J. Cryptology (1988) 1: 77–94

Journal of Cryptology

Zero-Knowledge Proofs of Identity

Uriel Feige, Amos Fiat,[1] and Adi Shamir
Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel

Abstract. In this paper we extend the notion of interactive proofs of assertions to interactive proofs of knowledge. This leads to the definition of unrestricted input zero-knowledge proofs of knowledge in which the prover demonstrates possession of knowledge without revealing any computational information whatsoever (not even the one bit revealed in zero-knowledge proofs of assertions). We show the relevance of these notions to identification schemes, in which parties prove their identity by demonstrating their knowledge rather than by proving the validity of assertions. We describe a novel scheme which is provably secure if factoring is difficult and whose practical implementations are about two orders of magnitude faster than RSA-based identification schemes. The advantages of thinking in terms of proofs of knowledge rather than proofs of assertions are demonstrated in two efficient variants of the scheme: unrestricted input zero-knowledge proofs of knowledge are used in the construction of a scheme which needs no directory; a version of the scheme based on parallel interactive proofs (which are not known to be zero knowledge) is proved secure by observing that the identification protocols are proofs of knowledge.

Key words. Proofs of knowledge, Zero knowledge, Digital identification.

1. Introduction

Zero-knowledge proofs [8] are an elegant technique to limit the amount of information transferred from a prover A to a verifier B in a cryptographic protocol. As defined in the original GMR paper [8], the proofs refer to language membership problems (is input I a member of language L?), and their applicability to any language L in NP was recently demonstrated by Goldreich *et al.* [7]. Additional properties of zero-knowledge proofs were investigated by Goldwasser and Sipser [10], Brassard and Crepeau [1], Chaum [2], and many others.

The name "zero-knowledge proofs" is slightly misleading, since the prover A reveals one bit of knowledge to the verifier B (namely that I belongs to L). Our first objective in this paper is to show that it is possible to extend this notion to "truly zero-knowledge proofs" which do not even reveal this single bit. The basic idea is to replace "knowledge" by "knowledge about knowledge": A's goal is not to prove

[1] Current address: EECS, University of California at Berkeley, Berkeley, CA 94720, U.S.A.

that I *belongs* to L, but to prove that he *knows* the status of I with respect to L. From B's point of view, he did not get any information whatsoever about "the real world" (I, L, and their relationships)—only about A's state of knowledge concerning the real world.

As a motivating (but technically inaccurate) example, consider a prover A who wants to prove to a skeptical B that he has settled Fermat's last theorem. With the type of proof introduced in this paper, A can convince B that he is a mathematical superstar without telling B anything new about the problem—not even whether he has found a proof or a counterexample!

A related idea was presented by Galil *et al.* [6], who defined the notion of "result indistinguishable protocols." However, the two models differ in their goals: in their model the prover A proves either that I belongs to L or that I does not belong to L. B knows which claim is being proven and gets a convincing proof, while the passive eavesdropper C (who is not allowed to participate or meddle in the protocol) cannot determine from the communication tape which claim is being proven and whether the tape constitutes a convincing proof. The main difficulty in extending such a result to our model is that we want the same party B to be ignorant of what is being proven and yet to be convinced by the proof.

The intuitive observation that the zero-knowledge paradigm could be used to prove knowledge rather than existence of witnesses is not new, and appeared (without proper formalization) in several parenthetical remarks by Goldwasser, *et al.* [8], Chor *et al.* [3], Galil *et al.* [6], Chaum [2], and in several other papers. Recently, Tompa and Woll [16] have independently addressed the same issue and developed formal definitions which are conceptually similar to but technically different from our definitions. As demonstrated in this paper, the formal definitions of "proofs of membership" and "proofs of knowledge" are quite different, and there is a fundamental philosophical difference between them. The notion of "knowledge" is very fuzzy, and *a priori* it is not clear what proofs of knowledge actually prove. Several researchers have investigated this notion from a different point of view (see, e.g., [11] and [15]), and we believe that a combined approach to knowledge and proofs of knowledge can have an important impact on areas outside cryptography (in particular logic and distributed computing).

2. Interactive Proofs of Knowledge

Our model differs from the original GMR model in several important aspects. We do not allow the prover to be infinitely powerful (such a prover knows everything about its inputs by definition), and restrict both the prover and the verifier to be polynomial-time probabilistic Turing machines. The machines, A and B, have a common input tape I, two communication tapes CA and CB, private work tapes WA and WB, and private random tapes RA and RB. In addition, each machine is given access to a private knowledge tape (KA and KB).

Definition. A polynomial-time predicate $P(I, S)$ is a predicate in which $|S|$ is polynomially related to $|I|$, and the truth value of the predicate can be checked in polynomial time.

In proofs of knowledge A tries to convince B that he has "knowledge" tying the common input I with a publicly known polynomial-time predicate $P(\cdot, \cdot)$. More specifically, if the contents of A's knowledge tape KA happen to be an S such that $P(I, S)$ is satisfied, an interactive proof of knowledge allows A to convince B of this fact. This model restricts A's proofs of knowledge to problems in NP, and typical examples of predicates are "S is a valid 3-colouring of graph I" and "S satisfies the CNF formula I." The role of the verifier's knowledge tape KB will be explained when we address zero-knowledge issues.

In the rest of this paper we adopt the following conventions:

1. $\bar{A}$ (straight A) represents the real prover who follows its designated protocol and whose knowledge tape contains S whenever it exists.
2. $\tilde{A}$ (crooked A) represents a polynomial-time cheater who can deviate from the protocol in an arbitrary way, and whose knowledge tape can contain arbitrary strings.
3. A represents either $\bar{A}$ or $\tilde{A}$.
4. $\bar{B}$ (straight B) represents the real verifier who follows its designated protocol.
5. $\tilde{B}$ (crooked B) represents an arbitrary polynomial-time program (which may try to extract additional information from $\bar{A}$).
6. B represents either $\bar{B}$ or $\tilde{B}$.
7. (A, B) represents the execution of the two party protocol in which A is the prover and B is the verifier.

