J. Cryptology (1996) 9: 129–148

Journal of
CRYPTOLOGY

The Power of Preprocessing in Zero-Knowledge Proofs of Knowledge*

Alfredo De Santis and Giuseppe Persiano
Dipartimento di Informatica ed App., Università di Salerno,
84081 Baronissi (Salerno), Italy
ads@udsab.dia.unisa.it
giuper@udsab.dia.unisa.it

Communicated by Claude Crépeau

Received 16 October 1992 and revised 27 March 1995

Abstract. We show that, after a constant-round preprocessing stage, it is possible for a prover to prove knowledge of a witness for any polynomial-time relation without any further interaction. The number of proofs that can be given is not bounded by any fixed polynomial in the size of the preprocessing. Our construction is based on the sole assumption that one-way functions and noninteractive zero-knowledge proof systems of *membership* exist.

Key words. Zero knowledge, Noninteractive proof systems, Proofs of knowledge.

1. Introduction

A *Zero-Knowledge Proof of Knowledge* (in short, ZKPK) is a protocol between two parties called the prover and the verifier. The prover wants to convince the verifier that he *knows* a witness w and that an input string x belongs to a language L (for example, x is a graph, L is the language of the Hamiltonian graphs, and w is a Hamiltonian path in x) without revealing any additional information. The concept of ZKPK is closely related to the concept of a zero-knowledge proof of *membership* where the prover wants to convince a polynomial-time verifier that an input string x belongs to the language L. It can be easily seen that in the interactive case ZKPK exist under the assumption of the existence of one-way functions since the zero-knowledge proof of membership for all NP of [24], as most proofs of membership proposed so far in literature, turn out to be also proof of knowledge.

* The research by Alfredo De Santis was partially supported by the Italian Ministry of the University and Scientific Research and by CNR. Part of Giuseppe Persiano's work was done while at Harvard University, partially supported by NSF Grant No. NSF-CCR-90-07677.

However, in the noninteractive setting, introduced in [7] and further developed and analyzed in [6], where prover and verifier share a random string, things are different. For example, it is not known whether the zero-knowledge proof of membership for 3SAT of [6] is also a proof of knowledge and actually it was suspected that no ZKPK existed in this setting. The authors, in [15], have presented a general procedure to construct noninteractive Zero-Knowledge Proofs of Knowledge (NIZKPK) from noninteractive zero-knowledge proofs of membership, thus proving the feasibility of the notion of proof of knowledge in the shared-string model. They also show that proving the existence of NIZKPK based only on one-way permutations is as hard as separating P from NP. This is unfortunate, as, besides having its own interest, ZKPK constitute an important building block of secure cryptographic protocols. The ability to give ZKPK in a noninteractive way would greatly reduce the communication complexity of many cryptographic applications and thus one would like to be able to base their existence on the most minimal assumption possible.

In this paper we consider the problem of obtaining NIZKPK based on the most minimal possible assumption and propose the concept of a *Noninteractive Zero-Knowledge Proof System of Knowledge with Preprocessing* (in short, NIZKPK with preprocessing).

In an NIZKPK with preprocessing, the prover and verifier perform a small interactive preprocessing stage during which the prover interactively proves that he has some specific knowledge (in our implementation the string he has committed to). Later, he can give to the verifier (or to anyone else who trusts the correctness of the interactive stage) any polynomial (in the length of the preprocessing stage) number of ZKPK. We prove the existence of NIZKPK with preprocessing under the weak assumption of the existence of one-way functions and of noninteractive zero-knowledge proofs of membership.

Our model perfectly fits many cryptographic scenarios in which two parties have the opportunity to interact for a while and establish some common knowledge and then they depart. For example, this is the case for a public-key cryptosystem where an interactive identification phase is needed when a user enters their public key into the public file. Also, this is the case for electronic cash protocols in which a bank and the user establish some common knowledge to open a bank account after which no further interaction is needed for the user to spend his money. We refer the reader to [14] for a communication-efficient protocol for electronic cash based on ZKPK with preprocessing.

Related Work

Noninteractive Zero Knowledge. The problem of reducing the communication needed for such a basic cryptographic primitive as zero-knowledge proof has recently emerged as a major line of research in Theoretical Cryptography. In [6] and [7] it is proved that interaction can be disposed of in zero-knowledge proofs of membership provided that a short random string is shared beforehand by the prover and verifier (see also [12]). More precisely, in [6] it is proved that under the quadratic residuosity assumption, a prover that shares a random string with a verifier can prove in writing (and thus noninteractively) and in zero-knowledge any polynomial number of NP theorems. Feige *et al.* [18] constructed a noninteractive zero-knowledge proof system for all NP based on certified trapdoor permutations. In this construction, as well as in that of [6], it is sufficient for the

prover to be a polynomial-time machine with a witness. Bellare and Yung showed [5] how to obtain the same result assuming only trapdoor permutations. If the prover has the power to invert one-way permutations, then the construction of [18] can be based on the existence of one-way permutations. In [11] it was shown how to obtain noninteractive zero-knowledge in a public-key scenario.

