Zero-Knowledge Proofs of Knowledge Without Interaction
(Extended Abstract)

ALFREDO DE SANTIS* GIUSEPPE PERSIANO†

Abstract

A *zero-knowledge proof system of knowledge* is a protocol between two parties called the prover and the verifier. The prover wants to convince the verifier that he "knows" the proof of a given theorem without revealing any additional information. This is different from a zero-knowledge proof system of *membership* where the prover convinces the verifier only of the veridicity of the statement.

Zero-knowledge proofs of knowledge are very useful tools in the design of secure protocols. Though, the concept of a proof of knowledge is a very subtle one and great care is needed to obtain a satisfying formalization.

In this paper, we investigate the concept of a zero-knowledge proof of knowledge in the non-interactive model of [5, 6]. Here, the prover and the verifier share a short random string and the only communication allowed is from the prover to the verifier. Although this is a simpler model than the interactive one, still formalizing zero-knowledge proofs of knowledge is a delicate task.

The main results of the paper are the following

- We present formal definitions for the concept of non-interactive zero-knowledge proofs of knowledge (NIZKPK). To reach a proper definition, it is necessary to distinguish between the cases in which the statement is given to the prover before or after the common string is presented. In the first scenario the theorem is independent from the common string, whereas in the second it might be chosen "maliciously" as a function of the common string. We also discuss a third different scenario which is the most useful for applications.
- We prove that NIZKPK's exist under complexity assumptions for all polynomial-time relations. In our construction it is sufficient for the prover to be an efficient algorithm that has a witness.
- We show that proving that one-way permutations (used as black-box functions) are sufficient for the existence of NIZKPK (where the prover is efficient) is as hard as proving $P \neq NP$.
- Finally, we demonstrate the applicability of the concept of NIZKPK to Cryptography by proving the security of a construction of public-key cryptosystems secure against chosen-ciphertext attacks based on NIZKPK.

*Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84081 Baronissi (SA), Italy. Partially supported by Italian Ministry of University and Research (M.U.R.S.T.) and by National Council for Research (C.N.R.) under grant 91.02326.CT12. E-mail: ads@udsab.dia.unisa.it

†Aiken Computation Laboratory, Harvard University, Cambridge, MA 02138, USA and Dipartimento di Informatica, Università di Salerno. Supported at Harvard by NSF Grant # NSF-CCR-90-07677. E-mail: persiano@endor.harvard.edu

1 Introduction

The concept of a "proof system of possession of knowledge" is one of the many contributions of [15]. It is a protocol between two parties called the prover and the verifier where the prover wants to convince the verifier that he "knows" the proof of a given theorem. This is a stronger notion of a

0-8186-2900-2/92 $3.00 © 1992 IEEE

proof system where the prover wants just to convince the verifier of the veridicity of the theorem in question. In the same paper, it was also proposed to formalize the concept of proof system of knowledge by using the notion of an *extractor*. The extractor is a polynomial-time algorithm that can use the prover as an oracle (for example, he can pretend to be the verifier and run the protocol and then "rewind" the prover and start again asking different questions) and produces a witness to the theorem in question. Following the suggestion of [15], [11] gave a formal definitions (other definitions have been given by [21] and [2]; see [2] also for a discussion of issues arising in defining interactive proofs of knowledge).

The relevance of zero-knowledge proofs of knowledge to the design of secure protocols has first been recognized in [13]. In their protocol for a secure oblivious transfer the receiver has to prove that he "knows" a square root of the integer he has just sent to the sender.

Encryption is a fundamental cryptographic primitive, which, by its own nature, is a non-interactive process. The construction of a cryptosystem secure against chosen-ciphertext attacks has been a long-standing open problem. In order to solve this problem, Blum, Feldman, and Micali [6] suggested that the sender proves "knowledge" (in a zero-knowledge and non-interactive fashion) of the message whose encryption he has sent. (A different construction, relying on non-interactive zero-knowledge proofs [5, 6], has been subsequently given in [19].)

However, while it is a simple exercise to show that most interactive zero-knowledge proofs (e.g., those of [16]) admit an extractor, the non-interactive case seems to present greater difficulty. Moreover, until now, no formalization of proofs of knowledge in the model of [5, 6] has been put forward.

Summary of results

Formalizing Possession of Knowledge. Following the lead of [15], we present the concept of a *Non-Interactive Zero-Knowledge Proofs of Knowledge* (NIZKPK in short). A NIZKPK is a non-interactive zero-knowledge proof system that admits an extractor. The extractor chooses a random string σ along with some auxiliary information (e.g., in our construction the factorization of the integers appearing in σ) and then feeds the prover with σ. From the proof given by the prover, the extractor using the auxiliary information relative to σ is able to compute a witness.

In our definition, we distinguish two cases depending on whether the theorem for which the prover wants to claim knowledge of a witness is chosen before the random string (and thus independently) or after the random string is presented to the prover.

