verifier algorithm and $R$ is a randomly chosen random tape for $\widehat{V}_{\varphi}$. It must produce a transcript $\tau$ such that $(R, \tau)$ is distributed like random members of the view of the real interaction between $P_{\varphi}$ and $\widehat{V}_{\varphi}$. Before describing the algorithm let us sketch the intuition.

$S$ will be trying to produce the prover moves in a conversation with $\widehat{V}_{\varphi,R}$. Of course, not knowing a satisfying assignment for $\varphi$, it can't really play the prover. But recall the atomic protocol is run not on input $\varphi$ but on input $\Theta = \varphi \vee \Phi$. The trick is that it suffices to know a satisfying assignment for $\Theta$.

Indeed, suppose we know some satisfying assignment for $\Theta$. This is not necessarily a satisfying assignment for $\varphi$. Still, we can "mimic the prover" by using this assignment in the atomic protocol. The verifier will never know it was not an assignment to $\varphi$, because the proof is ZK and hence witness indistinguishable [FeSh]: views of the verifier for different witnesses held by the prover are indistinguishable.

So if the simulator can find a satisfying assignment to $\Theta$ it can complete a simulation. How can it find one? It can force $\widehat{V}_{\varphi,R}$ to give it one! It will do this by forcing the verifier to reveal a pre-image $x_{i,1-b_i}$ of $y_{i,1-b_i}$ for some $i \in [n]$. This corresponds effectively to a satisfying assignment to $\Phi_{i,1-b_i}$ and hence to a satisfying assignment to $\Phi$ and hence to a satisfying assignment to $\Theta$.

But how does it get $x_{i,1-b_i}$? What $\widehat{V}_{\varphi,R}$ reveals is $x_{i,b_i}$, exactly to prevent the prover from getting $x_{i,1-b_i}$, because if the prover had the latter, it could cheat. But the simulator has an advantage: it can backup the verifier and run it twice for different choices of $b_1, \ldots, b_n$. First it runs it in a normal way on some "dummy" challenges $b'_1, \ldots, b'_n$, gets back the corresponding pre-images, and then claims that the real challenges $b_1, \ldots, b_n$ were different, in particular have $b_\alpha = 1 - b'_\alpha$ for some $\alpha \in [n]$. For the new challenges, it has the pre-image.

Let us now specify all this in full. Here is the algorithm for $S$ with input $\varphi$ and oracle access to $\widehat{V}_{\varphi,R}$:

(1) $S$ runs $\widehat{V}_{\varphi,R}$ to get the first message $M_1 = M_{1,1}M_{1,2}$. Here $M_{1,1}$ consists of strings $y_{i,j} \in \{0,1\}^n$ for $i = 1, \ldots, n$ and $j = 0, 1$, and $M_{1,2} = (R_1, \ldots, R_n)$ consists of $n$ strings to play the role of commitment setup strings. We let $\Phi_{i,j} = \text{FORMULA}_f(y_{i,j})$ be the formula corresponding to $y_{i,j}$ via Cook's theorem, as explained in Section 3.2.

(2) $S$ picks at random $b'_1, \ldots, b'_n \in \{0,1\}$ and lets $\Phi' = \Phi_{1,1-b'_1} \vee \ldots \vee \Phi_{n,1-b'_n}$. It then lets $\Theta' = \varphi \vee \Phi'$ and picks at random an assignment $T'$ to the variables of $\Theta'$. (This assignment is extremely unlikely to satisfy $\Theta'$, but that does

not matter!) For each $i = 1, \dots, n$ it then picks at random some coins $\rho'_i$ and computes an encapsulated circuit $C'_i = \text{EncCirc}_f(\Theta', T', R_i, \rho'_i)$ for $\Theta'$. We let $M'_{2,1} = (b'_1, \dots, b'_n)$ and $M'_{2,2} = (C'_1, \dots, C'_n)$. We view $M'_2 = M'_{2,1}M'_{2,2}$ as the second protocol message (from the prover).

(3) $S$ runs $\widehat{V}_{\varphi,R}(M_1M'_2)$ to get back its response $M'_3 = M'_{3,1}M'_{3,2}$. Here $M'_{3,1}$ consists of values $x_{i,b'_i}$ for $i = 1, \dots, n$ and $M'_{3,2}$ is a challenge vector. $S$ checks that $f(x_{i,b'_i}) = y_{i,b'_i}$ for $i = 1, \dots, n$. If this fails, it outputs the current partial conversation and halts. Else it continues.

