Zero-Knowledge Simulation of Boolean Circuits

Gilles BRASSARD [†] and *Claude CREPEAU* [‡]

Département d'informatique et de R.O.
Université de Montréal
C.P. 6128, Succursale "A"
Montréal (Québec)
Canada H3C 3J7

ABSTRACT

A zero-knowledge interactive proof is a protocol by which Alice can convince a polynomially-bounded Bob of the truth of some theorem without giving him any hint as to how the proof might proceed. Under cryptographic assumptions, we give a general technique for achieving this goal for *every* problem in **NP**. This extends to a presumably larger class, which combines the powers of non-determinism and randomness. Our protocol is powerful enough to allow Alice to convince Bob of theorems for which she does not even have a proof: it is enough for Alice to convince herself probabilistically of a theorem, perhaps thanks to her knowledge of some trap-door information, in order for her to be able to convince Bob as well, without compromising the trap-door in any way.

1. INTRODUCTION

Assume that Alice holds the proof of some theorem. A *zero-knowledge interactive proof* (ZKIP) is a protocol that allows her to convince a polynomially bounded Bob that she owns such a proof, in a way that he will gain *nothing* else than this conviction: engaging in the protocol with Alice gives Bob no hint on Alice's proof, or at least nothing he can make use of in polynomial time. In particular, it does not enable him to later convince anyone else that Alice has a proof of the theorem or even merely that the theorem is true (much less that he himself has a proof!). This notion was introduced by Goldwasser, Micali and Rackoff [GMR]; the reader is refered to this paper for formal definitions. An intuitive notion of ZKIP suffices to understand this extended abstract.

The early examples of ZKIP's were all number theoretic and restricted to problems in **NP** $\cap$ co-**NP** [GMR, GHY]. It was conjectured by Silvio Micali, and believed by most researchers, that such protocols could not exist for **NP**-complete problems. Under cryptographic assumptions, we show here that this intuition was wrong by providing a ZKIP for satisfiability. The same result was obtained independently and slightly earlier by [GMW] as they gave a ZKIP for graph 3-colouring. Obviously (because Karp reductions carry **NP** certificates), it suffices to find a ZKIP for any **NP**-complete problem in order to get one for every problem in **NP**. Protocols very similar to ours for satisfiability are also given in [Be, Ch]. Our protocol is more attractive in practice than that of [GMW], but we depend on a specific cryptographic assumption (quadratic residuosity) whereas they merely need to assume the existence of secure encryption schemes in the sense of [GM].

† Supported in part by NSERC grant A4107.
‡ Supported in part by an NSERC postgraduate scholarship; current address: M.I.T.

ZKIP's are conceivable even if Alice does not have a proof to start with. Let us assume that she merely has a *convincing argument* that the theorem is true. In this case, she might wish to convince Bob of the theorem with a level of confidence comparable to her own. This *transfer of confidence* is *zero-knowledge* if it does not provide a polynomially-bounded Bob with any information on the argument itself, except for its existence and Alice's knowledge of it. Our main result is that such protocols exists for a class of problems probably more extensive than **NP**.

To illustrate the ideas, let us assume that Alice wishes to convince Bob that some integer m (of her choosing) is the product of exactly k distinct primes. Alice is convinced of the truth of her claim because she randomly selected k distinct integers $p_1, p_2, \cdots, p_k$ that passed some probabilistic primality test [R, SS] to her satisfaction. Although proofs of primality for these factors exist since PRIMES $\in$ **NP** [Pr], there is no known feasible algorithm for Alice to get these proofs[1]. In other words, Alice knows (with an arbitrary small probability of error) that m is in the proper form, she knows there exists a short proof of this statement, but she cannot find the proof. Using our protocol, she can nonetheless convince Bob without compromising the factorization of m in any way (except for the fact that Bob will know the number of factors).

The above example illustrates the fact that our model does not assume that Alice has more computing power than Bob nor access to some oracle. Although she starts with one piece of additional knowledge (either a formal proof of some theorem or merely a convincing argument), this may be the result of her using trap-door information. The entire protocol itself can be carried out with polynomial time resources.