To demonstrate the subtlety of interactive proofs of knowledge, consider the following examples in which I is the product of two primes and A claims that it knows its factorization S. We do not insist that the proofs should be zero knowledge, and we use the fact that factoring and square-root extraction are computationally equivalent. The question we address in each case is whether A's proof of knowledge should convince $\bar{B}$:

Example 1. A extracts the square roots mod I of several substrings $R_1 \ldots R_k$ from RA, and sends RA and the roots of about one-quarter of the R_i to $\bar{B}$. $\bar{B}$ should not be convinced since it cannot verify that A really uses his random tape.

Example 2. $\bar{B}$ sends several substrings $R_1 \ldots R_k$ from RB to A, and receives their square roots mod I in one-quarter of the cases. Even though $\bar{B}$ itself could not use the results to compute S, it is clear that a slightly modified $\bar{B}'$ could compute S with high probability by sending A the squares mod I of these substrings. As A cannot distinguish between $\bar{B}$ and $\bar{B}'$, $\bar{B}$ should be convinced.

Example 3. Let f be a strong one-way function which is known to both parties. $\bar{B}$ sends several substrings $R_1 \ldots R_k$ from RB to A and receives the square roots of one-quarter of the values $f(R_1) \ldots f(R_k)$. The problem is not completely defined, but $\bar{B}$ should be wary of A's proof: even though arbitrary square roots cannot be extracted without knowledge of the factorization of I, the randomizing function f prevents even a modified $\bar{B}'$ from getting the second square root of the same number it needs to factor I.

Example 4. To show that $\bar{B}$'s skepticism in Example 3 can sometimes be justified, consider the function $f(x) = x^2 \pmod I$. This is probably a strong one-way function but it interacts badly with the problem at hand, and enables a cheating $\tilde{A}$ to extract square roots even when it does not know the factorization of I.

Example 5. Consider now the function $f(x) = ax^2 + b \pmod I$ with nonzero a and b. The multiplication by a and addition of b seems to destroy the ability of $\bar{B}'$ to factor I, which suggests that $\bar{B}$ should not be convinced. However, Pollard and Schnorr [13] prove that binary quadratic equations such as $ax^2 + b = y^2 \pmod I$ can be solved in polynomial time, and thus $\bar{B}'$ can prepare an x for which he already knows one square root y of $f(x)$, but A cannot know which. As a result, $\bar{B}$ should be convinced in this case that A knows the factorization of I.

Formalizing the concept of "interactive proofs of knowledge" is not easy. The GMR [8] definition of "interactive proofs of membership" is based on the language-recognition paradigm: some intrinsic property of the inputs I is used to define the subset L of "good" inputs, and then we can check the adequacy of a particular protocol $(\bar{A}, \bar{B})$ by verifying that for all I and $\tilde{A}$:

1. If I belongs to L, $\bar{B}$ accepts $\bar{A}$'s proof with overwhelming probability.
2. If I does not belong to L, $\bar{B}$ accepts $\tilde{A}$'s proof with negligible probability.

In other words, $\bar{B}$ accepts exactly L in the sense that the real $\bar{A}$ can convince $\bar{B}$ to accept at least L, but even a faulty $\tilde{A}$ cannot convince $\bar{B}$ to accept more than L.

The existence of some S which satisfies $P(I, S)$ is an intrinsic property of I which can be used to distinguish between "good" and "bad" inputs. However, there are many cases in which proofs of existence of S are meaningless, and what makes the proofs surprising in these cases is only their constructive nature which demonstrates knowledge. Consider, for example, the following predicates $P(I, S)$:

1. S is the complete factorization of I.
2. S is the discrete logarithm of I modulo a prime Q.
3. S is either a short witness for square freeness or a number larger than 1 whose square divides I.

In all these examples the mere existence of S can be proved by $\bar{B}$ in advance (in Examples 1 and 3 the predicate is always satisfiable, and in Example 2 the predicate can be satisfied when I is not a multiple of Q). When S always exists, the original GMR [8] definition degenerates to the uninteresting "$\bar{B}$ should always accept." What we really want is to distinguish between inputs I for which an S is "known" and inputs I for which no S is "known." However, the subset of I for which S is "known" is ill-defined: the same I should sometimes be accepted and sometimes be rejected by $\bar{B}$. Since the proper decision depends on A's state of knowledge rather than on I's intrinsic properties, we can try to reformulate the two conditions in terms of A instead of I:

1. If A is $\bar{A}$, $\bar{B}$ should accept its proofs with overwhelming probability for all I for which $P(I, S)$ is satisfiable.
2. If A is $\tilde{A}$, $\bar{B}$ should accepts its proofs with negligible probability for all I.

This definition is motivated by the assumption that $\bar{A}$ always has S on his knowledge tape, whereas $\tilde{A}$ does not. However, even if P is a difficult predicate, there can be infinitely many "easy" instances of I for which $\tilde{A}$ can compute the S by itself and then mimic $\bar{A}$'s proof to $\bar{B}$. Consequently, we have to allow $\bar{B}$ to accept $\tilde{A}$'s proofs occasionally, but only when it really happens to know S (by computing it, guessing it, or by obtaining help from its knowledge tape). In other words, we require that for all I and A (which can be either $\bar{A}$ or $\tilde{A}$):

1. If A knows S, $\bar{B}$ should accept A's proof for I with overwhelming probability.
2. If A does not know S, $\bar{B}$ should accept A's proof for I with negligible probability.

This makes it necessary to define the set of things that a particular program knows. An informal definition of this concept was given in a parenthetical remark in [8]:

> $\tilde{A}$ knows S if there is some polynomial-time Turing machine M with complete control over $\tilde{A}$ which prints S as a result of its interaction with $\tilde{A}$.

The notion of knowledge captured by this definition is very broad, and $\tilde{A}$ may not even be "aware" of its knowledge of S. For example, S may be stored in an inaccessible portion of $\tilde{A}$'s code, or may appear temporarily in its work tape, or may be computed only as a complicated function of values derived from polynomially many executions of $\tilde{A}$ during which M repeatedly changes the contents of $\tilde{A}$'s random and work tapes. Unfortunately, this notion of knowledge is incompatible with the definition of interactive proofs of knowledge given above: in the first part of the definition A may "know" S in some bizarre way which cannot be detected by the particular $\bar{B}$, and on the other hand the second part of the definition becomes the tautology "if A cannot convince anyone that it knows S, it should not convince $\bar{B}$ that it knows S."