(4) $S$ now picks at random another sequence of bits $b_1, \dots, b_n \in \{0,1\}$. If $(b_1, \dots, b_n) = (b'_1, \dots, b'_n)$ then it aborts (but this happens only with probability $2^{-n}$). Else it fixes an index $\alpha \in [n]$ such that $b_i \neq b'_i$. It lets $\Phi = \Phi_{1,1-b_1} \vee \dots \vee \Phi_{n,1-b_n}$ and $\Theta = \varphi \vee \Phi$. Now, notice that $1 - b_\alpha = b'_\alpha$ and $S$ knows $x_{\alpha,b'_\alpha}$, a pre-image of $y_{\alpha,b'_\alpha}$, from the previous step. Because of this, it can compute a satisfying assignment $T$ to the formula $\Phi_{\alpha,b'_\alpha}$. (This is via the properties of Cook's reduction as explained in Section 3.2.) But then $T$ also satisfies $\Phi$ and hence $\Theta$, so $S$ has in its possession a satisfying assignment to $\Theta$. Now the idea is to act like the real prover on input this assignment. (Note this assignment does not satisfy $\varphi$, but the verifier will never be able to tell, because it does satisfy the formula $\Theta$ on which the atomic protocol is performed, and the bit commitments are secure.) So for each $i = 1, \dots, n$ the simulator picks at random some coins $\rho_i$ and computes an encapsulated circuit $C_i = \text{EncCirc}_f(\Theta, T, R_i, \rho_i)$ for $\Theta$. We let $M_2 = M_{2,1}M_{2,2}$ where $M_{2,1} = (b_1, \dots, b_n)$ and $M_{2,2} = (C_1, \dots, C_n)$. We view $M_2$ as a second protocol message (from the prover).

(5) Backing up $\widehat{V}_{\varphi,R}$, the simulator $S$ computes $\widehat{V}_{\varphi,R}(M_1M_2)$ to get back its response $M_3 = M_{3,1}M_{3,2}$. Here $M_{3,1}$ consists of values $x_{i,b_i}$ for $i = 1, \dots, n$ and $M_{3,2}$ is a challenge vector $c_1, \dots, c_n$. $S$ checks that $f(x_{i,b_i}) = y_{i,b_i}$ for $i = 1, \dots, n$. If this check fails $S$ *cannot* abort or output this conversation. (One can check this would lead to an incorrect simulation.) Instead, it must return to Step 4 and try again, continuing this loop until the check does pass. (This is a standard procedure, used for example in [BMO1], and as there one can show that the expected number of tries in this process is at most 2.) So we go on assuming the check did pass.

(6) Having a satisfying assignment $T$ to $\Theta$, the simulator (now in guise of the prover) is able to answer the challenges $c_1, \dots, c_n$ by opening the appropriate parts of the encapsulated circuits $C_1, \dots, C_n$ just as the prover would. Namely $S$ can compute $D_i = \text{Answer}_f(\Theta, T, R_i, \rho_i, c_i)$ for $i = 1, \dots, n$ and let $M_4$ consist of $D_1, \dots, D_n$.

(7) Finally, $S$ can output $\tau = M_1M_2M_3M_4$ as a transcript of the interaction between the prover and $\widehat{V}_{\varphi,R}$.

Fix some witness selector $W\colon SAT \to \{0,1\}^*$ for the relation *Satisfy*$(\cdot,\cdot)$. That is, $W(\varphi)$ is a satisfying assignment to $\varphi$ for every $\varphi \in SAT$. As per Definition 6 we want to show that the probability ensembles $\mathcal{E}_1 = \{\overline{S}^{\widehat{V}_\varphi}(\varphi)\}_{\varphi \in SAT}$ and $\mathcal{E}_2 = \{\text{VIEW}(P, W, \widehat{V}, \varphi)\}_{\varphi \in SAT}$ are computationally indistinguishable. (Refer to Section 2.4 for the definition of $\overline{S}$.) We will do this under the assumption that $f$ is a one-way function. We will provide here only a brief outline of the intuition behind this proof.

