Everything Provable is Provable in Zero-Knowledge

Michael Ben-Or	Hebrew University
Oded Goldreich	Technion – Israel Institute of Technology
Shafi Goldwasser	M.I.T. Laboratory for Computer Science
Johan Håstad	Royal Institute of Technology, Sweden
Joe Kilian	M.I.T. Laboratory for Computer Science
Silvio Micali	M.I.T. Laboratory for Computer Science
Phillip Rogaway	M.I.T. Laboratory for Computer Science

Abstract

Assuming the existence of a secure probabilistic encryption scheme, we show that every language that admits an interactive proof admits a (computational) zero-knowledge interactive proof. This result extends the result of Goldreich, Micali and Wigderson, that, under the same assumption, all of *NP* admits zero-knowledge interactive proofs. Assuming envelopes for bit commitment, we show tht every language that admits an interactive proof admits a perfect zero-knowledge interactive proof.

1. Introduction

Suppose Bob is polynomially time-bounded, but Alice has unlimited computational resources. If ϕ is a satisfiable boolean formula, Alice can certainly convince Bob of this fact; she could send Bob a message y describing a satisfying truth assignment for ϕ, and Bob could check that y does indeed specify a satisfying truth assignment. In other words, the language L of satisfiable boolean formulas is in *NP*.

The interaction between Alice and Bob in this example is very simple: Alice sends a single message to Bob, and no other messages are sent between the two. If ϕ is satisfiable, there is some message y that Alice might send which will convince Bob to accept. But if ϕ is not satisfiable, then no message that Alice might send will convince Bob to accept.

In the paper of Goldwasser, Micali, and Rackoff [GMR], the authors extend the scenario above in two ways, to arrive at the notion of an *interactive proof for the language L*. First, the interaction between Alice and Bob is allowed to be more complicated, with Alice and Bob exchanging multiple messages. Secondly, Alice and Bob are taken to be *probabilistic*, and Bob may occasionally accept or reject erroneously. It is required that *if* an input is in L, then Alice can behave in such a way that Bob will almost always accept; but if an input is *not* in L, then, no mater what messages Alice sends, Bob will almost certainly reject.

A different notion of provability "beyond *NP*" was independently proposed by Babai [Bab]. This notion is called an *Arthur-Merlin protocol*. Babai's model is similar to that of [GMR], but is seemingly more limited, because the verifier is required to reveal to the prover all of his coin flips (right after making them). Though this loss of privacy seems an important restriction, Goldwasser and Sipser [GS] show that, in fact, the models are equivalent with respect to language recognition.

Let *IP* be the class of languages that admit interactive proofs. Clearly $NP \subseteq IP$, for an *NP*-interaction is a special type of *IP*-interaction, in which the prover (Alice) sends the one and only message, and the verifier (Bob) never errs. However, *IP* may be a much larger class of languages. For example, there is an interactive proof known for graph *non*isomorphism, even though there are not known to be succinct certificates for establishing that a pair of graphs are not isomorphic.

S. Goldwasser (Ed.): Advances in Cryptology - CRYPTO '88, LNCS 403, pp. 37-56, 1990.

© Springer-Verlag Berlin Heidelberg 1990

In this paper, we are concerned with *zero-knowledge* interactive proofs. A zero-knowledge interactive proof for a language L is an interactive proof for L for which, on any input in L, the prover divulges to the verifier no significant amount of information *except* that the input is in L. (The notion of zero-knowledge we refer to has sometimes been called *computational zero-knowledge*, to distinguish it from two other notions of zero-knowledge that appear in the literature, *perfect zero-knowledge* and *statistical zero-knowledge*. [Fo]) For example, a zero-knowledge interactive proof that ϕ is a satisfiable boolean predicate convinces the verifier that ϕ is satisfiable, but not, say, by exhibiting a satisfying truth assignment.

Though the models of [Ba] and [GMR] are equivalent with respect to language recognition, they are likely *not* the same with respect to zero-knowledge; zero-knowledge interactive proofs frequently make use of the verifier's ability to have secret coins.

It might seem that requiring an interactive proof be zero-knowledge is generally too much to hope for; one might expect that relatively few languages with interactive proofs admit zero-knowledge interactive proofs. This was shown to probably not be the case in a paper of Goldreich, Micali, and Wigderson [GMW1]. Here the authors show that, assuming the existence of a secure probabilistic encryption scheme, *every* language in *NP* admits a zero-knowledge interactive proof.

We generalize this result to establish that, under the same assumption, *every* language that admits an interactive proof admits a zero-knowledge interactive proof.

A brief note on the history of this theorem. The result was stated in [GMW1], attributed to Ben-Or; this proof was never published. A published sketch of a proof appears in the CRYPTO-87 paper of Chaum, Damgård, and van de Graaf [CDG]. However, the result seems to require a stronger assumption, such as a pair of claw-free trapdoor functions. We have been learned [I] that Russell Impagliazzo and Moti Yung independently and by different methods had a proof of this theorem, which will appear in journal-form shortly [Yu]; their work is sketched in [IY].

In this paper, we will point out some subtleties involved and formally prove the theorem. We then discuss the "physical model" in which a bit can be committed by putting it in an envelope, and we show how to obtain perfect zero-knowledge proofs for *IP* under this model. The technique used here is different from the method that employs encryption. (The proofs of [IY] and [CDG] can be adapted to the envelope model, as well, as pointed out by [Br] and [Yu].)

The paper is organized as follows. Section 2 gives the preliminaries needed to understand the main theorem, including both definitions and well-known or technical results. The reader familiar with this area might skim or skip this section. Section 3 is devoted to the proof of the main theorem. Section 4 shows how to obtain perfect zero-knowledge proofs for languages in *IP* in the model in which a bit can be committed by putting it in an envelope. The remainder of this section is an informal overview of the proof of the main theorem.

1.1. Overview of the construction

We wish to show that, if $(P \leftrightarrow V)$ is an interactive proof system for the language L, then P and V can be modified to P' and V', such that $(P' \leftrightarrow V')$ is an interactive proof system for L, *but P' is zero-knowledge over L.*

Suppose $(P \leftrightarrow V)$ is an interactive proof system for the language L. We would like to carry out the "same interaction" in a way that betrays essentially no information to V. To do this, we could have P encrypt each message that it sends to V. That is, P uses a secure encryption function, E. On the ith round, when P "would have" sent to V the string y_i, P instead sends to V the string $E(y_i, d_i)$, a random encryption of y_i. (We assume that $E(x,s) = E(y,t)$ implies $x = y$, and that from $E(x,s)$ and s one can efficiently compute x. Security is with respect to *nonuniform* poly-time computation.)

There are two immediate difficulties. First, how can V be expected to compute his responses to P, since he doesn't understand what P has sent? Second, how can V be convinced to accept

the string x if, as far as he can tell, P has sent him complete gibberish?

The first problem—that V won't know what to do with the messages he's received—is answered as follows. By the result of Goldwasser and Sipser [GS], there is an *Arthur-Merlin* protocol, $(M \leftrightarrow A)$, for the same language, L. In $(M \leftrightarrow A)$, Arthur sends only his coin flips, so Arthur needn't understand the messages he's received in order to respond.

The second problem—that Arthur can't tell whether or not he ought accept—is answered as follows. If Arthur could *guess* the encryption keys, $d_1, d_2, \ldots$, he would have no problem knowing whether or not to accept, for he could decrypt each message sent by Merlin and accept or reject based on the same predicate he would have used had the conversation been carried out unencrypted. Of course, Arthur can't be expected to guess $d_1, d_2, \ldots$, but the statement "If you guessed $d_1, d_2, \ldots$, you'd accept x based on this interaction" is in *NP*. Since, by [GMW1], all of *NP* can be proven in zero-knowledge, there is a way for Merlin to convince Arthur of the validitity of this statement that is in zero-knowledge.

The construction just given has the following defect:

To show that $(M \leftrightarrow A)$ is a zero-knowledge proof system for L, we need to argue that *any* A^* learns essentially nothing by interacting with M. Suppose A^* cheats by flipping a biased coin in place of his random tape, with bias, say, $p = 3/4$. If A^* several times interacts with M on a common input $x \in L$, and if M usually convinces A^* to accept, then, intuitively, A^* has learned something: that most strings taken from this distribution lead to accepting x when used for A's random coins. This is "real knowledge," for it is entirely possible that even though M usually convinces A to accept if A uses a fair coin, M usually fails to convince A to accept if A uses a 3/4–biased coin.

(If we tried to prove that M is zero-knowledge, here's where we'd get stuck. The simulator M_{A^*} simulates the behavior of a "virtual prover," P^*, interacting with A^*. On common input $x \in L$, P^* sends random encryptions of the appropriate length string of 0's. When this phase of the interaction is finished, we append a simulated proof that Arthur would accept *if* he correctly guessed $d_1, d_2, \ldots$. But it is not always appropriate to append this simulated proof! For in the real interaction, $(M \leftrightarrow A^*)$, A^* sometimes rejects strings in L. Quite possibly, A^* has chosen a distribution of strings for which A would *usually* reject strings in L. So if P^* always appends the simulated proof that Arthur would accept, the resulting view may significantly differ from the real view of the interaction. But P^* has no way to know if it ought or ought not send the simulated proof.)

