Non-Transitive Transfer of Confidence: A *Perfect* Zero-Knowledge Interactive Protocol for SAT and Beyond

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 *perfect* zero-knowledge interactive proof is a protocol by which Alice can convince Bob of the truth of some theorem in a way that yields no information as to how the proof might proceed (in the sense of Shannon's information theory). We give a general technique for achieving this goal for any problem in **NP** (and beyond). **The fact that our protocol is perfect zero-knowledge does not depend on unproved cryptographic assumptions.** Furthermore, our protocol is powerful enough to allow Alice to convince Bob of theorems for which she does not even have a proof. Whenever Alice can convince herself probabilistically of a theorem, perhaps thanks to her knowledge of some trap-door information, she can convince Bob as well, without compromising the trap-door in any way. This results in a non-transitive transfer of confidence from Alice to Bob, because Bob will not be able to convince anyone else afterwards. Our protocol is dual to those of [GrMiWi86a, BrCr86].

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 in [GwMiRac85] with examples of such protocols in the realm of number theory. Other number theoretic ZKIP's are given in [GaHaYu85]. If, in addition, the protocol contains no *information* on Alice's proof, in the sense of classical information theory, the ZKIP is said to be *perfect* zero-knowledge [GrMiWi86a]. In essence, this is so if the entire discussion between Alice and Bob could have been efficiently simulated by Bob alone, in the sense of probability distribution, provided the theorem is true. For applications of these notions in the realm of cryptographic protocol design, please consult [GrMiWi86a, b]. Although an intuitive notion of ZKIP suffices to understand this extended abstract, let us mention that formal definitions can be found in [GwMiRac85, GrMiWi86a].

† Supported in part by NSERC grant A4107.
‡ Supported in part by an NSERC post-graduate fellowship.

Until recently, ZKIP's were known only for some specific problems in **NP** $\cap$ **CoNP** [GaHaYu85, GwMiRac85]. It was conjectured by Silvio Micali, and believed by most researchers, that ZKIP's could not exist for **NP**-complete problems. Under cryptographic assumptions, this intuition was proven wrong by [GrMiWi86a] as they found a ZKIP for 3-COL. Independently but subsequently, [BrCr86] proposed a ZKIP for satisfiability. Since then, several other similar protocols have been discovered. Obviously (because Karp reductions actually carry **NP** certificates), it suffices to find a ZKIP for any **NP**-complete problem in order to get one for all problems in **NP**. However, the ZKIP's resulting from a sequence of Karp reductions are likely to be horrible and of no practical use.

In fact, 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 would *transfer her confidence* to Bob if she could convince him of the theorem with a level of confidence comparable to her own. This transfer of confidence is *zero-knowledge* if it yields no additionnal information to Bob; in particular, Alice's convincing argument remains secret. The main characteristic of a zero-knowledge transfer of confidence (ZKTC) is that it is *non-transitive*: Alice convinces Bob of the theorem in a way that Bob cannot convince anyone else afterwards.

One nice thing about the early ZKIP's of [GwMiRac85, GaHaYu85] is that their being zero-knowledge does not depend on any unproved cryptographic assumptions (although this is true in a non-constructive sense — either quadratic residuosity is hard, in which case they prove that their protocols are zero-knowledge, or quadratic residuosity is easy, in which case their protocols are *vacuously* zero-knowledge). On the other hand, the ZKIP's of [GrMiWi86a, BrCr86] depend on such assumptions in a crucial way: should they fail, Bob could efficiently get *complete* knowledge of Alice's proof. Moreover, this weakness is *retroactive*: even if Bob does not know an efficient algorithm to counter the cryptographic assumptions at the time the protocol takes place, he could go back to old instances and figure out each of Alice's previous proofs whenever he becomes aware of such an algorithm. He could even do so off-line should he merely know a slow, yet feasible, such algorithm. More, non of these protocols for **NP**-complete problems are *perfect* zero-knowledge.