Constructing NIZKPK. We show a general procedure that, under the assumption that *Dense Secure Public-Key Cryptosystems* exist, given a non-interactive zero-knowledge proof system of membership for an NP-complete theorem constructs a NIZKPK for all polynomial-time relations. A dense secure public-key cryptosystem is a particular kind of cryptosystem and it is possible to prove its existence assuming the difficulty of deciding quadratic residuosity, the security of inverting the RSA function and the security of the Diffie-Hellman secret key exchange.

One-way permutations and NIZKPK. We present strong evidence that the implication "if one-way permutations exist then NIZKPK for all polynomial-time relations exist" is not provable by standard techniques. We use the result of Impagliazzo and Rudich [17] by which proving the security of a secret-key agreement protocol that uses a one-way permutation as a black-box is as hard as proving $P \neq NP$. Our result is the first "impossibility" result about zero-knowledge proofs and suggests a separation between non-interactive and interactive settings with respect to zero-knowledge proofs.

Applications of NIZKPK. We show the applicability of NIZKPK to the design of secure protocols. In particular, we give the first proof of security for the construction of a cryptosystem secure against chosen-ciphertext attacks suggested in [6]. NIZKPK can also be used for a variety of other cryptographic applications such as memoryless secure digital signatures [3] and blind signatures which will be discussed in a forthcoming paper.

Notation. An efficient algorithm is an algorithm running in expected polynomial-time. If $A(\cdot)$ is a probabilistic algorithm, then for any input x, the notation $A(x)$ refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ.
If S is a probability space, then "$x \leftarrow S$" denotes the algorithm which assigns to x an element randomly selected according to S and $[S]$ denotes the set of strings to which S assigns positive probability. If F is a finite set, then the notation "$x \leftarrow F$" denotes the algorithm which assigns to x a randomly selected element of F. For example, the writing "$\sigma \leftarrow \{0,1\}^n$" means that σ is a randomly chosen n-bit string.
If $p(\cdot,\cdot,\cdots)$ is a predicate, the notation $\mathbf{Pr}(x \leftarrow S; y \leftarrow T; ... : p(x,y,\cdots))$ denotes the probability that $p(x,y,\cdots)$ will be true after the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, ...$.
By the writing $a \circ b$ we denote the string obtained concatenating the strings a and b. If a and b are binary strings of the same length, $a \oplus b$ denotes their bitwise XOR.

2 Non-Interactive Zero-Knowledge Proofs of Knowledge

Informally speaking, by a zero-knowledge proof system of knowledge we mean a protocol for two parties called the prover and the verifier where the prover convinces the verifier that he "knows" a witness to the fact that a certain input string x is in a language L without giving any additional information. For example, the prover convinces the verifier that he *knows* a legal 3-coloring of the input graph (not just prove that such a 3-coloring exists!) without giving any information on the coloring. A formalization of this notion has been given in [11] following the suggestion of [15] (see also [21] and [2] for discussion of the interactive case). Roughly speaking, according to [11] a prover gives proof of knowledge if there exists an efficient "extractor" that, using the prover as an oracle, can produce a witness to the theorem given in input. In interactive scenarios, the extractor is allowed to ask questions to the prover and then rewind the prover and ask different questions. It is not hard to see that, for example, the interactive zero-knowledge proofs of membership of [16] for NP are also proofs of knowledge.

In this work we consider the non-interactive model introduced in [6], where the prover and the verifier share a short random string and the communication is mono-directional: from the prover to the verifier only. In [5] it is proved that, under the quadratic residuosity assumption, all NP languages have zero-knowledge proofs in this non-interactive model. This has been subsequently improved in [12] that showed that trapdoor permutations are sufficient for efficient provers (see also [4]). We refer the reader to [5] for a formal definition of non-interactive zero-knowledge proof system (of membership).

Obtaining proofs of knowledge in this non-interactive setting does not seem to be as easy as in the usual interactive scenario. One of the major technical obstacle to this, is the fact that rewinding the prover does not seem to be of any help in this non-interactive settings, as the verifier (and hence the extractor) is not allowed to ask questions and therefore the extractor has to be substantially different from the ones used in the interactive case. In our model, the extractor works in two phases. First, he computes a string σ, which is given to the prover, along with some private information. In the second phase, the extractor, using the private information relative to σ, extracts a witness to the theorem from the non-interactive proof computed by the prover using σ.

In this section, we consider two different scenarios. The main difference between these scenarios is when the theorem to be proved is chosen: before, and thus independently from the common string, or after seeing the common string.

Before defining NIZKPK, we need the following two concepts.