One possible fix is to use the result of Goldreich, Mansour, and Sipser [GMS], which says that we may, without loss of generality, take $(M \leftrightarrow A)$ to be *one-sided*. That is, we may assume that regardless of the coins that A employs on $x \in L$, A will be convinced to accept x. (Consequently, it will *always* be appropriate for P^* to convince A^* to accept in the simulated interaction.) This is the course that we shall follow.

Another possible fix is to use a "coin flip into the well" for A's coins ([Bl]). To make this proposal work, it is necessary that the coin flips that are agreed to are *statistically indistinguishable* ([Fo]) from truly random coin flips.

It is important that, in our formulation, independent encryption functions are, effectively, used on each round of the interaction. For example, we can not prove that it will be zero-knowledge for the prover to choose a public key encryption algorithm, E, with decryption algorithm D, send E to the verifier, and then send a random encryption of y_i under E, when y_i would otherwise be sent. (The verifier is convinced via the assertion, "If you guessed D, you'd accept the corresponding unecrypted conversation.")

(Here is the problem with such a scheme: Merlin, having received strings $x_1, \ldots, x_i$ from A^*, computes y_i by a probabilistic function of $(x, x_1, \ldots, x_i)$. Thus y_i is drawn from a probability

space R_i which A^* has some influence over. The encryption function is secure, so $(x_1, \ldots, x_i)$ can be only slightly correlated to D. *But a weak correlation of* $(x, x_1, \ldots, x_i)$ *to* D *may result in* R_i *being strongly correlated to* D*, for* M *is not necessarily polynomial time.* In fact, there is no reason to assume that R_i is not precisely the space that A^* wants it to be. But we mustn't allow A^* to have such strong influence over R_i; what if A^* forces R_i to be the space, say, with unit probability mass on D? Then Merlin sends Arthur $E(D)$ (a random encryption of the decryption function). This possibly compromises the the encryption function. In general, we worry that A^* may be able to select a space R_i for Merlin such that taking y_i from R_i and encrypting y_i under E compromises E.)

In any case, requiring a public key cryptosystem is a *stronger* assumption than the commitment scheme $E(msg, rand)$ that we require.

2. Preliminaries

2.1. Interactive proof systems

The definition we give for an *interactive proof system* is essentially that of Goldwasser, Micali, and Rackoff [GMR]; see this paper for a more complete discussion of interacting Turing machines and interactive proofs. It does not significantly effect the model if one assumes that the prover never halts, the verifier sends the first message, and communication is done on a single communication tape. We build these assumptions into the definition:

An *interactive proof system*, $(P \leftrightarrow V)$, consists of a pair of probabilistic Turing machines, P and V, with common alphabet Σ. P and V each have distinguished *start* and *quiescent* states. V has distinguished *accept* and *reject* states, out of which there are no transitions. P and V operate on various one-way infinite tapes:

- P and V have a common read-only *input tape*.
- P and V each have a private *random tape*, and a private *work tape*.
- P and V have a common *communication tape*.
- V is polynomially time-bounded. This means that there is a polynomial p for which, on inputs of length n, V experiences at most $p(n)$ state transitions before it accepts or rejects. V does *not* transition when it is quiescent, and P is running.
- P is finite expected time. This means that there is a function f such that, on inputs of length n, P's expected computation time from start to quiescent states does not exceed $f(n)$, regardless of the messages P has received.
- There is a polynomial r such that P never writes more than $r(n)$ characters (including blanks) on the communication tape when the common input is of length n.

Execution begins with P in its quiescent state and V in its start state. V's entering its quiescent state arouses P, causing it to transition to its start state. Likewise, P's entering its quiescent state causes V to transition to its start state. Execution terminates when V enters its accept or reject state.

If P and V are given random tapes $\sigma, \tau \in \Sigma^\omega$, respectively, and are then run on input $x \in \Sigma^*$, with work tapes initially empty, then the final state that V enters is well-defined, and we say that $(P_\sigma \leftrightarrow V_\tau)(x)$ *accepts* or *rejects* accordingly. If we omit mention of σ and τ, then we may speak of the "probability that $(P \leftrightarrow V)(x)$ accepts," $Pr[(P \leftrightarrow V)(x) \text{ accepts}]$.

Definition. $(P \leftrightarrow V)$ is an *interactive proof system for the language* L if, for some $0 \leq \epsilon < 1/2$, we have both:

completeness: $\quad x \in L \implies Pr[(P \leftrightarrow V)(x) \text{ accepts}] \geq 1 - \epsilon$, and

soundness: $\quad (\forall P^*) \quad x \notin L \implies Pr[(P^* \leftrightarrow V)(x) \text{ accepts}] \leq \epsilon$.

$(P \leftrightarrow V)$ is a *one-sided interactive proof system* if, in place of completeness, we have:
perfect completeness: $x \in L \implies Pr[(P \leftrightarrow V)(x) \text{ accepts}] = 1$.

That is, a one-sided interactive proof *always* accepts x when $x \in L$, regardless of the contents of the random tapes.

The number ϵ in this definition is called the *error probability*. By the standard method of running the protocol multiple times, we may take the error probability to be any constant in $(0, 1]$—or even any error probability of the form $\epsilon(n) = 2^{-p(n)}$, where p is a polynomial.

In order to extend our discussion to speak of *knowledge*, we consider the possibility that V's work tape initially contains "some knowledge." Suppose P and V are given random tapes $\sigma, \tau \in \Sigma^\omega$, respectively, and are run on input $x \in \Sigma^*$, with $s \in \Sigma^*$ initially placed on V's work tape. Then not only is the final state of V well-defined, but so are:

- The *number of rounds*, $2m$, for which P and V interact. (The number of rounds is the number of messages sent between A and B.)
- The ith message sent from V to P, x_i. (A *message* is a prefix of the communication tape, from its left end to the first blank).
- The ith message sent from P to V, y_i.
- The (finite) prefix τ_0 of τ that V reads.

That is, $(P, V, \sigma, \tau, x, s)$ determine the number m and strings $\vec{x} = x_1; \ldots; x_m;$, $\vec{y} = y_1; \ldots; y_m;$, as above. We define from these the *public history* of the interaction and the *view* of the interaction:

$$(P_\sigma \overset{pub}{\longleftrightarrow} V_\tau)(x, s) = [x, \vec{x}, \vec{y}],$$
$$(P_\sigma \overset{view}{\longleftrightarrow} V_\tau)(x, s) = [x, s, \tau_0, \vec{y}].$$

Interpret the right hand side of each of these definitions as the binary encoding of the specified string, where '[', ';' and ',' are new (formal) symbols.

Informally, the public history is the interaction as it would be observed from the "outside;" the view is the interaction as seen from V's perspective.

If we omit mention of σ and τ, then $(P \overset{pub}{\longleftrightarrow} V)(x, s)$ and $(P \overset{view}{\longleftrightarrow} V)(x, s)$ are *probability spaces*. $(P \overset{pub}{\longleftrightarrow} V)$ and $(P \overset{view}{\longleftrightarrow} V)$ (no mention of x or s) are *families of probability spaces*.

2.2. Arthur-Merlin protocols

In the definition for an interactive proof, the verifier was not compelled to reveal his coins flips (the prefix of his random tape that he uses) to the prover. If the verifier does reveal his coin flips at each round, there is no reason for him to send anything else, since the prover himself could as well compute anything else the verifier would have sent. A seemingly weaker notion of an interactive proof, introduced by Babai [Ba] [BaM], is obtained by limiting the verifier's messages in this way.

Definition. An interactive proof for L, $(M \leftrightarrow A)$, is an *Arthur-Merlin protocol* if for some polynomials r and l, any interaction between M and A on an input of length n takes exactly $r(n)$ rounds, each message sent being of length $l(n)$. Moreover, the $r(n)$ messages sent by A, $x_1, \ldots, x_{r(n)}$, are precisely the prefix $\tau_0 = x_1 \ldots x_{r(n)}$ of A's random tape that A consumes.

The first condition alone is easily seen not to weaken the model from that of an interactive proof system. Surprisingly, the second condition does not weaken the model either:

Theorem 2.1. *If $(P \leftrightarrow V)$ is an interactive proof system for L, then there is a one-sided, Arthur-Merlin protocol $(M \leftrightarrow A)$ for L.*

The result without "one-sided" is due to Goldwasser and Sipser [GS]; it was extended to proofs of perfect completeness by Goldreich, Mansour, and Sipser [GMS]. Recently, J. Kilian [K] discovered a much simpler argument for Theorem 2.1.