We give here a general technique to design simple and efficient ZKIP's that provably offer Shannon security for Alice. Our technique extends naturally to the setting of transfer of confidence. This results in protocols that are dual to those of [GrMiWi86a, BrCr86]. A different but similar idea leading to Shannon security has been independently proposed by David Chaum [Ch86]. More precisely, we offer the following:

• *Shannon security for Alice:* our protocol is *perfect* zero-knowledge for problems in **NP**. After its completion, Alice knows *for sure* that nothing about her proof has transpired from the protocol. This does *not* depend on unproved cryptographic assumptions; it would even hold if Bob had infinite computing power (in wich case, of course, he would not need Alice to convince himself of the theorem!). Nothing is ever quite perfect, however: belief in the difficulty of factoring is needed for Bob to be convinced by Alice. The protocols of [GrMiWi86a, BrCr86] are therefore preferable if it is crucial that a proof be a proof (i.e.: Bob should not be fooled into believing a false statement), whereas the protocol given here is preferable if it is crucial that secrecy of Alice's proof be assured at all costs. For a discussion of this issue, please consult section 10.

• *Efficiency and simplicity:* for a variety of problems in **NP**, we get a *direct* ZKIP, without going through an unattractive sequence of reductions. In particular, we get direct and efficient ZKIP's for SAT, 3-COL, Hamiltonian Circuit, Clique, (Exact) Knapsack, etc. This is briefly discussed in section 8.

• *Power:* not only do we obtain ZKIP's for every problem in **NP**, but also ZKTC for any statement for which Alice has a convincing argument. Therefore, it allows Alice to convince Bob of theorems *for which she does not have a proof herself*. To illustrate the idea, 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 [Rab76, SoSt77] to her satisfaction. Although efficient certificates of primality exist for these factors, since PRIMES $\in$ **NP** [Pr75], no feasible algorithm is known for Alice to get them [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 transfer her confidence to Bob without compromising the factorization of m in any way (except for the fact that Bob will know the number of factors).

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 [2] that the Boolean function computed by the circuit is satisfiable, but he will have learned nothing else [3]. 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

Very little background on number theory is necessary to understand our protocols. 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 $z \equiv x^2 \pmod{n}$. Such an x is called a *square root* of z, modulo n. Let y be any fixed quadratic residue. A uniformly distributed random quadratic residue can be generated by choosing $w \in \mathbb{Z}_n^*$ and computing $z = w^2 y \ \mathbf{mod}\ n$. This holds in particular if $y = 1$. The crucial fact is that it is *information theoretically* impossible to distinguish a quadratic residue thus produced using any given $y \in \mathrm{QR}_n$ from one produced using $y = 1$.

Now, let $n = pq$ be the product of two distinct odd primes. The problem of extracting square roots modulo n is computationally equivalent to the problem of factoring n [Rab79]. We shall assume here the *factoring conjecture* to the effect that factoring n is infeasible when p and q are sufficiently large. Therefore, given n and $y \in \mathrm{QR}_n$, we assume it infeasible to compute a square root of y modulo n unless the factorization of n is known. Again, *this*

---

[1] Goldwasser and Kilian's new *provably correct and probably fast primality test* [GwKi86] allows Alice to "efficiently" (currently, the running time is a 12th power polynomial) get short certificates 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 non-transitive transfer of confidence protocol.

[2] Assuming he believes in the difficulty of factoring.

[3] Regardless of *any* assumptions.

cryptographic assumption will not be used to ensure that our protocol is perfect zero-knowledge. It is necessary, however, for Bob to be convinced by the protocol.

3. THE ENCRYPTION OF SECRETS

At the beginning of our protocol, Bob randomly chooses two distinct large primes p and q, and he discloses their product $n = pq$ to Alice. Bob also chooses and discloses to Alice some randomly chosen $y \in \mathrm{QR}_n$. Using the ZKIP of [BrCr86] (independently given in [Be86]), Bob convinces Alice that $y \in \mathrm{QR}_n$. Although he could [BrCr86], there is no point for Bob to additionally convince Alice that n is in the proper form, because he would only hurt himself by choosing n otherwise. The possibility that y might be a quadratic non-residue and that Bob fools Alice into thinking the opposite (which is a very unlikely event) is discussed in section 10. Following the factoring conjecture, we assume throughout that Alice cannot find a square root of y.