Having seen the many pitfalls along the path, we finally propose our formal definition of interactive proofs of knowledge. It combines most of the ideas discussed so far, but blends them in a different way:

Definition. A pair of interacting polynomial-time probabilistic Turing machines $\bar{A}$, $\bar{B}$ is called an interactive proof system of knowledge for the polynomial-time predicate $P(I, S)$ if:

1. Completeness: for all I for which $P(I, S)$ is satisfiable, the execution of $(\bar{A}, \bar{B})$ on input I succeeds with overwhelming probability. More formally,

   $\forall a, \exists c, \forall |I| > c$

   if $\bar{A}$ is given on its knowledge tape an S such that $P(I, S)$ and

   $\bar{B}$ is given the empty string on its knowledge tape $\Rightarrow$

   $\text{Prob}((\bar{A}, \bar{B}) \text{ accepts } I) > 1 - 1/|I|^a$.

As we are dealing with the real prover $\bar{A}$, its knowledge tape KA contains an appropriate S. The verifier's knowledge tape KB is taken to be empty.

2. Soundness: there exists a polynomial-time probabilistic Turing machine M (with complete control over A—see Remark 4 below) such that for all A, any initial contents of A's knowledge tape KA and random tape RA, and any sufficiently large I, if the execution of $(A, \bar{B})$ on input I succeeds with nonnegligible probability, then the output produced by M at the end of the execution of $M(A, RA, KA)$ on input I satisfies the predicate P with overwhelming probability. More formally,

$$\forall a, \exists M, \forall b, \forall A, \exists c, \forall |I| > c, \qquad \forall RA, \forall KA$$

$$\text{Prob}((A, \bar{B}) \text{ accepts } I) > 1/|I|^a \quad \Rightarrow$$

$$\text{Prob}(\text{output of } M(A, RA, KA) \text{ on } I \text{ satisfies } P) > 1 - 1/|I|^b.$$

Remarks. 1. The universal quantification over KA guarantees that whatever partial knowledge $\tilde{A}$ has, if this knowledge suffices in convincing $\bar{B}$ to accept, the same knowledge suffices for computing S.

2. We want $\bar{B}$'s conviction that A knows S to depend only on the randomness of its own coin flips, and thus we universally quantify over the RA and define the probabilities only over the random choices of RB.

3. The parameter a specifies the meaning of "nonnegligible" while the parameter b specifies the meaning of "overwhelming." The choice of M may depend on the former but not on the latter, since we want the asymptotic failure rate of the chosen M to be smaller than the inverse of any polynomial.

4. The uniformity of the "interrogator" M which extracts S from any $\tilde{A}$ that manages to convince $\bar{B}$ makes it possible to distinguish between "obvious" and "accidental" knowledge. Since the same M should work for all $\tilde{A}$, RA, and KA, it makes little sense for M to try to analyze $\tilde{A}$'s program or to meddle with its tapes. In fact, for our purposes it suffices to give M the power to reset and rerun $\tilde{A}$ polynomially many times without inspecting or modifying its tapes. This is similar to the "blackbox" simulation, as defined by Oren [12], in the context of variations on the GMR [8] definition of zero knowledge.

The GMR [8] definition of zero knowledge can be stated in the following way:

> An interactive proof system of membership in L is zero knowledge, if, for all inputs *restricted to L*, for all B its view of the communication in $(\bar{A}, B)$ can be recreated, by a polynomial-time (or alternatively, *expected* polynomial-time) probabilistic Turing machine M, with an indistinguishable probability distribution.

Informally, B's view of the communication is the contents of its communication and random tapes, and two sources are indistinguishable if no polynomial-time machine can tell from which of them a certain string input to it originated (for formal definitions see [8]). The above definition assumes uniformity of the algorithms executed by the verifier and by the distinguisher. We note that this definition is not suitable for cryptographic purposes. (The same observation is made independently by Tompa and Woll [16], Oren [12], and in later versions of the GMR paper.) It does not take into account previous information the verifier gathered, either from previous interactions with the prover or by other means, in his attempts to extract knowledge from the prover. We believe that in order to state that a machine cannot learn anything new, we need a formal tool of stating what a machine already knows. This is given by the knowledge tape. Recall that the contents S of KA represents the knowledge a truthful prover has when executing the protocol. Similarly, the contents of KB represents the knowledge the verifier has when executing the protocol. The simulating machine M does not have any knowledge tape of its own, but it can use the verifier's knowledge tape indirectly in the simulating process. Finally, the distinguisher may have access to KB, as we do not want even $\tilde{B}$ to distinguish between the communication in $(\bar{A}, \tilde{B})$ and the simulated communication. We reformulate the GMR definition of zero knowledge in the following way:

Definition. An interactive proof system of membership in L is zero knowledge, if, for all inputs *restricted to* L, for all B and KB its view of the communication in $(\bar{A}, B)$ can be recreated, by a polynomial-time probabilistic Turing machine M, with an indistinguishable probability distribution.

All protocols previously proved to be zero knowledge remain so even with respect to this stronger definition (see [12]). But this does not mean that the change to the original GMR definition is merely syntactical. Feige [4] demonstrated that there exist protocols which are zero knowledge with respect to the GMR definition, but are not zero knowledge with respect to our definition.

As demonstrated in this paper, proofs of knowledge make perfect sense even when the set of I for which S exists is polynomially recognizable (and in particular when S always exists). We exploit this extra degree of freedom by defining *unrestricted input* zero-knowledge proofs of knowledge:

Definition. An interactive proof system of knowledge is *unrestricted input* zero knowledge, if, for all inputs, for all B and KB its view of the communication in $(\bar{A}, B)$ can be recreated, by a polynomial-time probabilistic Turing machine M, with an indistinguishable probability distribution.

Remarks. 1. If the common input I is such that there is no S satisfying $P(I, S)$, then KA is empty, and $\bar{A}$ refuses to participate in the protocol. Thus the first task of M on input I is to decide whether there is any communication to recreate. This implies that a predicate $P(I, S)$ can have an unrestricted input zero-knowledge interactive proof system of knowledge only when the set of I for which S exists is recognizable in random polynomial time. This triviality of the input language should not be confused with triviality of assertions being proved. The knowledge the prover must possess may certainly be nontrivial.