The general technique allows Alice to guide Bob through the simulation of an arbitrary Boolean circuit without ever having to disclose its inputs or any intermediary results. At the end of the protocol, she can nonetheless convince Bob of the final outcome of the circuit. If this turns out to be 1, Bob will be convinced that the Boolean function computed by the circuit is satisfiable and that Alice holds a satisfying assignment, but he will known nothing else. The bottom line is that, whenever Alice can convince herself probabilistically of a fact or theorem, perhaps thanks to her knowledge of some trap-door information, she can convince Bob as well *without compromising the trap-door.*

2. NUMBER THEORETIC BACKGROUND

Let n be an integer. $\mathbb{Z}_n^*$ denotes the set of integers relatively primes to n between 1 and $n-1$. An integer $z \in \mathbb{Z}_n^*$ is a *quadratic residue* modulo n ($z \in \mathrm{QR}_n$) if there is an $x \in \mathbb{Z}_n^*$ such that $z \equiv x^2 \pmod{n}$. An integer $z \in \mathbb{Z}_n^*$ is a *quadratic non-residue* modulo n ($z \in \mathrm{QNR}_n$) if $z \notin \mathrm{QR}_n$. If p is a prime and if $z \in \mathbb{Z}_p^*$, it is easy to determine whether $z \in \mathrm{QR}_p$ because this is so if and only if $z^{(p-1)/2} \equiv 1 \pmod{p}$. Let $n = pq$ be the product of two distinct odd primes. Given $z \in \mathbb{Z}_n^*$, let z_p and z_q denote (z **mod** p) and (z **mod** q), respectively. Given the factorization of n, it is easy to determine whether $z \in \mathrm{QR}_n$ because this is so if and only if $z_p \in \mathrm{QR}_p$ and $z_q \in \mathrm{QR}_q$. Given the factorization of n and given $z \in \mathrm{QR}_n$, it is also easy (by a Las Vegas algorithm in general [Pe]) to find every $x \in \mathbb{Z}_n^*$ such that $z \equiv x^2 \pmod{n}$. This is however believed to be hard without the factorization of n.

[1] Goldwasser and Kilian's new *provably correct and probably fast primality test* [GK] allows Alice to "efficiently" (the running time is currently a 12th power polynomial) get short proofs for those primes on which the algorithm turns out to be fast. This might reduce the interest of this particular example, but not the interest of our general protocol.

Given $z \in \mathbb{Z}_n^*$, the Jacobi symbol (z/n) is defined as +1 if both z_p and z_q are quadratic residues modulo p and q, respectively, or if both are quadratic non-residues; it is defined as −1 otherwise. It is easy to compute (z/n) even if the factorization of n is unknown [RSA]. Let $\mathbb{Z}_n^*[+1]$ denote the set of $z \in \mathbb{Z}_n^*$ such that $(z/n) = +1$ and define $\mathbb{Z}_n^*[-1]$ similarly. Let $\text{QNR}_n[+1]$ denote $\text{QNR}_n \cap \mathbb{Z}_n^*[+1]$. It is clear that $\mathbb{Z}_n^*[-1] \subset \text{QNR}_n$; moreover, exactly half the members of $\mathbb{Z}_n^*[+1]$ are quadratic residues modulo n and the other half are quadratic non-residues. Both $\mathbb{Z}_n^*[+1]$ and QR_n are closed under multiplication modulo n, the product modulo n of two members of $\text{QNR}_n[+1]$ is a member of QR_n, and the product modulo n of a member of QR_n by a member of $\text{QNR}_n[+1]$ is a member of $\text{QNR}_n[+1]$. A uniformly distributed random element of QR_n can be obtained by randomly choosing some $x \in \mathbb{Z}_n^*$ and squaring it modulo n; given any fixed $y \in \text{QNR}_n[+1]$, a uniformly distributed random element of $\text{QNR}_n[+1]$ can be obtained by randomly choosing some $x \in \mathbb{Z}_n^*$ and computing $x^2 y \bmod n$. Furthermore, everything we have said so far, except for the definition of the Jacobi symbol, remains true if n is of the form $p^i q^j$, where p and q are distinct odd primes and i and j are positive powers of which at least one is odd.

It is believed that no efficient algorithm can distinguish a quadratic residue from a quadratic non-residue, even probabilistically speaking, as long as the latter has Jacobi symbol +1 and the factorization of n is unknown. For a more formal statement of this quadratic residuosity assumption (QRA) and for more background on number theory, please refer to [GM].

3. THE ENCRYPTION OF SECRETS

At the beginning of our protocols, Alice randomly chooses two distinct large primes p and q, and she discloses their product $n = pq$ to Bob. Following the QRA, we assume throughout that Bob cannot distinguish a quadratic residue modulo n from a quadratic non-residue, as long as the latter belongs to $\mathbb{Z}_n^*[+1]$. Alice also randomly chooses and discloses to Bob some $y \in \text{QNR}_n[+1]$. (It is proven in [GM] that this cannot help Bob distinguish residues from non-residues.) Using the zero-knowledge interactive protocol of [GHY], Alice convinces Bob that n is of the form $p^i q^j$ for distinct odd primes p and q, and positive integers i and j of which at least one is odd[2]. Using the zero-knowledge protocol of [GMR], Alice convinces Bob that $y \in \text{QNR}_n[+1]$.