Whenever she wishes to encrypt bit b, Alice randomly chooses some $w \in \mathbb{Z}_n^*$ and she computes the encryption $z = w^2 y^b \textbf{ mod } n$. She keeps secret w as her b*-witness* for z. Clearly, any quadratic residue can just as well encrypt a 0 or a 1. Therefore, z itself conveys *no* information on the bit. Using her witness, however, Alice can convince Bob of which bit she encrypted by z: it encrypts a 0 if Alice can exhibit a square root of z and it encrypts a 1 if Alice can exhibit a square root of $zy^{-1} \textbf{ mod } n$. We refer to this operation as Alice *opening* the bit encrypted by z. The only way some z could encrypt *both* 0 and 1 would be if Alice knew a 0-witness w_0 and a 1-witness w_1 for the same z. But she could then compute a square root of y as $w_0 w_1^{-1} \textbf{ mod } n$, without having used the help of Bob anywhere along the protocol, which we assumed to be infeasible for her.

4. COMPUTING ON ENCRYPTED BITS

Let b_1 and b_2 be two secret bits of Alice, let z_1 and z_2 be their encryptions as given to Bob, and let w_1 and w_2 be their witnesses as secretly kept by Alice. It is possible for Alice to convince Bob of whether or not z_1 or z_2 encrypt the same bit, without releasing any additional information. For this, it suffices for Alice to compute and give Bob

$$w = \begin{cases} w_1 w_2 y \textbf{ mod } n & \text{if both } b_1 = 1 \text{ and } b_2 = 1 \\ w_1 w_2 \textbf{ mod } n & \text{otherwise} \end{cases} .$$

Let $z = z_1 z_2 \textbf{ mod } n$. It is easy to see that $z = w^2 \textbf{ mod } n$ if $b_1 = b_2$, whereas $z = w^2 y \textbf{ mod } n$ otherwise. This is something Bob can verify. In other words, z is an encryption for the exclusive-or of b_1 and b_2, and w is Alice's witness for this.

Similarly, 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, and let w_i and $\hat{w}_i$ be their corresponding witnesses. 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: Independently for each i, $1 \le i \le k$, Alice convinces Bob by the above protocol that z_i and $\hat{z}_i$ encrypt the same bit.

Again, let $u = b_1 b_2 \cdots b_k$, let z_i encrypt b_i for each i, $1 \le i \le k$, and let w_i be its witness. Now, let $\hat{u} = \hat{b}_1 \hat{b}_2 \cdots \hat{b}_k$ be some k-bit string *different* from u, let $\hat{z}_i$ be an encryption of $\hat{b}_i$ for each i, $1 \le i \le k$, and let $\hat{w}_i$ be its witness. 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 $b_i \neq \hat{b}_i$). 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 $v_i = z_i \hat{z}_i \ \mathbf{mod}\ n$. 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. Let s be a safety parameter agreed upon between Alice and Bob. Alice randomly chooses s permutations $\sigma_1, \sigma_2, \cdots, \sigma_s$ of $\{1, 2, \cdots, k\}$ and ks random integers $x_{ij} \in \mathbb{Z}_n^*$ for $1 \le i \le k$ and $1 \le j \le s$. She then computes and discloses to Bob $a_{ij} = x_{ij}^2 v_{\sigma_j(i)} \ \mathbf{mod}\ n$ for each i, j. At this point, Bob selects a random subset $X \subseteq \{1, 2, \cdots, s\}$ and sends it to Alice as a challenge. In order to convince Bob, Alice must:

(i) for each $j \in X$, disclose some i_j such that $a_{i_j j}$ encrypts a 1, and open this $a_{i_j j}$ for Bob (by giving him $w = x_{i_j j} w_r \hat{w}_r \ \mathbf{mod}\ n$, where $r = \sigma_j(i_j)$ so that Bob can check that $a_{i_j j} = w^2 y \ \mathbf{mod}\ n$);

(ii) for each $j \not\in X$, disclose the permutation σ_j and use the string equality protocol to convince Bob that $a_{1j} a_{2j} \cdots a_{kj}$ encrypts the same string as $v_{\sigma_j(1)} v_{\sigma_j(2)} \cdots v_{\sigma_j(k)}$. ■

<u>Theorem</u>

(i) This protocol conveys no information on the bit strings u and $\hat{u}$, except for the fact that they are distinct, and

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

Proof (sketch).