2.3. Zero-knowledge

$C_n \in \Sigma^*$ $(n \in \mathbb{N})$, $\mathcal{C} = \{C_n\}$ is a *poly-size family of circuits* if there are polynomials p and q such that $|C_n| \leq p(n)$, and C_n encodes (via some fixed universal Turing machine) a (deterministic) algorithm which, on input $x \in \Sigma^*$, requires at most $q(|x|)$ steps before it outputs a bit, 0 or 1.

If C is an algorithm that outputs a bit, and R is a probability space, then we may speak of "the probability that C outputs a 1 on input drawn from R," $p_R^C = \sum_{\sigma \in \Sigma^*} Pr_R(\{\sigma\}) \cdot C(\sigma)$.

We define zero-knowledge in terms of families of probability spaces indexed by two variables, which are treated differently.

Definition. Let $R = \{R(x,s)\}$ and $S = \{S(x,s)\}$ be families of probability spaces, indexed by $\Sigma^* \times \Sigma^*$. Then R and S are *indistinguishable over L* if, for any poly-size family of circuits $\{C_n\}$, any polynomial q, and all sufficiently long x in L,

$$\left| p_{R(x,s)}^{C_{|x|}} - p_{S(x,s)}^{C_{|x|}} \right| < \frac{1}{q(|x|)}$$

for all $s \in \Sigma^*$. If R and S are indistinguishable over L, we write $R \equiv_L S$ to denote this.

Definition. P is *zero-knowledge over L* if, for any V, there exists an expected polynomial-time algorithm M_V such that $(P \overset{view}{\longleftrightarrow} V) \equiv_L M_V$.

Definition. $(P \leftrightarrow V)$ is a *zero-knowledge interactive proof system for L* if $(P \leftrightarrow V)$ is an interactive proof system for L, and P is zero-knowledge over L.

We have defined indistinguishability with respect to poly-size families of circuits. In the proof of the main theorem, it will be convenient to think of indistinguishability with respect to poly-size families of probabilistic polynomial time algorithms. As with circuits, this is a *non-uniform* concept; there may be no algorithm which, on input n, outputs the expected poly-time algorithm C_n. By an averaging argument, and exploiting nonuniformity, it is easy to see that the notion of indistinguishability is unchanged if we define indistinguishability with respect to poly-size families of circuits, or with respect to poly-size families of probabilistic polynomial time probabilistic algorithms.

2.4. Preliminary results

It is frequently convenient to assume that, when P and V interact, the interaction takes place for a fixed number of rounds, messages are of a fixed length, and V uses a fixed number of coin flips per round. The following proposition says that there is no loss of generality in making these assumptions. The proof is straightforward and has been omitted.

Proposition 2.2. *If $(P \leftrightarrow V)$ is a (zero-knowledge) (one-sided zero-knowledge) interactive proof system for L, then there exists an P', V', and polynomials r, l, t, such that $(P' \leftrightarrow V')$ is a (zero-knowledge) (one-sided zero-knowledge) interactive proof system for L, and on each input of length n, the interaction runs for exactly $r(n)$ rounds, each message exchanged of length $l(n)$, and V' flipping precisely $t(n)$ coins for each message that it sends.* ◇

With respect to language recognition, we may further assume that the prover is deterministic. This observation (actually, that *PSPACE* was enough for the prover), was first made by Feldman [Fe].

Proposition 2.3. *If $(P \leftrightarrow V)$ is an interactive proof system for L, then there is a deterministic P' for which $(P' \leftrightarrow V)$ is an interactive proof system for L. If $(P \leftrightarrow V)$ was an Arthur-Merlin protocol, then so will be $(P' \leftrightarrow V)$.* ◇

The next proposition depends on the fact that the "composition" of zero-knowledge interactive proofs remains zero-knowledge. A proof of Proposition 2.4 can be found in the paper of Tompa and Woll [TW].

Proposition 2.4. *If L admits a (one-sided) zero-knowledge interactive proof, then L admits a (one-sided) zero-knowledge interactive proof with error probability ϵ for any $0 < \epsilon \leq 1/2$.* ◇

To state the next lemma—which conceptually simplifies the argument of the main result—we define the *composition* of two interactive proof systems, $(P_1 \leftrightarrow V_1)$ and $(P_2 \leftrightarrow V_2)$. Let us assume that the former always uses $r(|x|)$ rounds on any input x, and that each message is of length $l(|x|)$. (By Proposition 2.2, this entails no loss of generality.) $((P_2 \circ P_1) \leftrightarrow (V_2 \circ V_1))$ is defined as follows: Initially, $P_2 \circ P_1$ and $V_2 \circ V_1$, acting on common input x, behave like P_1 and V_1, respectively, acting on common input x. This continues for the first $r(|x|)$ rounds. However, $P_2 \circ P_1$ and $V_2 \circ V_1$ each record the public history of the interaction during these $r(|x|)$ rounds. After that, $P_2 \circ P_1$ checks that the public history of the interaction is a public history that *could* arise in a $(P_1 \leftrightarrow V_1)$ interaction on x. If so, $P_2 \circ P_1$ behaves like P_2 acting on input of the public history of the preceeding interaction; if not, all future messages of $P_2 \circ P_1$ are the empty string. $V_2 \circ V_1$ continues by behaving like V_2, acting on input of the public history of the preceeding interaction.

The technical lemma we need is

Lemma 2.5. *Suppose $(P_2 \leftrightarrow V_2) \circ (P_1 \leftrightarrow V_1)$ is an interactive proof system for L. Suppose P_1 is zero-knowledge over L, and P_2 is zero-knowledge over L', where $L' = (P_1 \overset{pub}{\longleftrightarrow} V_1)(L)$ is the set of all public histories that might arise in a $(P_1 \leftrightarrow V_1)$ interaction about a string in L. Then $P_2 \circ P_1$ is zero-knowledge over L.*

Proof: Let r_1 be the polynomial such that $(P_1 \leftrightarrow V_1)$ uses $r_1(|x|)$ rounds on any input x, and let l be the polynomial such that each of these messages is of length $l(|x|)$.

Let W be any polynomial-time probabilistic algorithm that interacts with $P_2 \circ P_1$. We may assume that $(P_2 \circ P_1 \leftrightarrow W)$ uses $r(|x|)$ rounds on any input x, where r is a polynomial exceeding r_1 pointwise.

We must exhibit an expected polynomial time machine M (of two arguments) for which $(P_2 \circ P_1 \overset{view}{\longleftrightarrow} W) \equiv_L M$.

Begin by constructing from W machines W_1 and W_2 as follows. W_1, on input (x, s), behaves exactly like W would behave on (x, s), but only for $r_1(|x|)$ rounds. After that, W_1 immediately accepts (or rejects).

W_2 takes as input a pair (x, s), where we assume $s = [\ \hat{x}, \hat{s}, \tau, \vec{y}\ ]$, and $\vec{y}$ is of the form $y_1; \ldots; y_m;$. s is the view of part of a computation of W. W_2 runs W, to resurrect the state W would be in after the conversation indicated by s. After that, W_2 behaves like W, starting from this state.

Since P_1 is zero-knowledge over L, there is an expected poly-time M_1 such that $(P_1 \overset{view}{\longleftrightarrow} W_1) \equiv_L M_1$.

Since P_2 is zero-knowledge over L', there is an expected poly-time M_2 such that $(P_2 \overset{view}{\longleftrightarrow} W_2) \equiv_{L'} M_2$.

M is constructed by "composing" M_1 and M_2. On input (x, s), M first runs M_1, to compute a string which, without loss of generality, looks like $[x, s, \tau_1, \vec{y}_1]$. Next, M runs $M_2(\Lambda, [x, s, \tau_1, \vec{y}_1])$, producing a string $[\Lambda, [x, s, \tau_1, \vec{y}_1], \tau_2, \vec{y}_2]$. M outputs $[x, s, \tau_1\tau_2, \vec{y}_1\vec{y}_2]$.