The function $f$ shows up in two places in the protocol. First, $f$ is used in the construction of $Y$-values underlying the formula $\Phi$. Second, $f$ underlies the bit commitment scheme of the atomic protocol. The first use of $f$ is not a concern for the zero-knowledge, in the sense that the protocol would be ZK (but not computationally convincing or a computational proof of knowledge!) even if the function used to produce the $Y$-values was not one-way. The ZK depends however on the security of the bit commitment scheme, and hence indirectly on the one-wayness of $f$.

The privacy (cf. Section 3.3) of the bit commitment scheme means that when $S$, in Step (2), forms an encapsulated circuit using a dummy truth assignment $T'$, the verifier $\widehat{V}$ has no feasible way to detect it, and its behavior can change "only negligibly." Now, in Step (4) the simulator uses a satisfying assignment for $\Theta$ that is different from the one the prover would use. But since the atomic protocol is ZK it is also witness indistinguishable in the sense of [FeSh]. Furthermore, they show that witness indistinguishability is preserved under parallel repetition, so the protocol consisting of $n$ parallel repetitions of the atomic protocol is also witness indistinguishable. So the transcripts produced for the two different witnesses in protocol 4R-ZK have (computationally) indistinguishable distributions.

The formal proof would be by contradiction. We assume the ensembles are not computationally indistinguishable. So for any negligible function $\delta(\cdot)$ there is a distinguisher $D = \{D_\varphi\}_{\varphi \in SAT}$ and an infinite set $F$ of satisfiable boolean formulae such that

$$\left| \Pr\left[ D_\varphi(v) = 1 \;:\; v \xleftarrow{R} \overline{S}^{\widehat{V}_\varphi}(\varphi) \right] - \Pr\left[ D_\varphi(v) = 1 \;:\; v \xleftarrow{R} \text{VIEW}(P, W, \widehat{V}, \varphi) \right] \right|$$

is at least $\delta(|\varphi|)$ whenever $\varphi \in F$. Using $D$ we would do one of the following. Either construct a polynomial sized circuit family that defeated the privacy of the bit commitment scheme, which would contradict the security of this scheme as proven in [Na, HILL]. Or, build a distinguisher that would contradict the witness indistinguishability of $n$ parallel repetitions of the atomic protocol. We omit these proofs from this abstract. ▌

## Acknowledgments

We thank the (anonymous) referees of Eurocrypt 97 for comments which improved the presentation of the paper.

Mihir Bellare is supported in part by NSF CAREER Award CCR-9624439 and a Packard Foundation Fellowship in Science and Engineering.

## References

[BeGo] M. BELLARE AND O. GOLDREICH. On Defining Proofs of Knowledge. *Advances in Cryptology – Crypto* 92 *Proceedings*, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

[BMO1] M. BELLARE, S. MICALI AND R. OSTROVSKY. Perfect Zero-Knowledge in Constant Rounds. *Proceedings of the* 22nd *Annual Symposium on the Theory of Computing*, ACM, 1990.

[BMO2] M. BELLARE, S. MICALI AND R. OSTROVSKY. The true complexity of statistical zero-Knowledge. *Proceedings of the* 22nd *Annual Symposium on the Theory of Computing*, ACM, 1990.

[BeYu] M. BELLARE AND M. YUNG. Certifying permutations: Non-interactive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, Vol. 9, No. 1, pp. 149–166, Winter 1996.

[Bl] M. BLUM. Coin Flipping over the Telephone. IEEE COMPCON 1982, pp. 133–137.

[BDMP] M. BLUM, A. DE SANTIS, S. MICALI, AND G. PERSIANO. Non-Interactive Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 20, No. 6, December 1991, pp. 1084–1118.

[BlMi] M. BLUM AND S. MICALI. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, Vol. 13, No. 4, pp. 850–864, November 1984.

[BrCr] G. BRASSARD AND C. CRÉPEAU. Non-transitive Transfer of Confidence: A perfect Zero-knowledge Interactive protocol for SAT and Beyond. *Proceedings of the* 27th *Symposium on Foundations of Computer Science*, IEEE, 1986.

[BCC] G. BRASSARD, D. CHAUM AND C. CRÉPEAU. Minimum Disclosure Proofs of Knowledge. *J. Computer and System Sciences*, Vol. 37, 1988, pp. 156–189.