(i) Observe that whenever $j \in X$, the protocol tells Bob that the original bit strings are distinct in at least one place, but it gives no clue as to any single i such that $b_i \neq \hat{b}_i$ because the permutation σ_j is then kept secret. On the other hand, whenever $j \not\in X$, Bob gains no information whatsoever on the original strings. More formally, it is easy for Bob to simulate his discussion with Alice, given only the information that the strings are different (more details in the final paper).

(ii) Suppose that $v_1 v_2 \cdots v_k$ encrypts the identically zero string and that Alice cannot figure out a square root for y. The only thing Alice can do to hope convincing Bob that u and $\hat{u}$ are different is to guess *exactly* which subset X will be chosen by Bob and to encrypt identically zero strings with $a_{1j} a_{2j} \cdots a_{kj}$ for each $j \not\in X$ and non-identically zero strings with $a_{1j} a_{2j} \cdots a_{kj}$ for each $j \in X$. The results follows from the fact that there are 2^s equally likely choices of X for Bob. (Of course, Alice can convince Bob of anything she wants if she knows a square root of y.) ■

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 of Alice. For $1 \le i \le t$, let z_i be an encryption of b_i known to Bob, and let w_i be Alice's secret witness. 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. A similar idea was introduced in [CoFi85] and used in [BrCr86, Be86] to obtain a protocol dual to the one described here.

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 once and only 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. Let s be a safety parameter agreed upon between Alice and Bob. Alice randomly chooses s permuted truth tables for B and she discloses encryptions for each of them, keeping the witnesses. At this point, Bob selects a random subset $X \subseteq \{1, 2, \cdots, s\}$ and sends it to Alice as a challenge. In order to convince Bob that z is an encryption of $B(b_1, b_2, \cdots, b_t)$, Alice must:

(i) for each $j \in X$, open the entire encryption of the j^{th} permuted truth table, so that Bob can check that it is a valid truth table for B;

(ii) for each $j \notin X$, point out to the appropriate block in the encryption of the j^{th} permuted 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.

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 $t \leq 2$.

5. 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, 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.

6. ZKIP FOR THE NUMBER OF PRIME FACTORS

Let us now come back to the problem mentioned at the end of the introduction. Alice has selected k distinct primes $p_1, p_2, \cdots, p_k$ and she formed their product $m = p_1 p_2 \cdots p_k$. She wishes to convince Bob that m is 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 interesting 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 **mod** p is a certificate that p is composite (where primes have no certificates and composites have lots [Rab76, SoSt77]). 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 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 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 propeeties, such as being of the form $2q+1$, where q is also a prime.

7. 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 [Gi77]. 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 consider Babai's class **MA** [Ba85], which we would rather call **RNP**, to be the most natural [4]. This class is such that **NP** $\cup$ **BPP** $\subseteq$ **RNP**, hence **NP** is almost certainly a strict subset of **RNP**.

<u>Definition.</u> 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) $(\forall x \in \Sigma^*)[x \in X \iff (\exists a \in \Sigma^*)[\,|a| = p(|x|)$ **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 has an argument a for some $x \in X$, she can convince Bob with a ZKIP that $x \in X$. Let us first restrict ourselves to one-sided probabilistic algorithms.

---

[4] For a discussion as to why we favour **MA** over the seemingly more powerful **AM**, see section 9.