$M(x, s)$ can be described as the probability space resulting from performing the following experiment:

Experiment M:
$$\begin{array}{l} [x, s, \tau_1, \vec{y}_1] \leftarrow M_1(x, s). \\ [x, s, \tau_2, \vec{y}_2] \leftarrow M_2(\Lambda, [x, s, \tau_1, \vec{y}_1]). \\ \text{OUTPUT } [x, s, \tau_1\tau_2, \vec{y}_1\vec{y}_2]. \end{array}$$

$R(x, s) = (P_2 \circ P_1 \overset{view}{\longleftrightarrow} W)(x, s)$ can be viewed as the probability space associated with the following experiment:

Experiment R:
$$\begin{array}{l} [x, s, \tau_1, \vec{y}_1] \leftarrow (P_1 \overset{view}{\longleftrightarrow} V_1)(x, s). \\ [x, s, \tau_2, \vec{y}_2] \leftarrow (P_2 \overset{view}{\longleftrightarrow} V_2)(\Lambda, [x, s, \tau_1, \vec{y}_1]). \\ \text{OUTPUT } [x, s, \tau_1\tau_2, \vec{y}_1\vec{y}_2]. \end{array}$$

We now have two families of probability spaces, R of "real" prover-verifier interactions, and M of simulated interactions. Let's introduce one more, H, of "hybrid" interactions, with the following experiment used to define the probability space $H(x, s)$:

Experiment H:
$$\begin{array}{l} [x, s, \tau_1, \vec{y}_1] \leftarrow (P_1 \overset{view}{\longleftrightarrow} V_1)(x, s). \\ [x, s, \tau_2, \vec{y}_2] \leftarrow M_2(\Lambda, [x, s, \tau_1, \vec{y}_1]). \\ \text{OUTPUT } [x, s, \tau_1\tau_2, \vec{y}_1\vec{y}_2]. \end{array}$$

We now argue that $R \equiv_L M$. Suppose that this is not the case. Then there is some poly-size family $C = \{C_n\}$ and some polynomial q such that

$$\left| p^{C_{|x|}}_{R(x,s_x)} - p^{C_{|x|}}_{M(x,s_x)} \right| \geq \frac{1}{q(|x|)}$$

for infinitely many $(x, s_x) \in L \times \Sigma^*$. Then either

$$\left| p^{C_{|x|}}_{R(x,s_x)} - p^{C_{|x|}}_{H(x,s_x)} \right| \geq \frac{1}{2q(|x|)} \tag{1}$$

or

$$\left| p^{C_{|x|}}_{H(x,s_x)} - p^{C_{|x|}}_{M(x,s_x)} \right| \geq \frac{1}{2q(|x|)} \tag{2}$$

for infinitely many $(x, s_x) \in L \times \Sigma^*$. We show that both of these cases are impossible.

Case 1. ((1) holds infinitely often.) Choose an (x, s_x) for which (1) holds. Single out the coin flips used by V_1, and the messages $y_1, \ldots, y_{r_1(|x|)}$ that P_1 might send to V_1 in the interactions defining $R(x, s_x)$ and $H(x, s_x)$. There is a *particular* sequence of coin flips σ for V_1 to use, and a *particular* vector of messages $\tau = y_1 \ldots y_{r_1(|x|)}$ for P_1 to use, such that

$$\left| p^{C_{|x|}}_{R_{\sigma,\tau}(x,s_x)} - p^{C_{|x|}}_{H_{\sigma,\tau}(x,s_x)} \right| \geq \frac{1}{2q(|x|)},$$

where $p^{C_n}_{R_{\sigma,\tau}(x,s_x)}$ is the probability that C_n outputs 1 when Experiment R is run with σ and τ used for V_1's coins and P_1's messages, respectively; likewise for $p^{C_n}_{S_{\sigma,\tau}(x,s_x)}$. Consequently, we may

"hardwire" into C_n the values we obtain from $(P_1 \leftrightarrow V_1)$ interacting on (x, s_x) using σ and τ, to obtain a circuit which distinguishes $(P_2 \leftrightarrow V_2)(x, s_x)$ from $M_2(x, s_x)$ by at least $1/(2q(|x|)$. The existence of the family of circuits modified as specified here contradicts $(P_2 \leftrightarrow V_2)(\cdot,\cdot) \equiv_{L'} M_2(\cdot,\cdot)$.
Case 2, when (2) holds infinitely often, is handled analogously. ◇

For completeness, we state the following trivial proposition:

Proposition 2.6. *Let $L' \subseteq L$. If P is zero-knowledge over L, then P is zero-knowledge over L'.* ◇

2.5. Secure probabilistic encryption

The prover in our protocol will need the ability to securely commit a bit, and to convincingly decommit it. We formalize this by saying that a *secure probabilistic encryption scheme* is a function $E : \Sigma \times \Sigma^* \to \Sigma^*$ such that

(1) E is computable in polynomial time.
(2) *Unique decryption*: $E(\beta, x) = E(\beta', y)$ implies $\beta = \beta'$.
(3) Let $E_n(\beta)$ be the probability space obtained by setting $Pr(y) = 2^{-n} \cdot |\{x \in \Sigma^n : E(\beta, x) = y\}|$. We require that for any poly-size family of circuits $\mathcal{C} = \{C_n\}$, for any polynomial q, and for all sufficiently large n,

$$\left| p^{C_n}_{E_n(0)} - p^{C_n}_{E_n(1)} \right| < \frac{1}{q(n)}.$$

(Recall p^C_R is the probability that circuit C outputs 1 on input drawn from R. Note that to achieve the unique decryption condition with conventional encryption schemes, "certified primes" must be used [GK][AH].) We write $\{E_n(0)\} \equiv_{\mathbf{N}} \{E_n(1)\}$ to denote the security condition.

Without loss of generality, there is a polynomial q such that $|E(\beta, x)| = q(|x|)$ for all $x \in \Sigma^*$.

To encrypt a bit β with security parameter n, select a random n-bit string x and send $E(\beta, x)$. To decommit, reveal x. The unique decryption condition makes it impossible that the commited bit could be $1 - \beta$. Also, from x and $E(\beta, x)$ one can easily compute β.

To encrypt a string $m = \beta_1 \ldots \beta_\ell$ with security parameter n, send $E(\beta_1, x_1) \ldots E(\beta_\ell, x_\ell)$ for random n-bit strings $x_1, \ldots, x_\ell$. The encryption will be denoted $E_n(m, x)$, where $x = x_1 \ldots x_m$, and the corresponding probability space is denoted $E_n(m)$.

A secure encryption scheme exists if there are unapproximable predicates [GM], or if there are injective one-way functions [Ya][L][G]. (If f is injective one-way, then there are poly-time computable functions f' and a b such that $f'(x) = f'(y)$ implies $b(x) = b(y) \in \{0, 1\}$, and no poly-size circuit family can predict $b(x)$ given $f(x)$ by better than $1/2 + n^{-c}$, for any constant c. Given such f' and b, E as we have described it can readily be constructed.)

The crucial property we need of a secure probabilistic encryption scheme is the following:

Lemma 2.7. *Assume the existence of a secure probabilistic encryption scheme. Let $\{y_n\}$ be a collection of strings, where $|y_n| = l(n)$, for some nonconstant polynomial l. Then*

$$\{E_n(y_n)\} \equiv_{\mathbf{N}} \{E_n(0^{l(n)})\}.$$

Proof: Suppose to the contrary that there is a poly-size family $\mathcal{C} = \{C_n\}$ and a polynomial q such that

$$\left| p^{C_n}_{E_n(y_n)} - p^{C_n}_{E_n(0^{l(n)})} \right| \geq \frac{1}{q(n)}$$

for infinitely many $n \in \mathbf{N}$. Pick a particular n for which this holds. Define the strings y_n^i, for $0 \le i \le n$, by $y_n^i = y_n[1..i]0^{l(n)-i}$. Note $y_n^0 = 0^{l(n)}$, and $y_n^{l(n)} = y_n$. There exists, then, a j, $0 \le j < l(n)$, such that

$$\left| p^{C_n}_{E_n(y_n^j)} - p^{C_n}_{E_n(y_n^{j+1})} \right| \ge \frac{1}{l(n) \cdot q(n)}.$$

Note that y_n^j and y_n^{j+1} agree at all positions except the $(j+1)$-*st* where y_n^j is 1 and y_n^{j+1} is 0. Consequently, we may hardwire into C_n the values of y_n^j at each position except the $(j+1)$-*st*, to obtain a coin-flipping circuit which distinguishes encyptions of 0 from encryptions of 1 by at least $1/(l(n)q(n))$. Converting to a deterministic circuit, we contradict condition (3) about our encryption scheme. ◇

2.6. Zero-knowledge proofs for all of *NP*

The following lemma and theorem are due to Goldreich, Micali, and Wigderson [GMW1]. The proof of the first of these is omitted.

Lemma 2.8. *If secure probabilistic encryption is possible, then the language of (encodings of) 3-colorable graphs admits a (one-sided) zero-knowledge interactive proof.* ◇

Theorem 2.9. *If secure probabilistic encryption is possible, then any language in NP possesses a (one-sided) zero-knowledge interactive proof.*

Proof: Take $L \in NP$, and let M be a nondeterministic Turing machine for L. Fix a canonical transformation φ that takes any (M, x) (the encoding of a nondeterministic Turing machine and an input x) to a graphs G. φ is poly-time computable, and has the property that M accepts x iff $\varphi(M, x)$ is 3-colorable.

To prove $x \in L = L(M)$ in zero-knowledge, both the prover and the verifier compute the graph $G = \varphi(M, x)$, and engage in a zero-knowledge interactive proof (using Lemma 2.8) that G is 3-colorable. ◇

3. Proof of the main theorem

We now prove the main theorem of this paper:

Theorem 3.1. *Assuming a secure probabilistic encryption scheme exists, every language that admits an interactive proof admits a zero-knowledge interactive proof.*

Proof: Suppose L admits an interactive proof. Then, by Theorem 2.1, L admits a *one-sided, Arthur-Merlin* interactive proof $(M \leftrightarrow A)$. By Proposition 2.3, M may be assumed to be deterministic. By Proposition 2.4, we may take the error probability of $(M \leftrightarrow A)$ to be less than $1/5$.