Definition 1. A relation is a subset of $\{0,1\}^* \times \{0,1\}^*$. A *polynomial-time relation* $\mathcal{P}$ is a relation for which given an *instance* x and a *witness* w it is possible to check in time polynomial in $|x|$ whether $(x, w) \in \mathcal{P}$. For each polynomial-time relation $\mathcal{P}$ the language $L_{\mathcal{P}} = \{x : \exists w \text{ for which } (x, w) \in \mathcal{P}\}$ belongs to NP. Conversely, every NP language naturally defines a polynomial-time relation. Finally, if x is an instance, by W_x we denote the set of w s.t. $(x, w) \in \mathcal{P}$.

Definition 2. An extractor is a pair of probabilistic polynomial-time algorithms $(\mathbf{Ext_0}, \mathbf{Ext_1})$. Algorithm $\mathbf{Ext_0}$ on input 1^n outputs a pair (σ, aux), with $|\sigma| = n$.

Definition 3. Non-Interactive Zero-Knowledge Proof of Knowledge. A NIZKPK for a polynomial-time relation $\mathcal{P}$ is a non-interactive zero-knowledge proof system of membership (`Prover`, `Verifier`) for the language $L_{\mathcal{P}}$ such that

Validity. There exists an extractor $(\mathbf{Ext_0}, \mathbf{Ext_1})$ such that for all algorithms `Adversary`, for all constants $a > 0$, all sufficiently large n, and all x,

$$\begin{aligned}&\mathbf{Pr}((\sigma, aux) \leftarrow \mathbf{Ext_0}(1^n);\\ &\quad \Pi \leftarrow \mathtt{Adversary}(\sigma, x);\\ &\quad w \leftarrow \mathbf{Ext_1}(\sigma, aux, x, \Pi) :\\ &\qquad (x, w) \in \mathcal{P}) \geq p_{n,x} \cdot (1 - n^{-a}),\end{aligned}$$

where $p_{n,x}$ denotes the probability

$$\begin{aligned}&\mathbf{Pr}(\sigma \leftarrow \{0,1\}^n;\ \Pi \leftarrow \mathtt{Adversary}(\sigma, x) :\\ &\qquad \mathtt{Verifier}(\sigma, x, \Pi) = 1).\end{aligned}$$

Remarks. The Validity property holds with respect to any algorithm `Adversary`. That is, `Adversary` can even be an all-powerful machine and nevertheless the extractor is able to get a witness from his proof. This requires that the string σ output by $\mathbf{Ext_0}$ is uniformly distributed, that is it has the same distribution of the common string, or is statistically close to it. Also notice that in the above definition the instance x is given to `Adversary` together with the string σ; that is, `Adversary` is not allowed to choose x as a function of σ.

In our definition the extractor algorithm is required to have probability of success to be no significantly smaller than $p_{n,x}$. However, as the theorem is fixed and independent from the random string, it is possible to increase the probability of extracting a witness to $1 - 2^{-n}$ in the case that $p_{n,x}$ is greater than an inverse polynomial. This can be achieved by repeating the process a polynomial number of times and each time with a different string σ.

It is easy to see that the zero-knowledge proof system for quadratic non residuosity of [5] is indeed a NIZKPK for factorization. We have thus the following theorem.

Theorem 1. There exists a NIZKPK for the relation

$$\{(x, fac) | x \in \textit{Regular} \text{ and } fac \text{ is } x\text{'s factorization}\},$$

where *Regular* is the set of integers which are not perfect squares and have exactly two prime factors.

Strong NIZKPK. Let us now consider the following scenario. First, the random string σ is chosen at random. After seeing the string σ, `Adversary` comes up with an instance x and a Π that convinces with some probability `Verifier` that he knows a witness for x. The strong validity property requires that in this case the extractor, which as before works in two phases, has a no significantly smaller probability of obtaining a witness for x. We define a Strong NIZKPK to be a NIZKPK with the strong validity property. In other words, in a strong NIZKPK, even if `Adversary` chooses x in a malicious way after seeing the string σ and convinces `Verifier`, the extractor can still obtain a witness for x with a non negligible probability. This is captured by the following definition.

Definition 4. Strong NIZKPK. A *Strong* NIZKPK system for a polynomial-time relation $\mathcal{P}$ is a NIZKPK (`Prover`, `Verifier`) such that

Strong Validity. There exists an extractor $(\mathbf{Ext_0}, \mathbf{Ext_1})$ such that for any algorithm `Adversary`, for all constants $a > 0$, and all sufficiently large n,

$$\begin{aligned}&\mathbf{Pr}((\sigma, aux) \leftarrow \mathbf{Ext_0}(1^n);\\ &\quad (x, \Pi) \leftarrow \mathtt{Adversary}(\sigma);\\ &\quad w \leftarrow \mathbf{Ext_1}(\sigma, aux, x, \Pi) :\\ &\qquad (x, w) \in \mathcal{P}) \geq p_n \cdot (1 - n^{-a}),\end{aligned}$$

where p_n denotes the probability

$$\mathbf{Pr}(\sigma \leftarrow \{0,1\}^n; (x, \Pi) \leftarrow \texttt{Adversary}(\sigma) : \texttt{Verifier}(\sigma, x, \Pi) = 1).$$

Notice that as for Definition 3, **Adversary** is not restricted to run in polynomial time. Also, the probability of success of the extractor algorithm is bounded by the probability that the specific prover at hand convinces **Verifier**. Unlike in the interactive case or when we only consider validity, this probability cannot be pumped-up. In fact, if we try to run **Adversary** with the same random string we might obtain the same string Π which clearly would not help. On the other hand, if we feed **Adversary** with a different random string σ, then he might choose a different theorem and this, again, would not be of any help.