At this point, Bob could produce uniformly distributed random members of QR_n and $\text{QNR}_n[+1]$ by choosing a random $x \in \mathbb{Z}_n^*$ and computing either $x^2 \bmod n$ or $x^2 y \bmod n$. The fact that only Alice can distinguish between these two occurrences was the basis of Goldwasser and Micali's original probabilistic encryption [GM]. Here, we use this idea *in the reverse direction*: it will *always* be Alice that produces random members of QR_n and $\text{QNR}_n[+1]$. By convention, members of QR_n are used as encryptions of the bit 0 and members of $\text{QNR}_n[+1]$ are used as encryptions of the bit 1. Whenever Alice shows Bob the encryption z of some bit b, he has no clue as to which bit it encodes (under QRA). It is however possible for Alice to prove to Bob whether $b = 0$ or $b = 1$ by showing him some $x \in \mathbb{Z}_n^*$ such that $z = x^2 y^b \bmod n$. This operation will be refered to as *opening the secret* z. Notice that this is a zero-knowledge proof even though a square root of either z or zy^{-1} is given to Bob, because x was randomly chosen by Alice. For this reason, whenever she wishes to

[2] It would be nicer if Alice could convince Bob directly that n is of the form pq, but we offer in the sequel the first ZKIP capable of achieving this (and therefore we cannot use it yet). This is however of no consequence because Alice could only make herself more vulnerable by choosing $n = p^i q^j$ without $i = j = 1$.

open a secret z, there is no need for Alice to use the ZKIP of [GMR] in order to convince Bob of which among zy^{-1} or z belongs to $QNR_n[+1]$. We give in the last section of this paper a simplified ZKIP for quadratic residuosity when the target is chosen by Bob.

4. CAN BOB COMPUTE ON ENCRYPTED BITS?

Let b_1 and b_2 be two secret bits of Alice, and let z_1 and z_2 be their encryptions as given to Bob. Even though Bob has no knowledge of b_1 or b_2, he can still compute an encryption of some functions of b_1 and b_2. For instance, Bob can compute $z_1 y \bmod n$, which is an encryption for the negation of b_1. Similarly, Bob can compute $z_1 z_2 \bmod n$, which is an encryption of the exclusive-or of b_1 and b_2 because if $z_1 = x_1^2 y^{b_1} \bmod n$ and $z_2 = x_2^2 y^{b_2} \bmod n$, then

$$z_1 z_2 \bmod n = (x_1 x_2)^2 y^{b_1+b_2} \bmod n = x^2 y^{(b_1+b_2) \bmod 2} \bmod n,$$

where $x = x_1 x_2 y^{(b_1+b_2) \operatorname{div} 2} \bmod n$.

Could Bob compute an encryption of the **and** or the **or** of b_1 and b_2 given only z_1 and z_2? This remains an open question. We will show, however, that it *is* possible for Bob to do so *with the* (zero-knowledge) *help of Alice*. As a corollary, Bob can compute an encryption of arbitrary Boolean functions of bits for which he only has encryptions. After this computation, Alice can open the result for Bob without ever having had to open the input Boolean variables or any intermediary information. This idea leads to a simple ZKIP for SAT in Section 6.

5. HOW ALICE CAN HELP BOB COMPUTE ON ENCRYPTED BITS

Let $u = b_1 b_2 \cdots b_k$ be a k-bit string of Alice. For each i, $1 \le i \le k$, let z_i and $\hat{z}_i$ be two encryptions of b_i randomly chosen by Alice. It is easy for Alice to convince Bob that the k-bit strings encrypted by $z_1 z_2 \cdots z_k$ and $\hat{z}_1 \hat{z}_2 \cdots \hat{z}_k$ are identical without providing Bob with any additional information.