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 [Ad77]. Here, each time the probabilistic algorithm is run on any yes-instance, it accepts with probability at least ½, whereas it always rejects no-instances. It is well known that $\mathbf{RP} \subseteq \mathbf{NP} \cap \mathbf{BPP}$ and that co-$\mathbf{RP} \subseteq \mathbf{BPP}$, 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 Consider a problem $X \in \mathbf{RNP}$ 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 does not have a definite proof that $x \in X$, she convinced herself probabilistically that $<x,a> \in A$, hence $x \in X$. Assuming Bob believes that factoring is hard, it is then possible for Alice to efficiently transfer to Bob her confidence that $x \in X$ by a perfect zero-knowledge interactive protocol. The protocol remains perfect zero-knowledge even if factoring is easy.

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, of course. 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 the 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) 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**. A formal proof will be included in the final paper. ■

The above protocol does *not* work directly for $X \in \mathbf{RNP}$ 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 $A \in \mathbf{BPP}$ but $A \notin \mathbf{RP} \cup$ 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, but even then we do not quite get a *perfect* zero-knowledge protocol (but we come arbitrarily close).

(i) 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.

(ii) Even if Alice is in good faith, the above idea leaves the door open for Bob to cheat. Indeed, it could be that the circuit outcome is not what she expected because Bob could have 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 randomly selects an element of $\mathbb{Z}_n^*$, squares it, and then randomly decides whether to multiply it by Bob's quadratic residue y. She gives the result to Bob. At this point, Bob randomly decides whether to multiply it by y. Alice keeps track of her witness for the resulting encrypted bit.

In order to see why this is not quite perfect zero-knowledge, the formal definition is necessary [GrMiWi86a]. The problem comes from the fact that it will not be possible for Bob to simulate the arbitrarily small probability that the outcome of the majority circuit come out wrong (because he does not know in general what this probability should be). This issue will be discussed in the final paper.

Main Theorem. Consider any $X \in$ **RNP** and some $x \in X$ for which Alice has an argument a. Assume the proof that $X \in$ **RNP** is in the public domain [5]. Even though Alice may not have a definite proof that $x \in X$, she convinced herself probabilistically that $<x,a> \in A$, hence $x \in X$. Assuming Bob believes that factoring is hard, it is possible for Alice to efficiently transfer to Bob her confidence that $x \in X$ with an arbitrarily low probability of failure, by a protocol that discloses no additional information to Bob. Alice's secret argument remains uncompromised even if factoring is easy.

Proof (sketch). By the above discussion.

Let us stress again that this protocol is interesting even when $A \in$ **NP**, hence $X \in$ **NP** (as in section 6 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 $A \in$ **NP**. Nonetheless, she can 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 trap-door, she can convince Bob as well without compromising the trap-door.**

8. OTHER EXAMPLES OF ZKIP's (sketch)

Our general technique can be used directly to get efficient ZKIP's for a variety of **NP**-complete problems. A very useful sub-protocol, of which the boolean computation protocol from section 4 is but an instance, allows Alice to convince Bob that a given string of quadratic residues encrypts a secret permutation of some cleartext data (such as the adjacency matrix of a random permutation of some "cleartext" graph). Subsequent opening of a selected set of these residues easily yields ZKIP's for **NP**-complete problems as diverse as Hamiltonian Circuit, Clique and Exact Knapsack. Similarly, the string inequality protocol yields immediately an efficient perfect zero-knowledge interactive protocol for 3-COL.

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

9. RNP, ARTHUR-MERLIN GAMES AND INTERACTIVE PROTOCOLS [6]

As previously mentioned, our class **RNP** is equivalent to Babai's class **MA** in his Arthur-Merlin games (and similar to Papadimitriou's stochastic satisfiability in his games against nature [Pa85]). According to Babai, his other class **AM** is a better candidate for the generalization of **NP** and **BPP**. This is because he could prove that **MA** $\subseteq$ **AM**. The interest of **AM** is further increased by Goldwasser and Sipser's proof that, for any fixed $k \geq 2$, **AM** = **IP**(k), the class of languages that allow an interactive protocol with no more than k rounds [GwSi86]. All these considerations are theoretically very compelling.