By Proposition 2.2, $(M \leftrightarrow A)$ may be assumed to always use $r(n)$ rounds, each message of length $l(n)$, when M and A interact with common input x of length n. (r and l are polynomials.)

We will construct from M and A a zero-knowledge, one-sided interactive proof system $(P \leftrightarrow V)$ for L.

Suppose Arthur's random tape contains a given infinite string. On input x of length n, Arthur only uses the $(l(n) \cdot r(n))$-bit prefix of this string, $x_1 \cdots x_{r(n)}$, where $|x_i| = l(n)$. Arthur sends Merlin $x_1, \cdots, x_{r(n)}$, receiving (interleaved with theses queries) the messages $y_1, \cdots, y_{r(n)}$. Then Arthur accepts or rejects according to the deterministic, poly(n)-time computable predicate

$$P_A(x, x_1, \cdots, x_{r(n)}, y_1, \cdots, y_{r(n)})$$

that he possesses.

To transform $(M \leftrightarrow A)$ into $(P \leftrightarrow V)$, a zero-knowledge interactive proof system for L, we will have the prover, P, behave like M, and the verifier, V, behave like A, with the following exceptions: The prover will encrypt each message y_i that he sends to the verifier, and then convince the verifier that he (V) would accept if he knew the corresponding encryption keys.

That is, the protocol runs in two phases. In the first phase, if P and V share input x of length n, then on round i, when Merlin "would have" sent to Arthur the string y_i, P instead randomly selects an $nl(n)$-bit string d_i and sends to V the string $\alpha_i = E_n(y_i, d_i)$. For the second phase, after all $r(n)$ rounds of the first phase are completed, the prover decides whether or not A would have accepted the corresponding unencrypted conversation, a fact which P can easily discern using P_A. If A would have accepted, then P convinces V that A would have accepted. That is, P convinces V of the validity of the NP-assertion

$$(\exists\, d_1, \ldots, d_{r(n)}, y_1, \ldots, y_{r(n)})\ \ [(E_n(y_i, d_i) = \alpha_i \text{ for all } i)\ \ \wedge \\ P_A(x, x_1, \ldots, x_{r(n)}, y_1, \ldots, y_{r_n})].$$

P convinces V of this assertion by computing a graph G which is 3-colorable if and only if the preceding assertion holds, and then convincing V that G is 3-colorable using the method of [GMW1]. Enough rounds are used in this protocol to convince V that G is 3-colorable with probability at least $4/5$. Note G can be computed by a deterministic poly(n) time algorithm, φ, so both P and V "know" G after the first $r(n)$ interactions.

Let φ be the canonical map (appears in the proof of Theorem 2.9) that takes a tuple $(x, x_1, \ldots, x_{r(n)}, \alpha_1, \ldots, \alpha_{r(n)})$ to a graph G which is 3-colorable iff there is a guess $y_1, \ldots, y_{r(n)}$, $d_1, \ldots, d_{r(n)}$ for which $\alpha_i = E(y_i, d_i)$ and A would accept the corresponding unencrypted conversation according to P_A. We may assume that $|E(G)|$ is always a power of 2. Though we include details of Phase 2 for completeness, it can be viewed as a black box that accepts the public conversation with error probability $< 1/5$ whenever M would have accepted the corresponding unencrypted conversation.

Protocol for the prover, P (on input x of length n)

If $x \notin L$, all messages to the verifier are Λ. Otherwise ...

On rounds $1 \le i \le r(n)$: PHASE 1 ...

- Wait to receive a message x_i from the verifier.
- If $|x_i| \neq l(n)$, all future messages to the verifier are Λ. Otherwise ...
- Compute $y_i \leftarrow M(x, x_1, \ldots, x_i)$.
- Randomly select $d_i \in \Sigma^{nl(n)}$.
- Send $\alpha_i = E_n(y_i, d_i)$ to the verifier.

On round $i = r(n) + 1$: PHASE 2 ...

- Compute the graph $G = \varphi(x, x_1, \ldots, x_{r(n)}, \alpha_1, \ldots, \alpha_{r(n)})$, $V(G) = \{1, \ldots, \nu\}$.
- Compute a random (proper, vertex) 3-coloring of G, $\theta_i : V(G) \to \{01, 10, 11\}$.
- Randomly select $d_1^i, \ldots, d_\nu^i \in \Sigma^{2n}$.
- Send $E_n(\theta_i(1), d_1^i), \ldots, E_n(\theta_i(\nu), d_\nu^i)$ to the verifier.

For rounds $i \leftarrow r(n) + 2$ to ∞:

- Receive an edge $\{j, k\}$ from the verifier.
  All future messages are Λ if receive something not of this form.
- Send (d_j^{i-1}, d_k^{i-1}) to the verifier.
- Select a random 3-coloring of G, θ_i.
- Randomly select $d_1^i, \ldots, d_\nu^i \in \Sigma^{2n}$.
- Send $E_n(\theta_i(1), d_1^i), \ldots, E_n(\theta_i(\nu), d_\nu^i)$ to the verifier.

Protocol for the verifier, V (on input x of length n)

On round 1: PHASE 1 ...
- Read off first $l(n)$ bits of random tape into x_i.
- Send x_i to the prover.

On rounds $2 \leq i \leq r(n)$:
- Receive α_{i-1} from the prover.
- Read off next $l(n)$ bits of random tape into x_i.
- Send x_i to the prover.

On round $i = r(n) + 1$:
- Receive $\alpha_{r(n)}$ from the prover.
- Send Λ to the prover.

On round $i = r(n) + 2$: PHASE 2 ...
- Compute $G = \varphi(x, x_1, \ldots, x_{r(n)}, \alpha_1, \ldots, \alpha_{r(n)})$, $V(G) = \{1, \ldots, \nu\}$.
- Receive $(\alpha_1^i, \ldots, \alpha_\nu^i)$ from the prover.
- If not of this form, *reject*.
- Randomly select an edge $\{j, k\} \in E(G)$.
- Send $\{j, k\}$ to the prover.

For rounds $i \leftarrow r(n) + 3$ to $r(n) + 3 + 2m$:
- Receive (d_j^{i-1}, d_k^{i-1}) from the prover.
- If not of this form, or if it is not the case that for distinct $u, v \in \{01, 10, 11\}$ is $\alpha_j^{i-1} = E_n(u, d_j^{i-1})$, $\alpha_k^{i-1} = E_n(v, d_k^{i-1})$, *reject*.
- Receive $\alpha_1^i, \ldots, \alpha_\nu^i$ from the prover.
- If not of this form, *reject*.
- Randomly select $\{j, k\} \in E(G)$, and send $\{j, k\}$ to the prover.

accept.

We have three things to check: that V accepts all strings in L; that V usually rejects strings not in L, even if P is replaced by some *other* probabilistic algorithm; and that P is zero-knowledge over L.

The first two of these claims are easy. Choose $x \in L$, where $|x| = n$. Then for *any* strings $x_1, \ldots, x_{r(n)} \in \Sigma^{l(n)}$, we know that A interacting with M would accept when A sends messages $(x_1, \ldots, x_{r(n)})$. Since the interactive proof for graph 3-colorability is one-sided, P will always always be able to convince V that A would accept $(x, x_1, \ldots, x_{r(n)}, \alpha_1, \ldots, \alpha_{r(n)})$ if A knew the corresponding endcyption keys. So, in fact, we retain perfect completeness.

Suppose V is interacting with a corrupt prover, P^*, and the common input is x, a string of length n, where $x \notin L$. The probability that V will accept a string which A would not have accepted when given the corresponding unencrypted messages is at most $1/5$. But for any $x \notin L$, A accepts with probability at most $1/5$. Thus V fallaciously accepts x with probability at most $2/5$, so the proof system is sound.

We now show that P is zero-knowledge over L. By Lemma 2.5, if we prove that P is zero-knowledge for the first phase of the interaction, we will be done: the whole interaction is the composition of $(P \leftrightarrow V)$ restricted to the first phase, with $(P \leftrightarrow V)$ restricted to the second phase, and the second phase is zero-knowledge over the output of the first phase.

Let P_1 be the protocol that carries out the first phase of the interaction. Inquiries beyond the $r(n)$-*th* are answered with the empty string.

Let W be a probabilistic poly-time algorithm that interacts with P_1. We may assume that W flips exactly $t(n)$ coins on each round, where t is a polynomial, and n is the length of the common input.

We may assume that $(P_1 \leftrightarrow W)$ always uses exactly $r(n)$ rounds, each message of length $l(n)$, when the common input is of length n.