String equality protocol: For each i, $1 \le i \le k$, Alice gives Bob some $x_i \in \mathbf{Z}_n^*$ such that $z_i \hat{z}_i \equiv x_i^2 \pmod n$. Once again, this is a ZKIP because the encryptions were randomly chosen by Alice and not influenced by Bob. □

As above, let $u = b_1 b_2 \cdots b_k$ and let z_i encrypt b_i for each i, $1 \le i \le k$. Now, let $\hat{u} = \hat{b}_1 \hat{b}_2 \cdots \hat{b}_k$ be some k-bit string *different* from u and let $\hat{z}_i$ be an encryption of $\hat{b}_i$ for each i, $1 \le i \le k$. It is no longer so obvious that Alice can convince Bob that the strings encrypted by $z_1 z_2 \cdots z_k$ and $\hat{z}_1 \hat{z}_2 \cdots \hat{z}_k$ are different without yielding some additional information (such as a specific i for which $b_i \ne \hat{b}_i$). The fact that this is possible, and the technique that achieves this protocol, illustrate the core of our main result.

String inequality protocol: For each i, $1 \le i \le k$, let $v_i = z_i \hat{z}_i \bmod n$. The problem reduces to convincing Bob (by a ZKIP) that the string encrypted by $v_1 v_2 \cdots v_k$ is not identically zero. For this, Alice randomly chooses some permutation σ of $\{1, 2, \cdots, k\}$ and $x_i \in \mathbf{Z}_n^*$ for $1 \le i \le k$. She then computes and discloses to Bob $w_i = x_i^2 v_{\sigma(i)} \bmod n$ for each $1 \le i \le k$. At this point, Bob sends either challenge A or challenge B to Alice.

- If Bob sent challenge A, Alice must disclose some i such that w_i encrypts a 1, and open this w_i for Bob by giving him a square root of $w_i y^{-1}$ modulo n.

- If Bob sent challenge B, Alice must disclose the permutation σ and use the string equality protocol to convince Bob that $w_1 w_2 \cdots w_k$ encrypts the same string as $v_{\sigma(1)} v_{\sigma(2)} \cdots v_{\sigma(k)}$.

This process is repeated s times, for some safety parameter s agreed upon between Alice and Bob. In order to convince Bob, Alice must meet every single challenge. □

Theorem

(i) The only knowledge obtainable by Bob from this protocol is that $z_1 z_2 \cdots z_k$ and $\hat{z}_1 \hat{z}_2 \cdots \hat{z}_k$ encrypt distinct bit strings, and

(ii) Alice only has a probability 2^{-s} of convincing Bob of this when in fact the strings are identical.

Proof (sketch).

(i) Observe that whenever Bob chooses challenge A, he learns that the original bit strings are distinct in at least one place (if Alice was honest), but this gives him no clue as to any single i such that $b_i \neq \hat{b}_i$ because the permutation σ is then kept secret. On the other hand, whenever Bob chooses challenge B, he gains no information whatsoever on the original strings.

(ii) If in fact $v_1 v_2 \cdots v_k$ encrypts the identically zero string, the only thing Alice can do to hope convincing Bob of the contrary is to guess *exactly* which challenge Bob will choose for each round and to encrypt non-identically zero strings with $w_1 w_2 \cdots w_k$ whenever she expects Bob to use challenge A and identically zero strings otherwise. The results follows from the fact that there are 2^s equally likely sequences of choices for Bob. □

We are now ready for the main tool used in this paper. Consider any Boolean function $B : \{0,1\}^t \to \{0,1\}$ agreed upon between Alice and Bob, and any bits $b_1, b_2, \cdots, b_t$ known to Alice only. For $1 \leq i \leq t$, let z_i be an encryption of b_i known to Bob. Let $b = B(b_1, b_2, \cdots, b_t)$. Alice can produce an encryption z for b *and convince Bob that z encrypts the correct bit* without giving him any information on the input bits $b_1, b_2, \cdots, b_t$ nor on the result b.

Definition. A *permuted truth table* for the Boolean function B is a binary string of length $(t+1)2^t$ formed of 2^t blocks of $t+1$ bits. The last bit of each block is the value of B on the other t bits of the block, and each assignment of truth values occurs exactly once in the first t bits of some block. For example, here is a permuted truth table for the binary **or**: 011000111101, which should be read as 0 **or** 1 = 1, 0 or 0 = 0, 1 **or** 1 = 1 and 1 **or** 0 = 1.

Boolean computation protocol: Let the situation be as in the paragraph just before the above definition. Alice randomly chooses a permuted truth table for B and she discloses encryptions for each of its bits. At this point, Bob sends either challenge A or challenge B to Alice.

- If Bob sent challenge A, Alice must open the entire encryption of the permuted truth table, so that Bob can check that it is a valid truth table for B.
- If Bob sent challenge B, Alice must point out to the appropriate block in the encryption of the permuted truth table and use the string equality protocol to convince Bob that $z_1 z_2 \cdots z_t z$ encrypts the same bit string as this block.