We claim nonetheless that **MA** (i.e.: **RNP**) is a more *natural* class for *practical* purposes, at least in cryptographic settings. Consider, for instance, the example of section 6. It is natural that Alice has a convincing argument that m is the product of exactly k distinct primes. This corresponds to a Merlin-Arthur game, except for the fact that it is not because Alice has Merlin's infinite wisdom that she obtained the argument: it is because she built it from trap-door information. From Babai's theorem **MA** $\subseteq$ **AM**, we know that there is also an Arthur-Merlin protocol for the same problem, but it is likely to be far more complicated and less natural (at least if we follow Babai's proof directly).

If needed, our technique could nonetheless be applied to obtain zero-knowledge protocols for problems in **AM**, and therefore zero-knowledge versions of any interactive protocol with a bounded number of rounds, very much like the way it is done in [GrMiWi86a]. This does not seem to be very interesting in practice, however, because the resulting protocols would often be far too complicated. Moreover, this result is subsumed (except for the fact that our protocol would be *perfect* zero-knowledge, regardless of unproved assumptions) by independent work of Ben-Or and Impagliazzo, in which they show that every language which has an interactive proof system has a zero-knowledge one [BO86, Im86].

10. IS IT PREFERABLE TO TRUST BOB OR ALICE?

"Cheating" takes up a different meaning, depending on whether you are talking about Bob or Alice. For Bob to cheat means that he gains knowledge on (if not of) Alice's proof. Perhaps did he not quite obtain this Hamiltonian circuit he is desperately seeking, for instance, but he learned enough to drastically reduce his search. On the other hand, for Alice to cheat means that she succeeds in convincing Bob of a false "theorem", or at least of one for which she does not happen to have a proof or even a convincing argument.

It is also interesting to distinguish between *lucky* and *daring* successful cheating. The former refers to Alice or Bob figuring out, against all odds, a piece of information that will enable him/her to quietly go about his/her cheating with the certainty of being successful and undetected. The latter refers to Alice or Bob taking an illegal move that is almost certainly going to result in his/her cheating being detected at some point in the future, but that might nonetheless, with an exponentially small probability, allow him/her to succeed.

Finally, cheating is *retroactive* (or *off-line*) if it can take place some times after the protocol is completed, by looking back at its record. Conversely, it is *real-time* if it must be completed within some specific time bound.

---

[6] This section assumes you are familiar with [Ba85, GwMiRac85].

As pointed out in the introduction, previous ZKIP's for **NP**-complete problems [GrMiWi86a, BrCr86] were only "proven" zero-knowledge assuming unproved cryptographic assumptions, and they were known not to be *perfect* zero-knowledge. This means that Alice could never participate in such protocols with a quiet mind: an algorithmic breakthrough might allow Bob to cheat, even retroactively, and even if the new algorithm is not fast enough for a real-time response while the protocol is taking place. Even if the cryptographic assumptions turned out to be well-founded, Bob still has a (very slight) probability of lucky (hence undetectable) cheating. On the other hand, regardless of any assumptions, the only cheating Alice could attempt would be of the daring kind.

We presented here a ZKIP for **NP**-complete problems that does not base Alice's safety on unproved assumptions. The *only* way Bob can hope to learn anything about Alice's secret is to be daring right from the beginning and choose a quadratic non-residue as his y. He would almost certainly get caught by Alice while trying to convince her that $y \in QR_n$ [BrCr86, Be86], but he would otherwise be capable of distinguishing Alice's encryptions of 0 from her encryptions of 1. Asking Bob to disclose a square root of y at the very end of the protocol (which is not detrimental to him at that point, assuming he is honest), provides Alice with *certainty* that Bob has not learnt any of her secrets (and *never* will retroactively), because this completes the proof for Alice that the protocol was perfect zero-knowledge. On the other hand, Bob's belief that Alice cannot cheat our protocol depends on his belief of the factoring conjecture. Clearly, Alice could "open" any quadratic residue as either 0 or 1, whichever suits her best, if she could only obtain a square root of y, but she must be able to do so in real-time. Moreover, even if the factoring conjecture is true, Alice still has a (very slight) possibility of lucky (hence undetectable) cheating. Finally, retroactive cheating is not possible for Alice because it is meaningless.