We now present a definition of non-interactive zero-knowledge proofs of knowledge due to Silvio Micali.

Definition 5. Non-Interactive Zero-Knowledge Proof of Witness Possession. A NIZKWP for a polynomial-time relation $\mathcal{P}$ is a non-interactive zero-knowledge proof system of membership (Prover, Verifier) for the language $L_{\mathcal{P}}$ such that the following condition holds.

There exists an extractor ($\mathrm{Ext}_0, \mathrm{Ext}_1$) such that for all sufficiently large n, for almost all $\sigma \in \{0,1\}^n$, for all x, and for all π
if $(\sigma, aux) \in [E_0(1^n)]$ and $\texttt{Verifier}(\sigma, x, \pi) = 1$ then $E_1(\sigma, aux, x, \pi) \in W_x$.

The main difference between the concept of NIZKPK and NIZKWP consists in the fact that NIZKPK considers the set of all σ's for which the adversary can construct a convincing proof; NIZKWP is instead a string-by-string definition. We will further explore the relations between the two definitions in the final version of this paper.

3 Constructing a NIZKPK

In this section we show a general procedure that, given a non-interactive zero-knowledge proof system of membership for an NP-complete language, constructs a strong NIZKPK for all polynomial time relations, under the assumption that dense secure public-key cryptosystems exist. Informally speaking, a *d-Dense Secure Public-Key Cryptosystem*, (for d constant) is a secure cryptosystem such that

1. Each n-bit string defines an encryption algorithm.
2. If we pick an n-bit string at random, then we have probability at least n^{-d} of picking a *secure* encryption algorithm.
3. It is possible to generate uniformly distributed n-bit descriptions of encryption algorithms along with their corresponding decryption algorithms.

The assumption of the existence of dense secure public-key cryptosystems is provably no stronger than assuming the difficulty of breaking RSA, deciding quadratic residuosity and breaking the Diffie-Hellman key-exchange protocol. In fact, each of the assumptions above implies the existence of dense secure public-key cryptosystems.

We now show how dense secure cryptosystems can be constructed under the assumption of the difficulty of deciding quadratic residuosity. More precisely, the *Quadratic Residuosity Assumption* (QRA) states that there exists no efficient algorithm that decides the quadratic residuosity modulo a randomly chosen integer x product of two primes of the same length of a randomly chosen element y with Jacobi symbol $+1$ with probability significantly better than $1/2$ (see [14] for the number theoretic background). The Quadratic Residuosity Assumption has been introduced by Goldwasser and Micali [14] and has since been widely used in cryptography (see, e.g. [5]).

Theorem 2. Under the QRA, dense secure cryptosystems exist.

Proof's sketch: Our construction is based on the Goldwasser–Micali probabilistic encryption scheme [14] which we briefly discuss here.

The encryption $\texttt{GM}(x, y, b)$ of a bit b with the key (x, y) (where y has Jacobi symbol modulo x $+1$) is computed by choosing a random $r \in Z_x^*$ and setting $\alpha = y^b r^2 \bmod x$. If y is a quadratic non residue modulo x then it is possible to recover the bit b from α in polynomial-time using x's factorization. If y is a quadratic residue, instead, b cannot be recovered. If x is product of two primes of the same length and y has Jacobi symbol $+1$ then, under the QRA, no polynomial-time algorithm that does not know x's factorization can guess b with probability of success significantly better than $1/2$.

Let us now briefly sketch the construction of a dense secure cryptosystem based on the QRA. A string σ of length $n = m(m+1)$ defines an encryption algorithm in the following way. The string σ is seen as the concatenation of $m+1$ strings $x, y_1, \cdots, y_m$ of length m. A bit b is encrypted by computing $\mathtt{GM}(x, y_i, b)$, for all $1 \leq i \leq m$.

Let us now see why this constitutes a dense secure cryptosystem. First of all, with high probability, at least one of the y_i's is a quadratic non residue modulo x and thus decryption is possible. Second, from the result on the distribution of primes it follows that with probability $\Omega(n^{-2})$ a random x is product of two primes and the corresponding encryption scheme is secure. Finally, to generate pairs of encryption/decryption keys, we use the algorithm of [1] that in expected polynomial-time returns a random integer of given length in factored form. ■

The following two theorems also hold.

Theorem 3. If the RSA encryption scheme is secure then dense secure cryptosystems exist.