This process is repeated s times, for some safety parameter s agreed upon between Alice and Bob. In order to convince Bob that z is an encryption for $B(b_1, b_2, \cdots, b_t)$, Alice must succeed in meeting every single challenge. □

A theorem very similar to the one for the string inequality protocol can be stated and the proof is essentially identical. Notice that this protocol is interesting only for small t because it is exponential in t. In the sequel, we will use it *exclusively* with $t \leq 2$. A very similar Boolean computation protocol was discovered independently by Josh Benaloh [Be] as an application of the general tool of "cryptographic capsules" [CF].

6. ZKIP FOR SAT

The zero-knowledge interactive proof for satisfiability should now be obvious. Let $f: \{0,1\}^k \to \{0,1\}$ be the function computed by some satisfiable Boolean formula for which Alice knows an assignment $b_1, b_2, \cdots, b_k \in \{0,1\}$ such that $f(b_1, b_2, \cdots, b_k) = 1$. Assume the Boolean formula is given using arbitrary unary and binary Boolean operators. In order to convince Bob that the formula is satisfiable, Alice produces encryptions $z_1, z_2, \cdots, z_k$ of $b_1, b_2, \cdots, b_k$, respectively. She then guides Bob through the encrypted evaluation of the formula, one Boolean operator at a time[3], using the Boolean computation protocol (with $t \leq 2$). This results is an encryption z for the value of $f(b_1, b_2, \cdots, b_k)$. It then only remains for Alice to open z and show Bob that it encrypts a 1.

7. ZKIP FOR THE NUMBER OF PRIME FACTORS

Let us now come back to the problem mentioned in the introduction. Alice has selected k distinct primes $p_1, p_2, \cdots, p_k$ and she has formed their product $m = p_1 p_2 \cdots p_k$. She wishes to convince Bob that m is indeed the product of exactly k distinct primes. Let l be the number of bits in m. Each factor will be considered as a length l binary string, with leading zeroes if needed. As a first step, Alice encrypts each of the factors and she discloses these encryptions to Bob. The string inequality protocol is used to convince Bob that the factors are all distinct and that none of them is equal to 1. She then guides Bob through the simulation of a Boolean circuit for iterated multiplication. This produces the encryption of a length kl bit string, which Alice opens to show that it encrypts $(k-1)l$ zeroes followed by the binary representation of m.

At this point, Alice still has to convince Bob that each of these factors is a prime. If she had a proof of this, she could encode it as the input to a proof verification Boolean circuit and guide Bob through its evaluation. Recall, however, that her conviction that each of the p_i is prime comes from her own running of a probabilistic primality test. None of these runs can be considered as convincing by Bob because he cannot trust that Alice was honest in her coin tosses.

This is where our technique is most powerful. Consider a Boolean circuit with two l-bit inputs p and c that outputs 1 if and only if c is a certificate that p is composite (where primes have no certificates and composites have lots [R, SS]). Recall that Bob was given by Alice an encryption of each bit of each p_i. With the help of Alice, he can run as many randomly chosen c's as he wishes into the circuit for each p_i and ask her to open the circuit outcomes. If he ever gets a 1, he will know for sure that the corresponding p_i is composite and that Alice had been cheating (or perhaps that

[3] To save on the number of communications rounds, the various operators can be processed in parallel.

Alice was honest after all, and that she just discovered with him that this p_i is composite!). Otherwise, since he has complete control over the c's, he can convince himself, with any level of confidence, that m is the product of exactly k distinct primes. This protocol can be adapted if Alice wished instead to convince Bob that there are exactly k distinct primes in the factorization of m, regardless of their multiplicities. A more practical variation allows Alice to convince Bob that the prime factors of n have interesting properties, such as being of the form $2q+1$, where q is also a prime.

8. THE GENERAL PROTOCOL

Recall that **BPP** stands for the class of decision problems that can be solved in probabilistic polynomial time with bounded error probability [G]. It is reasonable to consider **BPP** as the *real* class of tractable problems (rather than **P**) because the error probability can always be decreased below any $\varepsilon > 0$ by repeating the algorithm $c \log \varepsilon^{-1}$ times and taking the majority answer, where c depends only on the original error probability. It is generally believed that there is no inclusion relation either way between **NP** and **BPP**: non-determinism and randomness seem to be incomparable powers. These powers can be combined in several ways. We believe the most natural to be Babai's class **MA** [Ba], which we would rather call **RNP** as *random* **NP**. This class is such that NP $\cup$ **BPP** $\subseteq$ **RNP**, hence **NP** is almost certainly a strict subset of **RNP**. For a discussion as to why we favour **MA** over the seemingly more powerful **AM** or interactive proof systems [GMR], please consult [BC].