2. To prove that a particular proof system is zero knowledge in our model we use the GMR [8] idea of resetting the simulation whenever it gets stuck. It is

interesting to notice that in proofs of knowledge the resettable simulation is used in two very different ways: in proving the soundness property we use it to extract from $\tilde{A}$ as much information as possible about S, while in proving the zero-knowledge property we use it to avoid situations in which $\tilde{B}$'s questions cannot be answered. In proofs of membership, soundness is based on $\tilde{A}$'s inability to satisfy $P(I, S)$, and the resettable simulation is used only to prove the zero-knowledge character of the protocol.

Theorem 1. *Under the assumption that secure public key encryption schemes exist, any polynomial-time predicate has an interactive proof system of knowledge which is restricted input zero knowledge.*

Proof (sketch). Under the assumption that NP reductions are one-to-one and efficiently invertible, it suffices to consider Blum's zero-knowledge interactive proof system of membership for graph Hamiltonicity. For the sake of completeness we sketch the protocol. The common input I is the adjacency matrix of a graph G.

1. $\bar{A}$ chooses randomly $|I|$ public key encryption schemes.
2. $\bar{A}$ permutes the nodes of the graph randomly.
3. $\bar{A}$ encrypts each entry in the permuted adjacency matrix using a different encryption scheme. He sends the encrypted matrix together with the public keys to B, keeping the secret trapdoor information (the private keys of the encryption schemes) to himself.
4. $\bar{B}$ chooses randomly one bit and sends it to A.
5. If B returns 0, $\bar{A}$ reveals all trapdoor information and the random permutation chosen. $\bar{B}$ can now decrypt the whole matrix and check that its nodes constitute a permutation of the nodes of I.
6. If B returns 1, $\bar{A}$ reveals trapdoor information only sufficient to decrypt a Hamiltonian cycle, leaving the other edges encrypted. From the matrix structure, it is easy to verify that a set of edges constitutes a Hamiltonian cycle.
7. If B returns anything else, $\bar{A}$ stops.

Steps 1–7 are iterated $|I|$ times. If all iterations succeed $\bar{B}$ accepts.

The executions of $(\tilde{A}, \bar{B})$ on input I and random tape RA can be described as an incomplete binary tree which describes $\tilde{A}$'s responses to $\bar{B}$'s requests. Each choice of RB corresponds to a particular path in the tree, where left sons correspond to $\bar{B}$ returning 0 (step 5) and right sons correspond to $\bar{B}$ returning 1 (step 6). Any son that corresponds to a problem $\tilde{A}$ does not answer properly is eliminated along with all its descendants, and thus the successful executions correspond to root-to-leaf paths in the full binary tree that survive the truncation.

The machine M we construct explores this tree by repeatedly resetting $\tilde{A}$ to the root, providing the necessary steering requests, and verifying which one of the two sons of each explored vertex corresponds to a correct answer. It is easy to see that if the truncated tree contains any vertices with degree 2, M can find at least one such vertex in $O(|I|^2)$ time. Since a nonnegligible fraction of the exponentially many leaves of the full binary tree survive by assumption, the polynomial-time M is guaranteed to succeed. M's success implies that it witnesses both a full decryption and a Hamiltonian cycle of the same permuted adjacency matrix. From this M can easily reconstruct S—a Hamiltonian cycle in I.

To complete the proof, we observe that the above protocol can be proved to be restricted input zero knowledge, using techniques as in [7]. □

Interesting cases of unrestricted input zero-knowledge proofs arise if for any I the predicate $P(I, \cdot)$ can be satisfied, but it is nontrivial to compute explicitly from I an S for which $P(I, S)$ is true.

Theorem 2. *Under the assumption that secure public key encryption schemes exist, any language in* NP $\cap$ co-NP *can form the basis of a nontrivial interactive proof system of knowledge which is unrestricted input zero knowledge.*

Proof. Since the problem is in NP, it can be represented by the boolean satisfiability formula $P_1(I, S)$. Since the problem is also in co-NP, its complement can be represented by the boolean satisfiability formula $P_2(I, S)$. Consider now the boolean predicate $P(I, S) = P_1(I, S) \vee P_2(I, S)$, which is satisfiable for all I. By Theorem 1 it has a valid interactive proof system of knowledge. Since the assumption that S exists is superfluous, the two definitions of zero knowledge coincide, and thus A can prove that he knows whether I is in the language or in its complement without revealing even this single bit of knowledge. □

3. An Efficient Identification Scheme

An identification scheme is a protocol which enables party A to prove his identity polynomially many times to party B without enabling B to misrepresent himself as A to someone else. Identification schemes are closely related to the notion of digital signatures, but there are no messages judges and disputes: the proof of identity is either accepted or rejected in real time, and as a result the requested access or service is granted or withheld. This is one of the fundamental problems in cryptography, and it has numerous practical applications. The basic problem with most of the current identification techniques (ID cards, credit cards, computer passwords, PIN numbers, etc.) is that A proves his identity by revealing a constant S in the form of a printed card or a memorized value. A sophisticated adversary who cooperates with a dishonest verifier B can use a xerox copy of the card or a recording of the secret value to misrepresent himself successfully as A at a later stage.

Our goal in the next two sections is to develop a truly practical scheme, which can be implemented in software in a fraction of a second even on the weak microprocessors embedded in smart cards. For reasonable choices of the parameters, our scheme is very fast and requires only few modular multiplications. If factoring is difficult, the new scheme is provably secure in the strong sense of the Goldwasser *et al.* [9] "paradoxical scheme."

Our scheme uses zero-knowledge proofs of knowledge. This way A can convince B that A knows the constant S (A's key), without revealing S itself, nor any partial information about it. The scheme assumes the existence of a trusted center whose sole purpose is to publish a modulus n which is the product of two large primes of the form $4r + 3$. Such moduli (which are known as Blum integers) are used in a variety of cryptographic applications, and one of their most useful properties is that -1 is a quadratic nonresidue whose Jacobi symbol is $+1 \pmod{n}$. After publishing n, the center can be closed since it has no further role in the protocol. Note that unlike the RSA scheme [14], everyone can use the same universal n, and no one should know its factorization.