Theorem 4. If the Diffie-Hellman secret-key exchange protocol is secure then dense secure cryptosystems exist.

Secure public-key systems vs dense secure public-key system. A secure public-key cryptosystem (see [14, 18]) is simply an efficient algorithm that outputs the description of a secure encryption algorithm along with the corresponding decryption algorithm. The existence of dense secure public-key cryptosystems seems to be a stronger assumption than the existence of secure public-key cryptosystems. In fact by comparing the two definitions, we note that, in the definition of secure public-key cryptosystem, nothing is guaranteed about the density of secure encryption algorithms among all the descriptions of a certain size; it might be well the case that only a negligible fraction is secure. Moreover, it does not guarantee that is possible to generate the encryption algorithm by itself without the decryption algorithm. As we shall see, both of these properties are necessary for our construction.

Theorem 5. If dense secure public-key cryptosystems and non-interactive zero-knowledge proof systems of membership for an NP-complete language exist then all polynomial-time relations have a Strong NIZKPK.

Proof's sketch: Let $\mathcal{P}$ be any polynomial-time relation, L an NP-complete language, (P, V) a Non-Interactive Zero-Knowledge Proof System of Membership for L and consider the following pair $(\mathtt{Prover}, \mathtt{Verifier})$.

Suppose that `Prover` has an instance x and a witness w such that $\mathcal{P}(x, w)$ holds and wants to convince `Verifier` that he knows such a witness without giving any additional information. `Prover` proceeds as follows.

Let $\sigma = \rho \circ \tau$ be the random reference string.

1. `Prover` splits randomly the witness w into n^{d+1} pieces $w_1, \cdots, w_{n^{d+1}}$ such that $w = w_1 \oplus \cdots \oplus w_{n^{d+1}}$.
2. `Prover` considers the random string ρ as the description of n^{d+1} randomly chosen encryption algorithms $E_1, \cdots, E_{n^{d+1}}$ and computes the encryptions $\alpha_i = E_i(w_i)$, $i = 1, 2, \cdots, n^{d+1}$.
3. `Prover` runs P's algorithm on input the random string τ to prove non-interactively and in zero-knowledge that the encryptions α_i define a witness for the instance x. This is done by computing y using the known reduction and proving that $y \in L$.

   Notice that this is a proof of membership. In fact, here `Prover` needs to prove only that his construction has been performed correctly; that is, that the α_i's are encryptions of strings whose xor is a witness for the instance x.

Let us convince ourselves that the above proof system is zero-knowledge. The proof given by the `Prover` consists of the encryptions $\alpha_1, \cdots, \alpha_{n^{d+1}}$ and a zero-knowledge proof of correctness. However, as the cryptosystem is d-dense, with very high probability at least one of the encryption algorithms E_i is secure and thus at least one of the strings w_i cannot be recovered by the verifier. Therefore, as w is the xor of the w_i's, with very high probability `Verifier` cannot get any information about w from the α_i's. Moreover, the proof given at step 3 is zero-knowledge and thus does not give any information about w.

For the validity property, let us consider the following extractor $(\mathbf{Ext}_0, \mathbf{Ext}_1)$. $\mathbf{Ext}_0$ randomly generates n^{d+1} encryption algorithms $E_1, \cdots, E_{n^{d+1}}$ along with their decryption algorithms $D_1, \cdots, D_{n^{d+1}}$ and feeds the prover with the string consisting of the description of the encryption algorithms $E_1, \cdots, E_{n^{d+1}}$. Now, the prover will produce the proof of the theorem he has chosen. In particular, he will compute the encryptions α_i from which $\mathbf{Ext}_1$, by using the decryption algorithms D_i, can compute the strings w_i and thus a witness to the theorem. ■

4 NIZKPK and One-way permutations

In this section, we present strong evidence that the implication "if one-way permutations exist then NIZKPK's for all polynomial time relations exist" is not provable by standard techniques.

Our proof uses the result of [17] by which proving the security of a secret-key agreement protocol that uses a one-way permutation as a black box is as hard as proving $P \neq NP$ and proceeds as follows. First, we show that NIZKPK and one-way permutations are sufficient for the existence of a secure public-key cryptosystem. This implies that if one-way permutations are sufficient for NIZKPK, then it is possible to construct a public-key cryptosystem, and thus a secret-key agreement proven secure based solely on one-way permutations. This is as hard as proving $P \neq NP$.

Our result is the first "impossibility" result about zero-knowledge proofs and suggests a separation between non-interactive and interactive settings with respect to zero-knowledge proofs.

In what follows, we actually show that even proving that one-way permutations are sufficient for a very weak form of NIZKPK, which we call Weak NIZKPK, is as hard as proving $P \neq NP$. In a Weak NIZKPK, the extractor $(\mathbf{Ext}_0, \mathbf{Ext}_1)$ is only required to be able to extract the witness to a theorem from the proof provided by `Prover`. That is, we have eliminated the quantification over all possible `Adversary`.