In a nutshell, compared to the protocols of [GrMiWi86a, BrCr86], ours reverses *exactly* the roles of Alice and Bob cheatingwise, except for retroactivity and real-time considerations: an algorithm capable of factoring in two hours, for instance, would spell doom to the protocol of [BrCr86], but it would be of no direct consequence here. Is it preferable to trust Bob or Alice? We do not know, but it sure is nice to have the choice! Finally, think of the following provocative consideration: assume that Alice claims to have proven Theorem **T** and she uses our protocol to convince a skeptical Bob of this. At the end of the protocol, regardless of any unproved assumptions, Bob will be non-constructively convinced that either Alice has a proof of **T** or she has hot results on integer factoring! In particular, no assumptions are needed if **T** says: ‘‘I have an efficient factoring algorithm’’...

11. OUTLINE OF THE DUAL PROTOCOL OF [BrCr86]

Because proceedings of the CRYPTO conference may not be widely distributed this year (1986), let us briefly describe the dual protocol presented in [BrCr86]. recall that it requires a cryptographic assumption to prevent Bob from figuring out Alice's secret and that it is not *perfect* zero-knowledge. Bob's confidence in Alice's good faith, however, is not based on any unproved assumptions. We assume here that the reader is familiar with Jacobi symbols and the quadratic residuosity assumption (QRA).

The main difference is that the composite number $n=pq$ is chosen by Alice rather than Bob. She also randomly selects some quadratic *non*-residue y whose Jacobi symbol is +1. She gives n and y to Bob. She uses ZKIP's of [GaHaYu85] and [GwMiRac85] to convince Bob that n has only two primes factors and that y is a quadratic non-residue, respectively. In order to encrypt some bit b, she chooses a random $x \in \mathbb{Z}_n^*$ and computes $z = x^2 y^b \ \mathbf{mod}\ n$, which is a quadratic residue if and only if $b=0$. Under QRA, Bob cannot distinguish encryptions of zeroes.

Alice can help Bob compute on encrypted bits in a way very similar to the one described above. The resulting protocol cannot be perfect zero-knowledge, however, because quadratic residues are obviously not information theoretically indistinguishable from quadratic non-residues. Further details are available from the authors.

12. A GRAPH THEORETIC PROTOCOL (sketch)

In this section, we outline an approach that allows trading the number theoretic assumption (factorization is hard) for an assumption related to graph isomorphism. We make no attempts here to formalize the concepts involved, as this idea will be developped in a further paper.

Definition. A graph G is *hard* if it is difficult, with high probability, to figure out an isomorphism between G and a randomly permuted isomorphic copy of G.

Assumption. Arbitrarily hard graphs exit and they can be constructed efficiently.

In order to achieve protocols similar to those described so far in this paper, it suffices to show how this assumption alows Alice to give Bob information theoretically indistinguishable encryptions of zeroes and ones. For this, Alice and Bob first agree on some graph $G = \langle V, E \rangle$. Then, Bob randomly selects a permutation $sigma : V \rightarrow V$ and uses it to produce the graph $H = \langle V, F \rangle$ isomorphic to G defined by $u, v \in F$ if and only if $\sigma(u), \sigma(v) \in E$. Bob gives H to Alice, keeping σ secret. By assumption, with high probability, Alice cannot compute σ from G and H (nor any other isomorphism between G and H). Using a ZKIP of [GrMiWi86a], Bob convinces Alice that G and H are isomorphic (notice that this ZKIP is perfect and that it does not depend on any assumptions).

After this initialisation, Alice encrypts zeroes as randomly permuted isomorphic copies of G and ones as randomly permuted isomorphic copies of H. She keeps the isomorphisms as witness. She can prove that two graphs encrypt the same bit by showing Bob an isomorphism between them. We invite the reader to figure out how Alice can prove that two graphs encrypt complementary bits, how to achieve coin-tossing in a well, and how to adapt the general protocol for any problem in **RNP**.