The identification scheme is a special case of Theorem 2, in which $\bar{A}$ proves to $\bar{B}$ that he knows whether a certain number is a quadratic residue or a quadratic nonresidue mod n without revealing even this single bit of information. It is a slightly modified version of the identification scheme described by Fiat and Shamir [5] (which leaked nothing but this bit). It incorporates several ideas from an unpublished zero-knowledge proof of quadratic residuosity due to Goldwasser, Micali, and Rackoff, and from the Galil *et al.* [6] "result indistinguishable" residuosity protocol, but it has much lower time and communication complexities. This efficiency is derived primarily from the use of the multiplicative properties of modular square roots to prove the simultaneous knowledge of the quadratic residuosity character of several numbers—something we do not know how to do with problems which are not number theoretic.

$\bar{A}$'s key generation protocol in the new scheme is:

1. Choose k random numbers $S_1, \ldots, S_k$ in Z_n.
2. Choose each I_j (randomly and independently) as $\pm 1/S_j^2 \pmod{n}$.
3. Publish $I = I_1, \ldots, I_k$ and keep $S = S_1, \ldots, S_k$ secret.

The S_j (which are witnesses to the quadratic residuosity character of the I_j) are effectively hidden by the difficulty of extracting square roots mod n, and thus A can establish his identity by proving that he knows these S_j. By allowing I_j to be either plus or minus a square modulo a Blum integer, we make sure that I_j can range over all the numbers with Jacobi symbol $+1$ mod n and thus the S_j exist (from B's point of view) regardless of I_j's character, as required in unrestricted input zero-knowledge proofs of knowledge.

To generate and verify a proof of identity, the parties execute the following protocol:

Repeat steps 1–4 t times:

1. $\bar{A}$ picks a random R, and sends $X = \pm R^2 \pmod{n}$.
2. $\bar{B}$ sends a random boolean vector $(E_1, \ldots, E_k)$.
3. $\bar{A}$ sends the value $Y = R \cdot \prod_{E_j=1} S_j \pmod{n}$.
4. $\bar{B}$ verifies that $X = \pm Y^2 \cdot \prod_{E_j=1} I_j \pmod{n}$.

$\bar{B}$ accepts the proof if and only if step 4 was carried out successfully in all t iterations.

Let $P(I, S)$ be a polynomial-time predicate, where I and S are tuples as defined above. The predicate's value is true if and only if for all $1 \leq j \leq k$ the following condition holds,

$$I_j S_j^2 = \pm 1 \pmod{n}.$$

We formalize the information the public center publishes as a family of predicates $C(I)$ which any input I must satisfy, and which is testable in polynomial time. In the context of our identification scheme, $C(I)$ is true if the modulo n implied by I is the same n which the center publishes. This modulo is known to be a Blum integer, as the center is trusted to follow the specifications of the identification scheme.

Definition. An interactive proof system of knowledge is unrestricted input zero knowledge (relative to a trusted center), if, for all inputs satisfying $C(I)$, for all B and KB its view of the communication in $(\bar{A}, B)$ can be recreated, by a polynomial-time probabilistic Turing machine M, with an indistinguishable probability distribution.

Theorem 3. *The above protocol is a proof of knowledge of $P(I, S)$, and is unrestricted input zero knowledge relative to a trusted center, for $k = O(\log\log n)$ and $t = \Theta(\log n)$.*

Proof (sketch). To prove that $\bar{A}$'s proof always convinces $\bar{B}$, we evaluate the verification condition:

$$Y^2 \cdot \prod_{E_j=1} I_j = \left(R \cdot \prod_{E_j=1} S_j\right)^2 \cdot \prod_{E_j=1} I_j$$
$$= R^2 \prod_{E_j=1} (S_j^2 I_j) = \pm R^2 = \pm X \pmod{n}.$$

Next we show that whenever $\bar{B}$ accepts $\tilde{A}$'s proof with nonnegligible probability, M can print out all the S_j with overwhelming probability. Let T be the truncated execution tree of $(\tilde{A}, \bar{B})$ for input I and random tape RA. Unlike the tree in Theorem 1, $\bar{B}$ may ask $2^k = (\log n)^{O(1)}$ possible questions at each stage, and thus the vertices in T may have polynomially many sons in terms of $|I|$. A vertex is called "heavy" if its degree is larger than $2^k/2$ (i.e., if more than half the executions of $(\tilde{A}, \bar{B})$ at this state are successful). Our goal in this part of the proof is to show that all the S_j can be computed from the sons of a heavy vertex and that a polynomial-time M can find a heavy vertex in T with overwhelming probability.

Let V be any heavy vertex in T and let Q be the set of queries in the form of boolean vectors $(E_1, \ldots, E_k)$ which are properly answered by $\tilde{A}$. It is easy to show that for any $1 \leq j \leq k$ a set Q of more than $2^k/2$ boolean vectors of length k must contain two vectors $(E'_1, \ldots, E'_k)$ and $(E''_1, \ldots, E''_k)$ in which $E'_j = 0$, $E''_j = 1$, and $E'_i = E''_i$ for all $i \neq j$. Since both queries were properly answered, the two verification conditions imply

$$X' = \pm Y'^2 \cdot \prod_{E'_l=1} I_l \pmod{n}$$

and

$$X'' = \pm Y''^2 \cdot \prod_{E''_l=1} I_l \pmod{n}.$$

However, $\tilde{A}$ must choose X before he obtains $\bar{B}$'s query, and thus $X' = X''$. By manipulating the equations we get

$$(Y''/Y')^2 = \pm 1/I_j \pmod{n},$$

and thus Y''/Y' is the desired S_j (recall that n is a Blum integer, and thus exactly one of $+1/I_j$ and $-1/I_j$ has a square root).

Next we show that at least half the vertices in at least one of the levels in T must be heavy. Let α_i be the ratio between the number of vertices at level $i+1$ and the number of vertices at level i in T. If $\alpha_i \leq (3/4)2^k$ for all $1 \leq i \leq t$, then the total number of leaves in T (which is the product of all these α_i) is bounded by $(3/4)^t 2^{kt}$, which is a negligible fraction of the 2^{kt} possible leaves. Since we assume that this fraction is polynomial, $\alpha_i > (3/4)2^k$ for at least one i, and thus at least half the vertices at this level must contain more than $2^k/2$ sons.