Theorem 6. One-way permutations and Weak NIZKPK for all polynomial-time relations imply secure public-key cryptosystems.

Proof's sketch: Let f be a one-way permutation and h be a hard-bit predicate for f. Each user U constructs his public file by running the algorithm $\mathbf{Ext}_0$ with input 1^n (n is the security parameter). The algorithm $\mathbf{Ext}_0$ returns a random string σ, which is placed in the public file, and some secret information aux which is kept in U's private file. To encrypt a bit b, the sender does the following. Randomly chooses a $x \in \{0,1\}^n$ such that $h(x) = b$ and sends to U the value $y = f(x)$ along with a zero-knowledge proofs of knowledge based on the string σ that he knows x such that $y = f(x)$. To decrypt, user U runs the algorithm $\mathbf{Ext}_1$ on input the proof he has just received.

Why is this a secure public-key cryptosystem? First of all, user U can decrypt. In fact, the algorithm $\mathbf{Ext}_1$ on input a zero-knowledge proof of knowledge is guaranteed to extract a witness to the theorem (in our case x, from which $b = h(x)$ can be easily computed). Moreover, the cryptosystem is secure. This can be proved from the fact that h is hard for f (thus knowing $f(x)$ does not give any information about $h(x)$) and from the fact that, being the proof zero-knowledge, it is not possible to obtain any information about the witness of the theorem being proved. ■

5 A Cryptosystem Secure Against Chosen-Ciphertext Attacks

The problem of constructing a public-key cryptosystem secure against chosen-ciphertext attacks has been one of the most important problems in Cryptography since the introduction of the public-key model [9]. In a chosen-ciphertex attack, the adversary has access to the decryption equipment and can ask the decryption of adaptively chosen ciphertext. Then, he is challenged to decrypt a new ciphertext α.

Blum, Feldman, and Micali [6] suggested that the sender proves "knowledge" (in a zero-knowledge and non-interactive fashion) of the message whose encryption he has sent. More precisely, they proposed the following encryption scheme. The public file of a user U contains a secure encryption algorithm E and a random string σ. The encryption of a message m consists of $E(m)$ and a non-interactive zero-knowledge proof of knowledge that the sender actually knows the encrypted message. The decryption algorithm first checks the proof and, only if convinced, decrypts the message. The rationale behind this construction is that conducting a chosen-ciphertext attack the adversary can only get decryptions of ciphertexts of which he knows the cleartext. Clearly, this would not help.

Subsequently, Naor and Yung [19] gave a different construction for a cryptosystem secure against chosen-ciphertext attacks. In their scheme, the public file of a user U contains a random string σ and a pair of secure (but not against chosen-ciphertext attacks) encryption algorithms E_1 and E_2. The encryption of a message m consists of the two encryptions $E_1(m)$ and $E_2(m)$ and of a zero-knowledge proof that the same message was encrypted by using the two algorithms. The proof is a non-interactive zero-knowledge proof of membership with random string σ. Thus, the construction of Naor and Yung dispenses with the need for a zero-knowledge proof of knowledge and can be based on the no-stronger assumption of the existence of trapdoor permutations [12]. In [10], it has been presented a cryptosystem secure against the stronger attack in which the adversary knows in advance the encryption he has to break. This kind of attack has been first discussed by Rackoff and Simon (see [20]).

We now prove that a secure cryptosystem and a Strong NIZKPK can be used to build a new cryptosystem secure against chosen ciphertext attacks.

For the construction of cryptosystems secure against chosen-ciphertext attacks (and in general for any cryptographic application) we need a stronger kind of NIZKPK. The strong validity property requires the extractor algorithm to be able to extract a witness from proofs of theorems chosen after seeing the random string. In some applications, we need to be able to extract witnesses and to simulate proofs for theorems chosen by an adversary with respect to the same common string. For example, in the proof (see Lemma 1) that our proposed cryptosystem is secure against chosen-ciphertext attacks, the adversary can ask for the decryptions of theorems of his choice (which involves the ability to extract witnesses) and we need to produce a challenge for him to break (which requires the ability to simulate proofs). This calls for a blending of the validity and zero-knowledge property. More precisely, we will use three polynomial-time algorithms `Init`, `Ext`, and `Sim`. `Init`, on input 1^n prepares a string σ of length n (it is sufficient for σ to be only indistinguishable from a truly random string as for the applications we are only dealing with polynomial-time algorithms) along with some private information aux which will enable `Ext` to extract the witness from "real" proofs and `Sim` to simulate proofs. Such systems can be constructed using only Strong NIZKPK in the following way. Let (P, V) be a Strong NIZKPK with polynomial-time prover and an on-line simulator. An on-line simulator (see [8]) $S = (S_0, S_1)$ is a pair of efficient algorithms. S_0 receives as input 1^n and outputs σ, a string of length n, and aux, auxiliary information about σ. Then S_1, on input x and aux, produces a "proof" Π of knowledge of a witness for x s.t. the pair (σ, Π) is indistinguishable from the view of V on input x. It is possible to show that if Strong NIZKPK exist for all polynomial-time relations so do Strong NIZKPK with on-line simulator.