An advantage of this protocol over the one based on number theory is that some standard "certified" hard graph could be constructed once and for all for use by all parties. This is possible because there is no trap-door involved.

13. OPEN PROBLEM

Could there be a ZKIP that allows neither Alice nor Bob to cheat, without making use of any unproved cryptographic assumptions? Could it be perfect zero-knowledge? Notice that putting back to back the current protocol with the one of [BrCr86] is a bad idea: instead of adding their strengths, this only results in the protocols adding their weaknesses.

ACKNOWLEDGEMENT

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

REFERENCES

[Ad77] Adleman, L., ‘‘Reducibility, randomness and intractability’’, *Proceedings of the 9th Annual ACM Symposium on the Theory of Computing*, 1977, pp. 151-163.

[Ba85] Babai, L., ‘‘Trading group theory for randomness’’, *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing*, 1985, pp. 421-429.

[Be86] Benaloh (Cohen), J.D., ‘‘Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols’’, *CRYPTO 86*, 1986.

[BO86] Ben-Or, M., *Private communication* through [GrMiWi86a], 1986.

[BrCr86] Brassard, G. and C. Crépeau, ‘‘Zero-knowledge simulation of Boolean circuits’’, Presented at *CRYPTO 86*, 1986.

[Ch86] Chaum, D., ‘‘Demonstrating that a public predicate can be satisfied without revealing any information about how’’, Presented at *CRYPTO 86*, 1986.

[CoFi85] Cohen, J. D. and M. J. Fischer, ‘‘A robust and verifiable cryptographically secure election scheme’’, *Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science*, 1985, pp. 372-382.

[GaHaYu85] 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.

[Gi77] Gill, J. ‘‘Computational complexity of probabilistic Turing machines’’, *SIAM Journal on Computing*, Vol. 6, no. 4, December 1977, pp. 675-695.

[GrMiWi86a] Goldreich, O., S. Micali and A. Wigderson, ‘‘Proofs that yield nothing but the validity of the assertion and the methodology of cryptographic protocol design’’, presented at a Workshop on Probabilistic Algorithms, Marseille, March 1986 (organized by Schnorr); Submitted to *27th Annual IEEE Symposium on the Foundations of Computer Science*, 1986.

[GrMiWi86b] Goldreich, O., S. Micali and A. Wigderson, ‘‘Methodological theorems for cryptographic protocol design’’, *in preparation*, 1986.

[GwKi86] Goldwasser, S. and J. Kilian, ‘‘A provably correct and probably fast primality test’’, *Proceedings of the 18th Annual ACM Symposium on the Theory of Computing*, 1986, pp. 316-329.

[GwMiRac85] 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.

[GwSi86] Goldwasser, S. and M. Sipser, ‘‘Arthur-Merlin games versus interactive proof systems’’, *Proceedings of the 18th Annual ACM Symposium on the Theory of Computing*, 1986.

[Im86] Impagliazzo, R., *Private communication* through Steven Rudich, 1986.

[Pa85] Papadimitriou, C. H., ‘‘Games against nature’’, *Journal of Computer and System Sciences*, Vol. 31, 1985, pp. 288-301.

[Pr75] Pratt, V., ‘‘Every prime has a succinct certificate’’, *SIAM Journal on Computing*, Vol. 4, 1975, pp. 214-220.

[Rab76] Rabin, M. O., ‘‘Probabilistic algorithms’’, in *Algorithms and Their Complexity: Recent Results and New Directions*, J. F. Traub (editor), Academic Press, New York, NY, 1976, pp. 21-39.

[Rab79] Rabin, M. O., ‘‘Digitalized signatures and public-key functions as intractable as factorization’’, *MIT/LCS/TR-22*, 1979.

[SoSt77] Solovay, R. and V. Strassen, ‘‘A fast Monte Carlo test for primality’’, *SIAM Journal on Computing*, Vol. 6, 1977, pp. 84-85.