Definition. Let Σ stand for $\{0,1\}$. A decision problem $X \subseteq \Sigma^*$ belongs to **RNP** if and only if there exists a predicate $A \subseteq \Sigma^* \times \Sigma^*$ and a polynomial $p(n)$ such that

(i) $A \in$ **BPP**, and

(ii) $(\forall x \in \Sigma^*)[x \in X \Leftrightarrow (\exists a \in \Sigma^*)[|a| = p(|x|)$ **and** $<x,a> \in A]]$
(such an a is refered to as an *argument* for x). □

Notice that this would correspond to the polynomial hierarchy characterization of **NP** had we insisted that $A \in$ **P**. The restriction $|a| = p(|x|)$ instead of the usual $|a| \le p(|x|)$ is there for a technical reason. Notice also that $X \in$ **NP** whenever $A \in$ **NP**.

Intuitively, $X \in$ **RNP** means that whenever $x \in X$, there is a (possibly hard to find) short argument for this, and that the validity of this argument can be checked probabilistically in polynomial time. We are about to prove that if $X \in$ **RNP**, if the proof that $X \in$ **RNP** is in the public domain, and if Alice knows an argument a for some $x \in X$, she can convince Bob with a ZKIP that $x \in X$. As a warm up, let us first restrict ourselves to one-sided probabilistic algorithms.

Recall that **RP** (sometimes refered to as **R**) is the class of decision problems that can be solved in polynomial time by a *one-sided* bounded error probabilistic algorithm [A]. Here, each time the probabilistic algorithm is run on a yes-instance, it accepts with probability at least ½, whereas it always rejects no-instances. It is well known that **RP** $\subseteq$ NP $\cap$ **BPP** and that co-**RP** $\subseteq$ **BPP**, but co-**RP** and **NP** are probably incomparable. Whenever x is a yes-instance of a co-**RP** problem, one can convince him/herself that this is so (by repeating the algorithm), but there does not have to exist a succinct proof of this.

Theorem (under QRA). Consider a problem $X \in$ **RNP** such that the corresponding A (refer to the definition of **RNP**) belongs to co-**RP**. Assume that the characterization A for X and a co-**RP** algorithm for A are in the public domain. Let Alice have an argument a for some $x \in X$. Although she may not have a definite proof that $x \in X$, she convinced herself probabilistically that $<x,a> \in A$, hence $x \in X$. It is then possible for Alice to convince Bob in polynomial time that $x \in X$ without disclosing any additional information.

Proof (sketch). Alice and Bob agree on a probabilistic one-sided Boolean circuit for the complement of A. (That is: on any yes-instance of A, using any random choices, the circuit outputs a 0; on any no-instance of A, the circuit outputs a 1 for at least 50% of the random choices.) Alice gives Bob an encryption for each bit of x, and she opens them to show that they encrypt x. Alice also gives Bob an encryption for each bit of a, but she keeps a itself secret. She then guides Bob through the evaluation of the Boolean circuit on input $<x,a>$, using Bob's coin tosses, until the encrypted outcome is obtained. She then opens this outcome to Bob, who can ascertain that it is indeed a 0. This process is repeated until Bob is convinced that $<x,a> \in A$, hence that $x \in X$. Clearly, this gives Bob no information on a (except for its mere existence and Alice's knowledge of it) because the *only* possible outcome for the Boolean circuit is 0, provided Alice was not trying to cheat. Bob does not even learn the length of a because it had to be exactly $p(|x|)$ by definition of **RNP**. □

The above protocol does *not* work directly for $X \in$ **RNP** in general, because it would not be zero-knowledge. Indeed, Bob would gain information on Alice's argument a from knowledge of which random choices made the circuit accept $<x,a>$ and which made it reject, or even merely from knowledge of the number of each of these occurrences. (Recall that if $A \in$ **BPP** but $A \notin$ **RP** $\cup$ co-**RP**, the probabilistic Boolean test circuit for A is expected to output sometimes 0 and sometimes 1 on the same input; the most frequent answer being correct with high probability.) Two ideas are needed to solve this difficulty:

- Alice and Bob agree in advance on the number of runs they wish to carry through the test circuit (depending on the error probability they are willing to tolerate). At the end of each run, Alice no longer opens the outcome. After all the runs are completed, Alice guides Bob through the evaluation of a *majority* Boolean circuit, using the previously obtained encrypted outcomes as input. It is only the resulting majority bit that Alice finally opens for Bob.
- Even if Alice is honest, the above idea leaves the door open for Bob to cheat: it could be that the circuit outcome is not what she expected because Bob had deliberately chosen the "random" coin tosses to make this occurrence 50% likely. Assuming Alice's good faith, this could yield up to one bit of information to Bob about the argument a, which is intolerable. Alice would be almost certain that Bob cheated, but it would be too late by then. In order to prevent this possibility, it is essential that all coins be tossed so that neither Alice nor Bob can influence the outcome, and such that Bob does not get to see the outcome (i.e.: coin tossing in a well). Fortunately, such a protocol is very simple: to toss a coin, Alice gives Bob a randomly chosen element of $\mathbf{Z}_n^*[+1]$ and Bob tells her whether to multiply it or not by the standard $y \in \mathrm{QNR}_n[+1]$.

Main Theorem (under QRA). Consider any $X \in \mathbf{RNP}$ and some $x \in X$ for which Alice knows an argument a. Assume the proof that $X \in \mathbf{RNP}$ is in the public domain[4]. Even though Alice may not have a definite proof that $x \in X$, she convinced herself probabilistically that $<x,a> \in A$, hence $x \in X$. It is possible for Alice to convince Bob in polynomial time that $x \in X$ and that she knows some argument for this without disclosing any additional information.

Proof (sketch). By the above discussion. $\square$

Let us stress again that this protocol is interesting even when $A \in \mathbf{NP}$, hence $X \in \mathbf{NP}$ (as in section 7 because PRIMES $\in$ **NP**), despite the reduction to SAT in these cases. This is so because Alice could know the argument a for x as a result of her choosing a in the first place (as trap-door information) and producing x from it. She might not, however, have an accepting computation for $<x,a>$, even though $A \in \mathbf{NP}$. She can nonetheless make use of our protocol. In other words, it does not require Alice to have more computing power than Bob or to have access to some **NP**-complete oracle. **As long as she can convince herself with the help of her own trap-door, she can convince Bob as well without compromising the trap-door.**

9. OTHER EXAMPLES OF ZKIP's

Our basic technique can be used in various situations. Let us briefly mention a few of them. It allows Alice to convince Bob of the quadratic residuosity of a member of $\mathbf{Z}_n^*[+1]$ *chosen by Bob* without yielding additional information, in a way much simpler than those of [GMR, GHY]. It also allows Alice to convince Bob that an encrypted function is a permutation (see below). More generally, all these building blocks can be used directly to obtain *efficient* ZKIP's for a variety of NP-complete problems such as Hamiltonian circuit, clique, knapsack, graph 3-colouring, etc.

Quadratic Residuosity Protocol: Bob shows some $z \in \mathbf{Z}_n^*[+1]$ to Alice and she is willing to convince him of whether it is a quadratic residue or not. Assume initially that $z \in QR_n$. Alice uses her knowledge of the factors of n to compute some $x \in \mathbf{Z}_n^*$ such that $z = x^2 \bmod n$. Because z was chosen by Bob, it would be far from a ZKIP if Alice revealed x to Bob as proof (it could give Bob a 50% chance of factoring Alice's master secret n). Instead, Alice randomly generates some $u \in \mathbf{Z}_n^*$. She then computes and discloses $w = u^2 \bmod n$. At this point, Bob sends either challenge A or challenge B to Alice.

- If Bob sent challenge A, Alice must disclose u so that Bob can check that $w = u^2 \bmod n$, hence that w is a quadratic residue.
- If Bob sent challenge B, Alice must disclose $ux \bmod n$ so that Bob can check that $(ux)^2 \equiv wz \pmod n$, hence that w has the same quadratic character as z.

This process is repeated s times for some safety parameter s agreed upon between Alice and Bob. The protocol is very similar if $z \notin QR_n$ but it requires that some standard $y \in QNR_n[+1]$ has already been proven once and for all. Thus, the protocol of [GMR] must be used *the very first time* in order to make ours effective. $\square$

[4] i.e.: the predicate A and the BPP algorithm for A are already known to Bob.

A similar protocol is independently given in [Be]; its essence was already in [CF]. Notice also that our protocol would *not* work for quadratic non-residuosity if n had more than two distinct prime factors, whereas the protocol of [GMR] could still be used.