Let us now consider the following pair (`Prover`, `Verifier`).

Input: $(x, w) \in \mathcal{P}$ and string $\rho = \rho_0 \circ \cdots \rho_{2n}$, where $|\rho_i| = n$.

```
1. Randomly choose $n$ different indices $j_1, \cdots, j_n \in \{1, \cdots, 2n\}$.
2. For $i = 1, \cdots, n$
       Run $P$ on input $\rho_{j_i}, x, w$ to obtain $\alpha_i$, a
```

proof that he knows a witness for x.

3. Run P on input ρ_0 to obtain a proof that he knows $\alpha_1, \cdots, \alpha_n$ and $j_1, \cdots, j_n$ s.t., for all i, V on input $(x, \rho_{j_i}, \alpha_i)$ accepts. Let Π be P's output.
4. Send Π to `Verifier`.

Now let $(\mathtt{Ext}_0, \mathtt{Ext}_1)$ and (S_0, S_1) be the extractor and the simulator for (P, V). Algorithm `Init`, `Ext` and `Sim` are defined as follows.

`Init` randomly chooses a subset $\mathcal{I}$ of n indices in $\{1, \cdots, 2n\}$. For each $i \in \mathcal{I}$, `Init` constructs ρ_i, and its relative information aux_i, by running S_0 on input 1^n. For indices $0 \leq i \leq 2n$ not in $\mathcal{I}$, ρ_i and aux_i are computed by running $\mathtt{Ext}_0$ on input 1^n. The output of `Init` consists of $\rho = \rho_0 \circ \cdots \circ \rho_{2n}$ and $aux = aux_0 \circ \cdots \circ aux_{2n} \circ \mathcal{I}$.

Algorithm `Ext` on input an instance x and a proof Π proceeds as follows. First, it runs $\mathtt{Ext}_1$ on input ρ_0, aux_0, x and Π (notice that `Init` always construct ρ_0 by running $\mathtt{Ext}_0$) to obtain $\alpha_1, \cdots, \alpha_n$ and the indices $j_1, \cdots, j_n$. Let j_i be an index not in $\mathcal{I}$; if no such j_i exists `Ext` fails, but this happens with very low probability. `Ext` outputs w computed by running $\mathtt{Ext}_1$ on input ρ_{j_i}, aux_{j_i}, x and α_i.

Let us now describe `Sim`. `Sim` receives as inputs ρ, aux and x and has to simulate a proof of knowledge of a witness for x. This is accomplished by running S_1 on input ρ_i, aux_i and x for all $i \in \mathcal{I}$ to obtain α_i (remember that for $i \in \mathcal{I}$, ρ_i has been "prepared" by S_0). Finally, the proof Π is computed by running P on input ρ_i, for $i \in \mathcal{I}$, and the α_i's so obtained.

Let us now describe our cryptosystem secure against chosen-ciphertext attacks. Let (`Prover`, `Verifier`) be the NIZKPK described above and $\mathcal{G}$ be a secure public-key cryptosystem.

Construction of public and private keys.
Input: security parameter 1^n.

1. Compute $(E(\cdot,\cdot), D(\cdot)) = \mathcal{G}(1^n)$.
2. Randomly choose $\sigma \in \{0,1\}^n$.

The public file of user U is $PK_U = (\sigma, E)$ and U's secret file consists of $SK_U = D$.

Encryption of a bit. To encrypt a bit b with public key (σ, E), the user performs the following computations:

1. Randomly choose a n-bit string r and compute $\alpha = E(b, r)$.
2. Run `Prover` and compute Π, a proof (based on the random string σ) that the sender "knows" b, r such that $\alpha = E(b, r)$.
3. The encryption of the bit b consists of the pair (α, Π).

To decrypt the bit received, the receiver first checks that Π is a convincing proof. If convinced, he computes $D(\alpha)$ obtaining the bit b. If not convinced, he simply outputs "?".

Lemma 1. The above cryptosystem is secure against chosen-ciphertext attacks.

Proof's sketch: Assume it is not secure against chosen-ciphertext attacks. That is, there exists an efficient algorithm *Alg* that first asks for the decryption of polynomially many (say n^c, where $c > 0$ is a constant) adaptively chosen ciphertexts. Then, *Alg*, on input a randomly chosen encryption (α, Π), guesses the bit encrypted with probability significantly greater than 1/2. We use *Alg* to construct an algorithm that breaks the encryption E.
Input: a public key E and a randomly chosen encryption α.