M_W simulates a "virtual prover," $\hat{P}_1$, interacting with W. M_W uses its coins at odd positions for $\hat{P}_1$'s coins, and its coins at even positions for W's coins. M_W, after simulating the interaction, outputs the view of this interaction. Note that M_W is polynomial time.

Here is the protocol for $\hat{P}_1$:

On rounds $1 \leq i \leq r(n)$:
- Wait to receive a message x_i from the verifier.
- If $|x_i| \neq l(n)$, all future messages are Λ. Otherwise ...
- Randomly select $d_i \in \Sigma^{nl(n)}$.
- Send $E_n(0^{l(n)}, d_i)$ to the verifier.

On future rounds:
- Send Λ to the verifier.

We argue that $(P_1 \leftrightarrow W)(\cdot,\cdot) \equiv_L M_W(\cdot,\cdot)$. The auxiliary string plays no role in the proof (other than to be given to W), so we omit further mention of it. Denote the space $(P_1 \leftrightarrow W)(x)$ by $S_{r(|x|)}$, and $M_W(x)$ by $S_0(x)$. Assume for contradiction that these families of spaces are computationally *distinguishable* over L. That is, there exists a polynomial size family of circuits $C = \{C_n\}$ and a polynomial h such that

$$\left| p^{C_{|x|}}_{S_0(x)} - p^{C_{|x|}}_{S_{r(|x|)}(x)} \right| \geq \frac{1}{h(|x|)}$$

for infinitely many x in L.

A "probability walk" is now used. Let $S_j(x)$ be the probability space obtained by using the real prover, P_1, for the first j rounds with V, and the virtual prover, $\hat{P}_1$, for the remaining rounds. That is, $S_j(x)$ is the probability space defined by the interaction between W and the following prover, $\tilde{P}_j$.

On rounds $1 \leq i \leq r(n)$:
- Wait to receive a message x_i from the verifier.
- If $|x_i| \neq l(n)$, all future messages are Λ. Otherwise ...
- Compute $y_i = \begin{cases} M(x, x_1, \ldots, x_i), & \text{if } i \leq j; \\ 0^{l(n)}, & \text{otherwise.} \end{cases}$
- Randomly select $d_i \in \Sigma^{nl(n)}$.
- Send $E_n(y_i, d_i)$ to the verifier.

On future rounds:
- Send Λ to the verifier.

Observe that this agrees with our previous definition of $S_0(x)$ and $S_{r(|x|)}(x)$, and that the defining algorithms for $S_j(x)$ and $S_{j+1}(x)$ differ in behavior only on the $(j+1)$-*st* round, at which point S_j uses $\hat{P}_1$ while S_{j+1} uses P_1.

By the triangle inequality, there are infinitely many x in L for which there is an associated i, $0 \le i < r(|x|)$, such that

$$\left| p_{S_i(x)}^{C_{|x|}} - p_{S_{i+1}(x)}^{C_{|x|}} \right| \ge \frac{1}{r(|x|) \cdot h(|x|)}. \tag{1}$$

Using $\mathcal{C}$, we construct a poly-size family of expected polynomial time algorithms, $\mathcal{C}' = \{C'_n\}$, and an infinite collection of strings $\{z_n\}$, $|z_n| = l(n)$, such that C'_n effectively distinguishes the probability space $E_n(z_n)$ from the probability space $E_n(0^{l(n)})$.

Choose $x \in L$, $|x| = n$, and i, for which the bound in (1) holds. We show how to modify C_n to obtain C'_n. Let

$$\left| p_{S_i(x)}^{C_n} - p_{S_{i+1}(x)}^{C_n} \right| = \epsilon_n,$$

where $\epsilon_n \ge 1 \,/\, r(n)h(n)$.

Consider the first $i+1$ rounds between a prover and W. An $f(n) = (i+1)t(n)$-bit prefix, σ, of W's random tape, and strings $\alpha_1, \ldots, \alpha_i$ that the prover sends to W determine (1) the first $i+1$ messages, $x_1, \ldots, x_{i+1}$, that W sends; (2) W's state after this portion of the conversation; and (3) the string y_{i+1} that the prover will next encrypt and send to W (recall that the prover is deterministic on each round up to the point at which it encrypts).

Let $S_i^\sigma(x)$ be the probability space obtained by having $\tilde{P}_i$ interact with W on input x, where W's random tape has prefix σ. Then

$$p_{S_i(x)}^{C_n} = \sum_{\sigma \in \Sigma^{f(n)}} 2^{-f(n)} \cdot p_{S_i^\sigma(x)}^{C_n}.$$

Similarly, for S_{i+1}.

Now, by the triangle inequality,

$$\epsilon_n = \left| p_{S_i(x)}^{C_n} - p_{S_{i+1}(x)}^{C_n} \right| \le \sum_{\sigma \in \Sigma^{f(n)}} 2^{-f(n)} \left| p_{S_i^\sigma(x)}^{C_n} - p_{S_{i+1}^\sigma(x)}^{C_n} \right|,$$

so there is a *particular* $\sigma \in \Sigma^{f(n)}$ which achieves

$$\left| p_{S_i^\sigma(x)}^{C_n} - p_{S_{i+1}^\sigma(x)}^{C_n} \right| \ge \epsilon_n.$$

Fix such a σ. For this σ, the prover induces a certain distribution on the first i messages it sends W, $\alpha = \alpha_1 \ldots \alpha_i$, $|\alpha| = g(n) = il(n)q(n)$, where $q(n)$ is the number of bits needed to encrypt a bit under E. Let λ_α be the probability that the prover's first i messages will be α, $\sum \lambda_\alpha = 1$. Define $S_i^{\sigma,\alpha}(x)$ as the probability space obtained by having $\tilde{P}_i$ interact with W on x, where W has random tape prefixed by σ, and $\tilde{P}_i$'s initial responses are α. Then

$$p_{S_i^\sigma(x)}^{C_n} = \sum_{\alpha \in \Sigma^{g(n)}} \lambda_\alpha \cdot p_{S_i^{\sigma,\alpha}(x)}^{C_n}.$$

Similarly, for S_{i+1}.

Now, as before

$$\epsilon_n \leq \left| p^{C_n}_{S^\sigma_i(x)} - p^{C_n}_{S^\sigma_{i+1}(x)} \right| \leq \sum_{\alpha \in \Sigma^{g(n)}} \lambda_\alpha \left| p^{C_n}_{S^{\sigma,\alpha}_i(x)} - p^{C_n}_{S^{\sigma,\alpha}_{i+1}(x)} \right|,$$

so there is a *particular* $\alpha \in \Sigma^{g(n)}$ which achieves

$$\left| p^{C_n}_{S^{\sigma,\alpha}_i(x)} - p^{C_n}_{S^{\sigma,\alpha}_{i+1}(x)} \right| \geq \epsilon_n.$$

Now σ and $\alpha = \alpha_1 \ldots \alpha_i$ determine $x_1, \ldots, x_{i+1}$ and y_{i+1} such that *if* the interaction determined by σ and α is executed, and *then* $d \in \Sigma^{nl(n)}$ is selected at random, and *then, either*

case 1: $E_n(y_{i+1}, d)$ is sent to W, *or*

case 2: $E_n(0^{l(n)}, d)$ is sent to W,

and *then*, $\hat{P}_1$ and W are allowed to continue their interaction (which will last another $r(n) - i - 1$ rounds)—if all this is done, then the probability space associated with case 1 and the probability space associated with case 2 are distinguishable by C_n by at least ϵ_n.

C'_n is a probabilistic polynomial time algorithm that has σ and α "hardwired in." C'_n begins by bringing the state of W up to the state it would be in if its random tape began with σ, and it received messages $\alpha_1, \ldots, \alpha_i$ from the prover. The messages $x_1, \ldots, x_{i+1}$ that W would send are determined during this process, and they are recorded. C'_n expects a string α_{i+1} as input. This input is fed to W as its $(i+1)$-*st* message from the prover. From now on, C'_n uses its real random tape, and comes up with a query x_{i+2} for W to have made. However, C'_n answers its own queries using $\hat{P}_1$. This continues until C'_n has constructed a complete conversation, $((x_1, \alpha_1), \ldots, (x_{r(n)}, \alpha_{r(n)}))$, together with associated coin flips for W (which is σ with some random $(t(n)(r(n) - i - 1)$-bit string appended). C'_n constructs the associated view of the conversation, and feeds this to C_n to obtain a bit, 0 or 1. C'_n outputs this bit.

The poly(n) length of α and σ guarantees that C'_n is expected polynomial time. And

$$\left| p^{C'_n}_{E_n(y_{i+1})} - p^{C'_n}_{E_n(0^{l(n)})} \right| \geq \epsilon_n$$

by our construction.