To find a heavy vertex in T, M chooses polynomially many random vertices at each level, and determines their degrees by repeated resets and executions of $\tilde{A}$. To ensure a uniform probability distribution in spite of the uneven degrees of the vertices, M should explore random paths in the untruncated tree, and restart from the root whenever the path encounters an improperly answered query. Since a nonnegligible fraction of the leaves is assumed to survive the truncation, this blind exploration of T can be carried out in polynomial time.

The last part of the proof deals with the zero-knowledge aspect of the protocol. $\tilde{A}$ can easily cheat $\bar{B}$ with probability 2^{-k} per iteration by guessing the $(E_1, \ldots, E_k)$ vector, preparing $X = \pm R^2 \prod_{E_j=1} I_j \pmod n$ in step 1, and providing $Y = R$ in step 3. By using the GMR [8] idea of resettable simulation, M can mimic the communication in $(\bar{A}, \tilde{B})$ with an indistinguishable probability distribution in $O(t \cdot 2^k)$ expected time, which is polynomial in $|I|$ by our assumptions on k and t. Since the existence of S_j which satisfy $I_j S_j^2 = \pm 1 \pmod n$ is guaranteed for any I_j with Jacobi symbol $+1$ (recall our assumption that everybody knows that n is a Blum integer), and this property is checkable in polynomial time, the protocol is unrestricted input zero knowledge. □

An interesting modification can eliminate the public key directory and lead to a "keyless" identification scheme. It assumes that the trusted center (which knows the factorization of n) issues smart cards to users after properly checking their physical identity. No further interaction with the center is required either to generate or to verify proofs of identity. In this version of the scheme the center creates a string I which contains the user's name, ID number, physical description, or any other information provers or verifiers may want to establish. The only limitation on this string is that its length is logarithmic in the length of n, resulting in a polynomial number of possible identities. Thus, the greater the amount of information needed in order to recognize a user uniquely, the longer n must be. (This limitation on the length of the string is theoretically necessary for subtle reasons which are mentioned in the next paragraph, but in practice it has very little significance.) The public keys I_j are derived from I, by applying a randomly chosen but publicly known deterministic transformation $T(I, j)$, and the secret square roots S_j are computed and stored in the card by the center. We assume that the smart cards are built by some tamper-free technology, and thus the contents of their memories cannot be read or copied even by the users possessing the cards. When A wants to prove his identity, B can derive the I_j directly from A's claimed identity rather than from a public key directory. The actual identification protocol remains the same as in our original scheme, and it convinces B that A knows the S_j that correspond to those I_j. These values could only be computed by the trusted center when the real A requested a card.

The above convenient scheme offers provable security. No outsider is expected to produce a smart card by his own, as there is negligible probability that the set of numbers to which he can extract modular square roots intersects the set of possible I_j's. This claim is proved by a simple counting argument, assuming that the number of possible identities is polynomial in $|n|$ and that the transformation T is chosen randomly from a set of n possible transformations. The identification protocol, being a proof of knowledge, guarantees that provers do not cheat verifiers. The zero-knowledge property guarantees that no information leaks, and thus the users of the system gain no advantage in trying to copy or forge smart cards.

We want to point out the role of *unrestricted input* zero knowledge in this scheme. Consider the possible choices for the transformation T. In cases of unrestricted input proofs of knowledge, we may choose T by randomly choosing a fixed offset m and constructing $I_j = I + j + m \bmod n$ (if the result happens to have Jacobi symbol -1 it can be multiplied by a standard number with Jacobi symbol -1). Thus there is no problem in chosing T, and the transformation gives out no information. On the other hand, if I_j needs to be of some special form (quadratic residue), then the transformation by itself might reveal information. If we try to use the fixed-offset transformation as before, we run into trouble if the result happens to be a quadratic nonresidue with Jacobi symbol $+1$. This number cannot form the basis of a restricted input interactive proof of quadratic residuosity, and so the trusted center must discard it. From this we can deduce that the result was not a quadratic residue. It is true that this I_j was really computed at random (as m is chosen at random), and so knowing that it is a quadratic nonresidue does not constitute useful information. But this same phenomenon is likely to occur for many different strings I, and as these strings are not independent among themselves, we get many quadratic non-residues with dependencies which are difficult to analyze. These kind of problems were the motivation to introducing unrestricted input protocols to identity schemes.

4. The Parallel Version of the Identification Scheme

Interactive proofs are usually iterated versions of some basic protocol with a constant number of elementary operations and a constant probability of cheating. This can be done either by repeating the protocol as a whole (sequential execution) or by repeating each elementary operation by itself (parallel execution). Parallel executions change the direction of the communication only a constant number of times (which is preferable for theoretical as well as practical reasons), but unfortunately they are not zero knowledge for very subtle technical reasons.

The problem of sequential versus parallel executions of protocols has attracted considerable attention, and was dubbed "the case where intuition fails" in the literature. The information $\tilde{B}$ can compute as a result of his parallel interaction with $\bar{A}$ seems to be very specialized, and thus it is natural to speculate that parallel versions of zero-knowledge protocols release only "unusable" information. We formalize the above speculation and prove that our identification scheme is secure, *even if the parallel version of the identification protocols is used.*

We observe that identification schemes are typically composed of two stages:

1. Initialization. In this stage a link between each user U_j and its public key K_j is established (e.g., via distribution of directories by a trusted center).
2. Operation. In this stage any user U_j can demonstrate its identity by performing some identification protocol related to its public key K_j (e.g., demonstrate knowledge of a modular square root of K_j).

Let h be the security parameter of the identification scheme. As we are dealing only with polynomial-time users (verifiers and provers), the lifetime of a system is limited by some polynomial in h. Likewise, the number of users is polynomial in h, as otherwise no directory could be created. We do not want to fix the number of users *a priori*, and so we define:

Definition. A *polynomial ensemble of users PE* maps for each security parameter h a polynomial number (in h) of polynomial-time (in h) probabilistic Turing machines ("users").

A user is called a "good" user (denoted by $\bar{U}$), if throughout the lifetime of the identification system it does not deviate from the protocols dictated by the scheme. An ensemble of users may contain users which are good and users which are not. The security guarantee is given only to the good users.