1. Compute $(\sigma, aux) \leftarrow \mathtt{Init}(1^n)$.
2. For $i = 1, \cdots, n^c$
   Let (α_i, Π_i) be the ciphertext chosen by *Alg* on input σ, E and all decryptions previously received. If $\mathtt{Verifier}(\sigma, \alpha_i, \Pi_i)$ rejects then send "?" to *Alg*. Else use `Ext` on input $(\sigma, aux, \alpha_i, \Pi_i)$ to compute b_i, r_i such that $\alpha_i = E(b_i, r_i)$. Send b_i to *Alg*.
3. Compute Π by running `Sim` on input (σ, aux, α).
4. Run *Alg* on input (σ, E, α, Π) and let b be the output of *Alg*.

Output: b, as a guess of the bit encrypted by α.

We have only assumed *Alg* to be successful when he receives "real encryptions" and when σ is truly random. To prove that the above algorithm breaks α, we need to show that at step 4, *Alg* has still probability significantly better than 1/2 of success. This is proved by using the properties of (`Prover`, `Verifier`). ∎

We have thus the following theorem.

Theorem 7. If Strong NIZKPK with polynomial-time prover and on-line simulator exist for all polynomial-time relations then there exists a public-key cryptosystem secure against chosen-ciphertext attacks.

Acknowledgments
We thank Silvio Micali for many stimulating discussions and for letting us include some of his ideas in this paper. We acknowledge discussions with Benny Chor, Ivan Damgård, and Phil Rogaway. We thank Oded Goldreich for his helpful comments.

The second author would like to thank Michael Rabin for his constant encouragement and support.

Literature Referenced

[1] E. Bach, *How to Generate Factored Random Numbers,* SIAM Journal of Computing, vol. 17, n. 2, April 1988.

[2] M. Bellare and O. Goldreich, *On Defining Proofs of Knowledge*, to appear in Proceedings of CRYPTO '92.

[3] M. Bellare and S. Goldwasser, *A New Paradigm for Digital Signatures and Message Authentication Based on Non-Interactive Zero-Knowledge Proofs*, in "Advances in Cryptology - CRYPTO 89", vol. 435 of "Lecture Notes in Computer Science", Springer-Verlag.

[4] M. Bellare and M. Yung, *Certifying Cryptographic Tools: The case of Trapdoor Permutations*, to appear in Proceedings of CRYPTO '92.

[5] M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero Knowledge*, SIAM Journal on Computing, vol. 6, December 1991, pp. 1084–1118.

[6] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Applications*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 103–112.

[7] A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof-Systems*, in "Advances in Cryptology - CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag, pp. 52–72.

[8] A. De Santis and M. Yung, *Cryptographic Applications of the Non-Interactive Metaproof and Many-prover Systems*, in "Advances in Cryptology - CRYPTO 90", vol. 537 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 366–377.

[9] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Trans. on Information Theory, vol. 22, n.6, 1976, pp. 644–654.

[10] D. Dolev, C. Dwork, and M. Naor, *Non-Malleable Cryptography*, in Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, 1991, pp. 542–552.

[11] U. Feige, A. Fiat, and A. Shamir, *Zero-knowledge Proofs of Identity*, Journal of Cryptology, vol. 1, 1988, pp. 77–94.

[12] U. Feige, D. Lapidot, and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String,* Proceedings of the 22th Annual Symposium on the Theory of Computing, 1990, pp. 308–317.

[13] M. Fischer, S. Micali, and C. Rackoff, *A Secure Protocol for the Oblivious Transfer*, Eurocrypt 1984.

[14] S. Goldwasser and S. Micali, *Probabilistic Encryption*, Journal of Computer and System Science, vol. 28, n. 2, 1984, pp. 270–299.

[15] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989, pp. 186–208.

[16] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but their Validity, and a Methodology of Cryptographic Protocol Design,* Proceedings of the 27th IEEE Symposium on the Foundations of Computer Science, 1986, pp. 174–187.

[17] R. Impagliazzo and S. Rudich, *Limits on the Provable Consequences of One-way Permutations,* Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 1989, pp. 49–61.

[18] S. Micali, C. Rackoff, and B. Sloan, *The Notion of Security for Probabilistic Cryptosystems,* SIAM Journal on Computing, vol. 17, n. 2, April 1988.

[19] M. Naor and M. Yung, *Public-key Cryptosystems Probably Secure Against Chosen Ciphertext Attacks,* Proceedings of the 22th Annual ACM Symposium on Theory of Computing, 1990, pp. 427–437.

[20] C. Rackoff and D. R. Simon, *Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack*, in "Advances in Cryptology - CRYPTO 91", Ed. J. Feigenbaum, vol. 576 of "Lecture Notes in Computer Science", Springer-Verlag, pp. 433–444.

[21] M. Tompa and H. Woll, *Random Self-Reducibility and Zero-knowledge Interactive Proofs of Possession of Information,* Proceedings of 28rd Symposium on Foundations of Computer Science, 1987, pp. 472–482.