Finally, here is the permutation problem. Let m be some integer agreed upon between Alice and Bob. Let σ be a permutation of $\{1, 2, \cdots, m\}$ randomly and secretly chosen by Alice. This permutation can be naturally represented by a table of mk bits, where $k = \lceil \log_2 m \rceil$. Alice discloses to Bob an encryption for each of these bits, so that it will not be possible for her to change her originally chosen permutation. At this point, Bob would like to be convinced that he was given the encryption of a permutation, not just of any function from $\{1, 2, \cdots, m\}$ to $\{1, 2, \cdots, 2^k\}$. No doubt the reader has seen our technique used enough times by now to design his/her own ZKIP. This problem has applications if one wishes to keep an electronic poker face [Cr], and its solution is central to the above mentioned efficient ZKIP's for Hamiltonian circuit, clique, knapsack, etc.

10. OPEN PROBLEM

Can Bob compute encryptions of arbitrary Boolean functions of encrypted Boolean inputs *without the help of Alice*? For instance, given encryptions for the bits b_1 and b_2, can he compute an encryption for (b_1 **and** b_2)? If so, this might allow a dramatic improvement in our protocols, including the possibility of *publishing* ZKIP's (an idea originally investigated by Manuel Blum).

ACKNOWLEDGEMENT

We wish to thank Josh Benaloh, Joan Feigenbaum, Oded Goldreich, Shafi Goldwasser, Silvio Micali, Jean-Marc Robert, Steven Rudich and Moti Yung for fruitful discussions.

REFERENCES

[A] Adleman, L., "Reducibility, randomness and intractability", *Proceedings of the 9th Annual ACM Symposium on the Theory of Computing*, 1977, pp. 151-163.

[Ba] Babai, L., "Trading group theory for randomness", *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing*, 1985, pp. 421-429.

[Be] Benaloh (Cohen), J. D., "Cryptographic capsules: a disjunctive primitive for interactive protocols", *these CRYPTO 86 Proceedings*, Springer-Verlag, 1987.

[BC] Brassard, G. and C. Crépeau, "Non-transitive transfert of confidence: a *perfect* zero-knowledge interactive protocol for SAT and beyond", *Proceedings of the 27th Annual IEEE Symposium on the Foundations of Computer Science*, 1986, pp. 188-195.

[Ch] Chaum, D., "Demonstrating that a public predicate can be satisfied without revealing any information about how", *these CRYPTO 86 Proceedings*, Springer-Verlag, 1987.

[CF] Cohen (Benaloh), J. D. and M. J. Fisher, "A robust and verifiable cryptographically secure election scheme", *Proceedings of the 26th Annual IEEE Symposium on the Foundations of Computer Science*, 1985, pp. 372-382.

[Cr] Crépeau, C., "A zero-knowledge Poker protocol that achieves confidentiality of the players' strategy, *or* How to achieve an electronic Poker face", *these CRYPTO 86 Proceedings*, Springer-Verlag, 1987.

[GHY] Galil, Z., S. Haber and M. Yung, "A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems", *Proceedings of the 26th Annual IEEE Symposium on the Foundations of Computer Science*, 1985, pp. 360-371.

[G] Gill, J. ''Computational complexity of probabilistic Turing machines'', *SIAM Journal on Computing*, vol. 6, no. 4, 1977, pp. 675-695.

[GMW] 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 Annual IEEE Symposium on the Foundations of Computer Science*, 1986, pp. 174-187.

[GK] Goldwasser, S. and J. Kilian, ''Almost all primes can be quickly certified'', *Proceedings of the 18th Annual ACM Symposium on the Theory of Computing*, 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, "The knowledge complexity of interactive proof-systems'', *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing*, 1985, pp. 291-304.

[Pe] Peralta, R., ''A simple and fast probabilistic algorithm for computing square roots modulo a prime number'', *IEEE Transactions on Information Theory*, to appear.

[Pr] Pratt, V., ''Every prime has a succinct certificate'', *SIAM Journal on Computing*, vol. 4, 1975, pp. 214-220.

[R] Rabin, M. O., ''Probabilistic algorithms'', in *Algorithms and Their Complexity: Recent Results and New Directions*, J. F. Traub (editor), Academic Press, New York, New York, 1976, pp. 21-39.

[RSA] Rivest, R. L., A. Shamir and L. Adleman, ''A method for obtaining digital signatures and public-key cryptosystems'', *Communications of the ACM*, vol. 21, no. 2, 1978, pp. 120-126.

[SS] Solovay, R. and V. Strassen, ''A fast Monte Carlo test for primality'', *SIAM Journal on Computing*, vol. 6, 1977, pp. 84-85.