Definition. An identification system scheme is *secure* if for any polynomial ensemble of users the probability of an impersonation event in a randomly chosen identification system is negligible. An impersonation event is the event that at least once during the lifetime of the system one user impersonates some other good user. More formally:

$$\forall PE, \forall c, \exists N \text{ such that } \forall h > N, \qquad \forall U_i, \bar{U}_j, \quad i \neq j,$$
$$\text{Prob}[(U_i, \bar{B}) \text{ accepts } K_j] < 1/h^c.$$

The probability is taken over the coin tosses of the complete history of the system up to the time of the attempted impersonation: the coin tosses of U_i, of U_j, of $\bar{B}$, of the trusted center (if there exists one), and of all other users. In particular, U_i is not expected to succeed in falsely representing himself as U_j even after verifying many real proofs of identity of U_j. This means that U_j's proofs of identity are not expected to leak any useful information. This does not mean that the proofs of identity are formally zero knowledge. In fact, there are two main differences between protocols which are zero knowledge and protocols which release no useful information (at least in our context of identification schemes):

1. The communication tape resulting from zero-knowledge interactive proofs must not contain distinguishing features which are difficult to reproduce by "ignorant" polynomial-time machines. With protocols which release no useful information we only demand that if such distinguishing features do occur, they would not help the verifier in extracting the prover's secret knowledge. Note that we do not exclude the possibility of disclosure of partial information, but again, as long as this partial information is not sufficient in order to perform the prover's part of the protocol.

2. The zero-knowledge property must hold with respect to *all* large enough common inputs. With "no useful information transfer" we no longer have this universal quantification over the input strings. Instead, we include the choice of the common input in the probability space. Thus, these protocols protect the privacy of the prover's extra knowledge only for *almost all* large enough common inputs. This change has no significance in most cryptographic applications, as the formal tool cryptography uses in order to define as "secret" the extra information the prover has, is by letting it be chosen at random.

The following lemma demonstrates that the parallel version of the identification protocol retains the completeness and soundness properties of the serial version of the protocol. This is a restricted analogue to Theorem 3, as we do not prove the zero-knowledge property. But as it turns out, this lemma is the first step toward proving that the parallel version releases no useful information.

Lemma 4. *The parallel identification protocol is a proof of knowledge of $P(I, S)$ for $k = O(\log\log n)$ and $t = \Theta(\log n)$.*

Proof. The proof of the completeness property is exactly the same as in the proof of Theorem 3.

Soundness: we prove that A must know a modular square root S_m of each one of the input strings I_m in order to have nonnegligible probability of executing $(A, \bar{B})$. Without loss of generality, we only show how M can extract S_1, by playing the role of $\bar{B}$ in a polynomial number of executions (A, M). Recall that M's advantage over $\bar{B}$ in extracting knowledge from A is his ability to reset A with the same random tape.

Assume that the contents of A's knowledge tape and random tape are such that the execution $(A, \bar{B})$ has probability of success ε. The probability is taken over the contents of $\bar{B}$'s random tape RB, which contains kt bits. Logically divide RB into two parts: RB_1 which produces t bits for E_1^1 to E_1^t, and RB_2 which, for each j, $2 \leq j \leq k$, produces the bits E_j^1 to E_j^t. The possible outcomes of the executions of $(A, \bar{B})$ can be summarized in a large boolean matrix H whose rows correspond to all possible choices of RB_2, its columns correspond to all possible choices of RB_1, and its entries are 0 if $\bar{B}$ rejects A's proof and 1 if $\bar{B}$ accepts A's proof. Note that this value is well defined since the executions become deterministic once RA and RB are chosen.

To extract S_1, M tries to find two 1's along the same row in H. M can probe H by executing (A, M) with properly chosen "random" strings RB_2 and RB_1, and its goal is to minimize the number of probes.

The fraction of 1's in H is at least ε, but their locations can be chosen by an adversary who knows the probing strategy and tries to foil it. We call a row "heavy" if the fraction of 1's along it is at least $\varepsilon/2$. Since the width of H is 2^t, and $\varepsilon > 2^{-t+1}$ (by our assumption that the probability of success is nonnegligible), a heavy row contains at least two 1's.

The obvious probing strategy uses $O(1/\varepsilon^2)$ probes in the following way:

1. Choose $O(1/\varepsilon)$ random rows.
2. Probe $O(1/\varepsilon)$ random entries in each row.

Since the fraction of heavy rows in H is at least $\varepsilon/2$ (the worst case being $\varepsilon/2$ rows which are all 1, and the rest of the rows with a fraction of 1's slightly less than $\varepsilon/2$), the first step chooses at least one heavy row with constant probability and the second step finds two 1's along this row with constant probability. However, a better probing strategy is:

1. Probe $O(1/\varepsilon)$ random entries in H.
2. After the first 1 was found, probe $O(1/\varepsilon)$ random entries along the same row.

Since at least half the 1's in H are located in heavy rows, this strategy succeeds with constant probability in just $O(1/\varepsilon)$ probes, which is polynomial in $|I|$ by our assumption that ε represents nonnegligible probability.

The two successes found by this probing strategy are located in the same row, and so M can extract S_1 by manipulating the verification conditions in a way similar to that used in the proof of Theorem 3. $\square$

Before we turn to the main theorem of this section, we sketch the intractability of factoring (IF) assumption: any family of boolean circuits, which can factor with nonnegligible probability properly chosen integers n (multiples of two primes of approximately equal size, each with large divisors of the order of its multiplicative subgroup), must grow in a rate faster than any polynomial in the size of the input.

Theorem 5. *Under the intractability of factoring assumption, the parallel version of the identification scheme is secure.*

Proof. The lifetime of our identity scheme is polynomial in its security parameter, and so is the number of users. This implies that the complete history of the system can be simulated by one polynomial-time procedure P. This procedure may be nonuniform, as it must incorporate the description of the algorithm each user runs, and this may change for different sizes of the security parameter. Thus P may be viewed as a family of boolean circuits with polynomial growth rate. By the IF assumption, the probability that P factorizes a large enough n which is properly chosen by the trusted center is negligible.