Set $z_n = y_{i+1}$. The family of probabilistic polynomial time algorithms $C' = \{C'_n\}$ (indexed by the same infinite set of naturals as in (1)) so constructed constitutes a poly-size family of probabilistic polynomial algorithms that distinguishes $\{(E_n(z_n)\}$ from $\{(E_n(0^{l(n)})\}$ by at least ϵ_n. By our remark that distinguishability by polynomial size families of probabilistic polynomial time algorithms implies distinguishability by poly-size families of circuits, we have contradicted Lemma 2.7. Our original assumption—that $(P_1 \leftrightarrow W)$ is distinguishable from M_W—is therefore in error.

That P itself is zero-knowledge follows from Lemma 2.5. The second phase of the interaction depends only on the public history of the first phase of the interaction. Recall that, by one-sidedness, whenever $x \in L$, A would accept when interacting with M, so the graph G generated following the interaction will always be 3-colorable. Since the second phase of the interaction is precisely the graph-isomorphism protocol applied to a deterministic poly-time computable function of the public history, Lemmas 2.8 and 2.6 tell us that the second phase of the interaction is zero-knowledge over the possible public histories. ◇

4. Notarized Envelopes: Description and Implementation

The interactive proof that a graph is 3-colorable ([GMW1]) can be implemented in perfect zero-knowledge using *envelopes* for committing strings. For each vertex, the prover puts into a vertex-labeled envelope a slip of paper giving the color of that vertex. These envelopes are placed before the verifier. The verifier chooses an edge and the prover allows the verifier to open the envelopes for the edge's endpoints. As a consequence of this protocol, all of *NP* can be implemented with envelopes in perfect zero-knowledge.

It is natural to ask if every language in *IP* can be proven in zero-knowledge using envelopes for commitment. The proof of the preceding section does not immediately give a solution to this problem. In this section, we answer this question in the affirmative.

4.1. Introduction to notarized envelopes

We now consider a stronger type of commitment scheme, known as *notarized envelopes*. Notarized envelopes allow one to commit and decommit a sequence of bits, $b_1, \ldots, b_n$, just as with ordinary envelopes. However, with notarized envelopes one can additionally prove any single *NP* assertion, $P(b_1, \ldots, b_n)$, during or after the commital stage. In our implementation using ordinary envelopes, this proof is in perfect zero-knowledge. If $P(b_1, \ldots, b_n)$ does not hold (or a poly-time bounded commitor does not have a witness of this fact), then the verifier will reject with probability at least $1/n^c$, where c is a constant which depending on P. This probability may be amplified arbitrarily by standard techniques.

A notarized envelope scheme may be thought of as a set of three protocols: A *commital* protocol, a *decommital* protocol, and a *zero-knowledge proof* protocol. Nearly all of the complexity of our implementation comes from the zero-knowledge protocol.

4.2. An implementation of notarized envelopes

Our reduction from notarized envelopes to ordinary envelopes is essentially a simplified version of Kilian's reduction from notarized envelopes (or, in his terminology, *commital with zero-knowledge proofs*) to oblivious transfer([K1]). However, our protocol has somewhat different properties from Kilian's, due to the fact that we are using envelopes instead of oblivious transfer. Using oblivious transfer, one can noninteractively commit bits with zero-knowledge proofs. Our scheme requires a constant number of rounds of interaction. It is not hard to show that any implementation based on ordinary envelopes must have some interaction, so our solution is optimal up to constant factors. Also, our implementation achieves perfect zero-knowledge, whereas Kilian's only achieves statistical zero-knowledge.

Commital and Decommital

We first present our protocols for committing and decommitting a set of bits, $b_1, \ldots, b_n$. In our protocols, we adopt the convention that Alice commits the bits, and Bob acts as the verifier.

Protocol Commit$(b_1, \ldots, b_n)$ /* Commit $b_1, \ldots, b_n$ */

1: Alice uniformly chooses bits $x_1, \ldots, x_{2n}$ subject to

$$b_i = x_i \oplus x_{i+n}.$$

2: Alice commits the x_i's to Bob, using ordinary envelopes. Bob is allowed to know which envelope is supposed to contain which x_i.

Protocol Decommit(i) /* Decommit b_i */

1: Alice opens the envelopes containing x_i and x_{i+n}. Bob computes $b_i = x_i \oplus x_{i+n}$.

Clearly, Bob gets no information about any of the b_i's from the commital protocol, and, on decommit, Bob only gains information that bit which is being decommitted.

Zero-knowledge Proofs

Our implementation of zero-knowledge proofs is somewhat more complicated. We first use the simple observation that it suffices to consider predicates, P, which are in NC^1 ([K1]). Furthermore, given an NC^1 predicate, $P(b_1, \ldots, b_n)$, the predicate $P'(x_1, \ldots, x_{2n})$ defined by

$$P'(x_1, \ldots, x_{2n}) = P(x_1 \oplus x_{n+1}, \ldots, x_n \oplus x_{2n}),$$

will also be in NC^1. Now, if $P'(x_1, \ldots, x_{2n})$ is in NC^1, then by a theorem of Barrington [Ba], there is a polynomial sized width 5 permutation branching programs (W5PBP) for P'.

A branching program B may be thought of a a sequence of triples,

$$(i_1, \pi_1^0, \pi_1^1), \ldots, (i_m, \pi_m^0, \pi_m^1),$$

and a special element, a. For $j \in [1, m]$, we have $i_j \in [1, n]$, and $\pi_j^0, \pi_j^1 \in S_5$, where S_5 is the group of permutations on 5 elements. The special element a is also in S_5, and must not be equal to the identity. A branching program, B, realizes a predicate $P(b_1, \ldots, b_n)$ if the product

$$\prod_{j=1}^{m} \pi_j^{b_{i_j}} = \begin{cases} a & \text{when } P(b_1, \ldots, b_n) \text{ is true;} \\ I & \text{when } P(b_1, \ldots, b_n) \text{ is not true.} \end{cases}$$

Here, I represents the identity element for S_5. Given an NC^1 circuit for P', Barrington shows how to construct a canonical branching program which realizes P', which we denote by $B_{P'}$.

We can now describe our protocol for giving zero-knowledge proofs of some NC^1 predicate, P. We assume that Alice has committed bits $b_1, \ldots, b_n$, generating bits $x_1, \ldots, x_{2n}$.

Protocol Prove$(x_1, \ldots, x_{2n}, P)$ /* prove $P(b_1, \ldots, b_n)$ */

1: Let $B_{P'}$ be a canonical W5PBP for P'. We write

$$B_{P'} = ((i_1, \pi_1^0, \pi_1^1), \ldots, (i_m, \pi_m^0, \pi_m^1), a)$$

Alice computes the sequence $A_1, \ldots, A_m$ by

$$A_j = \pi_j^{x_{i_j}}.$$

She then uniformly chooses $R_1, \ldots, R_{m-1}$, where $R_i \in S_5$. For convenience, we define $R_0 = R_m = I$. Finally, she computes a new sequence, $B_1, \ldots, B_m$, defined by

$$B_i = R_{i-1} A_i R_i^{-1}.$$

She then commits A_i, R_i, B_i, for $i \in [1, m]$, using ordinary envelopes.

2: Bob uniformly chooses one of the following three types of queries to make of Alice:

a: Bob asks Alice to reveal $B_1, \ldots, B_m$. He rejects if

$$\prod_{i=1}^{m} B_i \neq a.$$

b: Bob asks Alice to, for some $j \in [1, m]$, reveal A_i, R_{i-1}, R_i, B_i. He rejects if

$$B_i \neq R_{i-1} A_i R_i^{-1}.$$

The values of R_0, R_m are assumed to be I, and thus do not have to be revealed.

c: Bob asks Alice to, for some $j \in [1, m]$, reveal A_j and x_{i_j}. He rejects if

$$A_j \neq \pi_j^{x_{i_j}}.$$

Remark: The sequence of B_i's may be thought of as a randomized version of the sequence of A_i's. For any choice of $R_1, \ldots, R_{m-1}$ (assuming $R_0 = R_m = I$, we have

$$\prod_{i=1}^{m} B_i = \prod_{i=1}^{m} A_i.$$

Furthermore, it is not hard to show that if $R_1, \ldots, R_{m-1}$ are distributed uniformly, then the sequence $B_1, \ldots, B_m$ will be distributed uniformly over all sequences with the given product.

We claim that this protocol constitutes a perfect zero-knowledge proof system for P. First we show that this is indeed a proof system.