[BCY] G. BRASSARD, C. CRÉPEAU AND M. YUNG. Constant round perfect zero knowledge computationally convincing protocols. *Theoretical Computer Science*, Vol. 84, No. 1, 1991.

[FFS] U. FEIGE, A. FIAT, AND A. SHAMIR. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, Vol. 1, 1988, pp. 77–94.

[FLS] U. FEIGE, D. LAPIDOT, AND A. SHAMIR. Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String. *Proceedings of the* 31st *Symposium on Foundations of Computer Science*, IEEE, 1990.

[FeSh] U. FEIGE AND A. SHAMIR. Witness Indistinguishable and Witness Hiding Protocols. *Proceedings of the* 22nd *Annual Symposium on the Theory of Computing*, ACM, 1990.

[Fo] L. FORTNOW. The Complexity of Perfect Zero-Knowledge. In *Advances in Computing Research*, Ed. S. Micali, Vol. 18, 1989.

[GoKa] O. GOLDREICH AND A. KAHAN. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, Vol. 9, No. 3, 1996, pp. 167–190.

[GoKr] O. GOLDREICH AND H. KRAWCZYK. On the Composition of Zero Knowledge Proof Systems. *SIAM J. on Computing*, Vol. 25, No. 1, pp. 169–192, 1996.

[GMW] O. GOLDREICH, S. MICALI AND A. WIGDERSON. Proofs that yield nothing but their validity or all languages in NP have zero knowledge proof systems. *Journal of the Association for Computing Machinery*, Vol. 38, No. 1, July 1991.

[GoOr] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, Vol. 7, No. 1, 1994, pp. 1–32.

[GoMi] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Computer and System Sciences*, Vol. 28, 1984, pp. 270–299.

[GMR] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. on Computing*, Vol. 18, No. 1, pp. 186–208, February 1989.

[HILL] J. Håstad, R. Impagliazzo, L. Levin and M. Luby. Construction of a pseudo-random generator from any one-way function. Manuscript. Earlier versions in STOC 89 and STOC 90.

[ImLu] R. Impagliazzo and M. Luby. One-way Functions are Essential for Complexity-Based Cryptography. *Proceedings of the* 30th *Symposium on Foundations of Computer Science*, IEEE, 1989.

[ImYu] R. Impagliazzo and M. Yung. Direct Minimum-Knowledge Computations. *Advances in Cryptology – Crypto* 87 *Proceedings*, Lecture Notes in Computer Science Vol. 293, C. Pomerance ed., Springer-Verlag, 1987.

[IS1] T. Itoh and K. Sakurai. On the complexity of constant round ZKIP of possession of knowledge. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E76-A, No. 1, January 1993.

[Na] M. Naor. Bit Commitment using Pseudo-Randomness. *Advances in Cryptology – Crypto* 89 *Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.

[NOVY] M. Naor, R. Ostrovsky, R. Venkatasan, M. Yung. Perfect zero knowledge arguments for NP can be based on general complexity assumptions. *Advances in Cryptology – Crypto* 92 *Proceedings*, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

[OsWi] R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, IEEE, 1993.

[ToWo] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive-Proofs of Possession of Information. *Proceedings of the* 28th *Symposium on Foundations of Computer Science*, IEEE, 1987.

[Ya] A. C. Yao. Theory and Applications of Trapdoor functions. *Proceedings of the* 23rd *Symposium on Foundations of Computer Science*, IEEE, 1982.

## A Constant round ZK via coin flipping plus NIZK

The protocol stated in Figure 1 as obtained by combining [Bl, FLS] is folklore. First use Blum's coin flipping in the well protocol [Bl] to get a common random string, then do a NIZK [BDMP] proof, which can be done with a trapdoor permutation [FLS, BeYu]. In somewhat more detail, the first move is the verifier committing. For a four round ZK protocol we need a "certified one-way permutation." (Based on algebraic assumption, e.g. Discrete Logarithm. An arbitrary trapdoor permutation won't suffice.) After this the prover sends bits in the clear, the verifier de-commits, and the XOR of the prover bits and the verifier's decomitted bits is declared to be the common random string. The non-interactive ZK (NIZK) proof is run on the latter. The reason the full protocol is an argument, not a proof, is that the verifier's first round committals are done using a computational assumption.