Assume that the identification scheme is not secure. This means that we are expected to reach a stage where $(U_i, \bar{B})$ has nonnegligible probability of success on input K_j, where $i \neq j$. By Lemma 4 this implies that with overwhelming probability, U_i can compute modular square roots for any one of the k numbers I_m comprising U_j's public key. We prove that the square roots that U_i knows are independent of the square roots that $\bar{U}_j$ knows, and thus with probability $1 - 1/2^k$ the two users hold between them all modular square roots of at least one input number. This implies factorization of n, and contradicts the IF assumption.

Each $\pm I_m$ has four square roots mod n, but only two of them are known to $\bar{U}_j$ (otherwise it could have factored n by itself). We claim that even an infinitely powerful $\tilde{B}$ cannot determine from the X's and Y's sent by $\bar{U}_j$ during the execution of $(\bar{U}_j, \tilde{B})$ which square roots $\bar{U}_j$ actually uses. To prove this, consider the defining equation for Y:

$$Y = R \cdot \prod_{E_l=1} S_l \pmod{n}.$$

If $\bar{U}_j$ replaces S_m by one of the other three square roots, the Y is multiplied by one of the three nontrivial square roots of 1. This effect can be canceled by dividing R by the same square root, which leaves $X = \pm R^2 \pmod{n}$ unchanged. Since the R's are randomly chosen, $\bar{U}_j$ produces the same X, Y values with the same probability distribution in both cases. This symmetry argument proves that $\bar{U}_j$ cannot leak to $\tilde{B}$ during the executions of $(\bar{U}_j, \tilde{B})$ which square root of Q he can compute from the S_m he knows. □

A crucial argument in the proof of Theorem 5 is that in order to execute U_j's protocol, U_i must *know* (or have a good chance to compute) some square roots. Thus, in the proof of the security of the parallel identification scheme we reap the reward of having defined the notion of proofs of knowledge.

Finally, we want to discuss improvements to the efficiency of the parallel identification protocol. In Lemma 4 we restricted k to $O(\log \log n)$ and t to $\Theta(\log n)$ in order to emphasize the similarity between the lemma and Theorem 3. But in Theorem 3 the restriction $k = O(\log \log n)$ was only used in the proof of the zero-knowledge property. As Lemma 4 does not attempt to prove this property, the condition on k can be relaxed to $k = O(\log n)$. Now consider the restriction $t = \Omega(n)$. This was necessary in the proof of Lemma 4, because the goal was to demonstrate that the prover knows a modular square root of each one of the I_m's comprising the input string. But a slight change in the proof of Theorem 5 shows that the proof carries through even if the prover U_i knows a square root of a product of the numbers in just one subset of the I_m's. If we change the goal of Lemma 4 to proving the above, we can relax the condition on t and replace it by a joint condition $kt = \Theta(\log n)$. The proof of the lemma is only simplified, as, instead of the boolean matrix H used in the proof of the lemma, we have just a Boolean vector.

The curve $kt = \Theta(\log n)$ essentially represents a trade between memory and randomness. For larger values of k, the prover needs more memory in order to save square roots of more numbers. For larger values of t, the prover needs to choose more random numbers R, as t corresponds to the number of iterations. Thus the interplay between k and t allows flexibility in the choice of parameters for the scheme. Additional information on the practical aspects of the identification scheme and its optimized implementations can be found in [5].

Acknowledgments

It is a pleasure to acknowledge the invaluable contributions of our many friends, who inspired, improved, and debugged our vague initial concepts. In particular we would like to thank Gilles Brassard, David Chaum, Oded Goldreich, Shafi Goldwasser, Joe Halpern, Rafi Heiman, Silvio Micali, Yoram Moses, Yair Oren, Charlie Rackoff, Ron Rivest, Martin Tompa, and Heather Woll.

References

1. Brassard, G., and C. Crepeau, Nontransitive Transfer of Confidence: A Perfect Zero Knowledge Interactive Protocol for SAT and Beyond, *Proceedings of FOCS*, 1986, pp. 187–195.

2. Chaum, D., Demonstrating that a Public Predicate Can Be Satisfied Without Revealing Any Information about How, *Proceedings of CRYPTO* 86, Lecture Notes in Computer Science, Vol. 263, Springer-Verlag, Berlin, 1987, pp. 195–199.
3. Chor, B., S. Goldwasser, S. Micali, and B. Awerbuch, Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults, *Proceedings of FOCS*, 1985, pp. 383–395.
4. Feige, U., Interactive Proofs, M.Sc. Thesis, Weizmann Institute of Science, 1987.
5. Fiat, A., and A. Shamir, How To Prove Yourself: Practical Solutions to Identification and Signature Problems, *Proceedings of CRYPTO* 86, Lecture Notes in Computer Science, Vol. 263, Springer-Verlag, Berlin, 1987, pp. 186–194.
6. Galil, Z., S. Haber, and M. Yung, A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public Key Cryptosystems, *Proceedings of FOCS*, 1985, pp. 360–371.
7. Goldreich, O., S. Micali, and A. Wigderson, Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design, *Proceedings of FOCS*, 1986, pp. 174–187.
8. Goldwasser, S., S. Micali, and C. Rackoff, Knowledge Complexity of Interactive Proof Systems, *Proceedings of STOC*, 1985, pp. 291–304.
9. Goldwasser, S., S. Micali, and R. Rivest, A Paradoxical Solution to the Signature Problem, *Proceedings of FOCS*, 1984, pp. 441–448.
10. Goldwasser, S., and M. Sipser, Arthur Merlin Games versus Interactive Proof Systems, *Proceedings of STOC*, 1986, pp. 59–68.
11. Halpern, J., and Y. Moses, Knowledge and Common Knowledge in a Distributed Environment, *Proceedings of PODC*, 1984, pp. 50–61.
12. Oren, Y., On the Cunning Power of Cheating Verifiers: Some Observations about Zero-Knowledge Proofs, *Proceedings of FOCS*, 1987, pp. 462–471.
13. Pollard, J. M., and C. P. Schnorr, An Efficient Solution of the Congruence $x^2 + ky^2 = m \pmod{n}$, *IEEE Transactions on Information Theory* (1987), 702–709.
14. Rivest, R., A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the Association for Computing Machinery*, **21** (1978), 120–126.
15. Rosenschein, S., Formal Theories of Knowledge in AI and Robotics, *New Generation Computing*, **3** (1985), 345–357.
16. Tompa, M., and H. Woll, Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information, *Proceedings of FOCS*, 1987, pp. 472–482.