Lemma 4.1. *If $P'(x_1, \ldots, x_{2n})$ does not hold, then Bob will reject with probability at least $1/3m$.*

Proof: To simplify our argument, we assume that all of the x_i's are defined, that is, Alice never produced any empty or defective envelopes. Clearly, Alice gains nothing by such a tactic. Now, if

1. $\prod_{i=1}^{m} B_i = a$,
2. $(\forall i \in [1, m]) B_i = R_{i-1} A_i R_i^{-1}$, and,
3. $(\forall j \in [1, m]) A_j = \pi_j^{x_{i_j}}$,

then we have,

$$\begin{aligned} \prod_{j=1}^{m} \pi_j^{x_{i_j}} &= \prod_{i=1}^{m} A_i \\ &= \prod_{j=1}^{m} B_j \quad \text{(by the above remark)} \\ &= a. \end{aligned}$$

Therefore, if $P'(x_1, \ldots, x_n)$ does not hold, one of the above three equalities must not hold. If equality (1) does not hold, then test (a) will detect always detect this fact. If equality (2) does not hold, then test (b) will detect this fact with probability at least $1/m$. If equality (3) does not hold, then test (c) will detect this fact with probability at least $1/m$. Since each test will be invoked with probability $1/3$, the lemma follows. ◇

It is not hard to see that our protocol achieves perfect zero-knowledge. Bob is only allowed to make a single test, either (a), (b), or (c). If he makes test (a), all he sees is a random sequence of elements whose product is a. Tests (b) and (c) allow Alice to get information about x_{i_j}, for some value of j. However, this will give him no information about any of the bits b_i, since each is represented as an exclusive-or of two of the x_i's.

4.3. *IP* in perfect zero-knowledge with envelopes

The notarized envelope scheme just described gives us zero-knowledge proofs for all of *IP*:

Theorem 4.2. *Assuming envelopes for bit commitment, every language that admits an interactive proof admits a perfect zero-knowledge interactive proof.*

Proof: Let language L be in IP. Let $(M \leftrightarrow A)$ be a one-sided Arthur-Merlin protocol for L, rejecting strings not in L with probability at least $1/3$. We assume without loss of generality that there exist polynomials r, l such that on input $x \in L$, where $|x| = n$,

1. The protocol $(M \leftrightarrow A)$ takes $r(n)$ rounds.
2. Each of Arthur's messages, and Merlin's responses are $l(n)$ bits long.
3. Arthur's decision predicate is in NC^1. (This property is not really necessary, but simplifies our proof slightly.)

We adopt the following notation. The string A_i denotes Arthur's ith message, and M_i denotes Merlin's ith response. We denote by a_j^i the jth bit of Arthur's ith message, and by m_j^i the jth bit of Merlin's ith message. We denote by $\mathcal{A}(x, A_1, \ldots, A_{r(n)}, M_1, \ldots, M_{r(n)})$ the decision predicate computed by Arthur at the end of the protocol. Given some vector $\vec{A} = A_1, \ldots, A_{r(n)}$, and input x, of length n, we define $\mathcal{A}_{\vec{A},x}(M_1, \ldots, M_{r(n)})$ to be equal to $\mathcal{A}(x, A_1, \ldots, A_{r(n)}, M_1, \ldots, M_{r(n)})$. Any circuit for $\mathcal{A}$ can be trivially transformed into a circuit for $\mathcal{A}_{\vec{A},x}$ without increasing its size or depth.

We now exhibit a modified protocol $(M' \leftrightarrow A')$ which uses envelopes. We claim that this protocol will be in perfect zero-knowledge, and will also be a one-sided "weak" proof system for L. By "weak " we mean that for any $x \notin L$, A' will accept with probability at most $1 - 1/|x|^c$, for some fixed c.

Protocol $(\mathbf{M}' \leftrightarrow \mathbf{A}')(x)$

1: For $i \in [1, r(n)]$ Merlin and Arthur execute the following two steps. Arthur sends Merlin a random string A_i. Merlin computes his answer, M_i, and runs protocol **commit**$(m_1^i, \ldots, m_{l(n)}^i)$. The commital protocol will generate $Q = 2r(n)l(n)$ bits, which we denote by $x_1, \ldots, x_Q$.

2: Let $\vec{A} = A_1, \ldots, A_{r(n)}$. The prover executes protocol **prove**$(x_1, \ldots, x_Q, \mathcal{A}_{\vec{A},x})$. A' accepts iff he doesn't reject in protocol **prove**.

This protocol is clearly in perfect zero-knowledge, since the commital and proof protocols are in perfect zero-knowledge. Since the protocol is one-sided, the prover will always be able to execute protocol **prove**, so this doesn't give any information.

To see that this protocol remains a "weak" proof, we note that if $x \notin L$, then with probability at least $1/3$, $\mathcal{A}_{\vec{A},x}(M_1, \ldots, M_{r(n)})$ will not hold. This is due to the definition of $\mathcal{A}_{\vec{A},x}$, and the fact that $(M \leftrightarrow A)$ is a proof system. In this case, there is some c, depending on $\mathcal{A}$, such that A' will reject during the **prove** protocol with probability at least $1/n^c$. Hence, if $x \notin L$, then A' will reject with probability at least $1/3n^c$. This probability of rejection may be made exponentially close to 1, maintaining both the one-sideness and the perfect zero-knowledge properties, by the standard trick of running the above protocol many times in succession. $\diamond$

It is interesting that the above proof never uses the ability to decommit notarized envelopes.

References

[AH] Adleman, L., and M. Huang, "Recognizing Primes in Random Polynomial Time," *Proceedings of the 19th STOC*, 1987, pp. 462–469.

[Bab] Babai, L., "Trading Group Theory for Randomness," *Proceedings of the 17th STOC*, 1985, pp. 421–429.

[Bar] Barrington, D., "Bounded Width Polynomial Size Branching Programs Recognize Exactly Those Languages in NC^1," *Proceedings of the 18th STOC*, 1986, pp. 1–5.

[BaM] Babai, L. and S. Moran, "Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes," manuscript.

[Bl] Blum, M., "Coin Flipping by Telephone," *IEEE COMPCON*, 1982, pp. 133–137.

[BHZ] Boppana, R., J. Håstad, and S. Zachos, "Does co-NP Have Short Interactive Proofs?", Information Processing Letters, 1987, pp. 127–132.

[Br] Brassard, G., *personal communication,* Augest 1988.

[CDG] Chaum, D., I. Damgård, and J. van de Graaf, "Multiparty Computations Ensuring Privacy of Each Party's Input and Correctness of the Result," Proceedings of Crypto-87, pp. 87–119.

[Fe] Feldman, P., "The Optimum Prover Lives in *PSPACE*," manuscript.

[Fo] Fortnow, L., "The Complexity of Perfect Zero-Knowledge," *Proceedings of the 19th STOC*, 1987, pp. 204–209.

[G] Goldreich, O., "Randomness, Interactive Proofs and Zero-Knowledge (a survey)," Technion Technical Report, 1987.

[GK] Goldwasser, S., and J. Kilian, "Almost All Primes Can Be Quickly Certified," *Proceedings of the 18th STOC*, 1986, pp. 316–329.

[GM] Goldwasser., S., and S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, Vol. 28, No. 2, 1984, pp. 270–299.

[GMR] Goldwasser, S., S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs," *Proceedings of the 17th STOC*, 1985, pp. 291–305

[GMS] Goldreich, M., Y. Mansour, and M. Sipser, "Interactive Proof Systems: Provers that Never Fail and Random Selection," *Proceedings of the 28th FOCS*, 1987, pp. 449–461.

[GMW1] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design," *Proceedings of the 27th FOCS*, 1986, pp. 174–187.

[GMW2] Goldreich, O., S. Micali, and A. Wigderson, "How to Play Any Mental Game, or, A Completeness Theorem for Protocols with Honest Majority," *Proceedings of the 19th STOC*, 1987, pp. 218–229.

[GS] Goldwasser, S., and M. Sipser, "Arthur Merlin Games versus Interactive Proof Systems," *Proceedings of the 18th STOC*, 1986, pp. 59–68.

[I] Impagliazzo, R., *personal communications*, 1987.

[IY] Impagliazzo, R., and M. Yung, "Direct Minimum-Knowledge Computations," Proceedings of Crypto-87, pp. 40–51.

[K1] Kilian, J., "Founding Cryptography on Oblivious Transfer," *Proceedings of the 20th STOC*, 1988, pp. 20–31.

[K2] Kilian, J., "Primality Testing and the Cryptographic Complexity of Noisy Communications Channels," MIT Ph.D. Thesis (in preparation), 1988.

[L] Levin, L., "One-way Functions and Pseudorandom Generators," *Proceedings of the 17th STOC,* 1985, pp. 363–368.

[MRS] Micali, S., C. Rackoff and R. Sloan, "The Notion of Security for Probabilistic Cryptosystems," *SIAM Journal of Computing*, 17(2):412–426, April 1988.

[O] Oren, Y., "On the Cunning Power of Cheating Verifiers: Some Observations about Zero Knowledge Proofs," *Proceedings of the 28th FOCS*, 1987, pp. 462–471.

[TW] Tompa, M., and H. Woll, "Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information," *Proceedings of the 28th FOCS*, 1987, pp. 472–482.

[Ya] Yao, A.C., "Theory and Applications of Trapdoor Functions," *Proceedings of the 23rd FOCS*, 1982, pp. 80–91.

[Yu] Yung, M., *personal communication*, Augest 1988.