Noninteractive Zero Knowledge with Auxiliary Language. A model related to ours has been proposed in [13]. In this paper the notion of noninteractive zero-knowledge with auxiliary language is proposed, where all the interaction needed for a zero-knowledge proof is squeezed to an interactive preprocessing phase. The length of this interactive phase bounds the overall size of theorems that can be later proved in a noninteractive manner. Seen from a different angle, the prover needs to know the length of the theorems (but not the theorems themselves!) he wants to prove in advance. The implementation proposed in [13] is based on any one-way function and after a preprocessing of size n only theorems of total size at most $n^{1/3}$ can be noninteractively proved. It is interesting to note that, even though our model is a stronger one, we still manage to base our protocol on the very weak assumptions of the existence of one-way functions and noninteractive zero-knowledge proofs of language membership.

Constant-Round Zero Knowledge. A different approach to the study of communication in zero-knowledge proofs has consisted in studying the round complexity of zero-knowledge proofs. On the negative side, Goldreich and Krawczyk [23] have proved that only BPP languages have three-round zero-knowledge proofs in the original model of [25] where the verifier is a polynomial-time machine and no limitation is imposed on the computational power of the prover. Brassard *et al.* [10] have considered the dual case when the prover is a polynomial-time machine and the verifier may have infinite computing power. They presented constant-round zero-knowledge proofs for all NP in this model under the assumption of the existence of *one-way certified group actions* (e.g., discrete logarithm). Feige and Shamir [19] have instead obtained constant-round zero-knowledge proofs of knowledge for all NP in the case when both the prover and the verifier are polynomial-time machines and were able to base their construction on the weaker assumption of the existence of one-way functions. The problem of obtaining constant-round zero-knowledge proofs for all NP in the model of [25] remains open. Bellare *et al.* [4] have given constant-round *perfect* zero-knowledge proof systems for random self-reducible languages (e.g., graph isomorphism, quadratic residuosity).

Proofs of Knowledge. The relevance of ZKPK to the design of secure protocols was first recognized in [21]. Later, the concept of an interactive proof of knowledge was put forward in the proceedings version of [25], although no formal definition was given. Proofs of knowledge turned out to be a very subtle object difficult to formalize and, since then, various definitions have been proposed for the interactive case, the first being the one of [17]. The concept of proof of knowledge has also been explored in [30], that showed that all random self-reducible languages have such proofs. See also [3] for a discussion of definitional issues regarding proofs of knowledge. It is easy to see that the proofs of membership of [24] for all NP are also proofs of knowledge according to the

definition of [17]. Furthermore, the zero-knowledge proofs of [19] for the case when prover and verifier are both polynomial time are also proofs of knowledge.

The concept of a computationally convincing proof of knowledge where the prover is restricted to being polynomial time was studied in [9] where a definition capable of taking into account very nasty behaviors from the prover was proposed. In the same paper it was also proved that the constant-round arguments of [10] are also proofs of knowledge according to the proposed definition. A much easier to prove constant-round computationally convincing ZKPK is presented in [8].

The study of the noninteractive case is instead much more recent. In [15] the authors gave a definition of proof of knowledge in the shared-string model of [6] and showed the existence of ZKPK for all relations under complexity assumption. In the same paper they also showed that proving that one-way permutations are sufficient for their existence (remember that in the interactive case, one-way functions are sufficient for ZKPK) is as hard as separating P from NP.

Communication Complexity in Distributed Multiparty Computation. The communication complexity of a protocol is an important measure for distributed multiparty protocols. Bar-Ilan and Beaver [1] were the first to investigate the round complexity for secure function evaluation and exhibited a noncryptographic method that saves a logarithmic number of rounds. Beaver *et al.* [2] showed how any function can be securely computed using only a constant number of rounds of interactions, under the assumption that one-way functions exist. Feldman and Micali [20] gave a constant-round protocol for the well-studied Byzantine agreement problem.

2. Notation and Cryptographic Background

We denote by N the set of natural numbers. If S is a probability space, then "$x \leftarrow S$" denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation "$x \leftarrow F$" denotes the algorithm which assigns to x an element selected with uniform probability from the set F. For example, writing "$\sigma \leftarrow \{0,1\}^n$" means that the string σ is picked with uniform distribution among all strings of length n.

If $p(\cdot,\cdot,\ldots)$ is a predicate, the notation $\mathbf{Pr}(x \leftarrow S; y \leftarrow T; \ldots : p(x,y,\ldots))$ denotes the probability that $p(x,y,\ldots)$ will be true after the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, \ldots$.

The notation $\{x \leftarrow S; y \leftarrow T; \ldots : (x,y,\ldots)\}$ denotes the probability space over $\{(x,y,\cdots)\}$ generated by the ordered execution of the algorithms $x \leftarrow S$, $y \leftarrow T, \ldots$.

We say that a function $f\colon \mathrm{N} \to \mathrm{N}$ is negligible if for all constants d there exists a constant n_d such that for all $n > n_d$ it holds that $f(n) < n^{-d}$.

An *efficient* algorithm is a probabilistic algorithm running in expected polynomial time. An *efficient nonuniform algorithm* $D = \{D_x\}$ is a family of efficient algorithms where D_x has a program of size polynomial in $|x|$. That is, D_x has "wired-in" a polynomial amount of information about x.

The symbol $a \oplus b$ denotes the bitwise xor of the binary strings a and b. If a is shorter than b, then we pad a with zeros so that the two strings have the same length.

Let $\vec{x}$ be an m-ple $(x_1, x_2, \ldots, x_m)$ of strings. Writing $|\vec{x}|$ denotes the number of strings in $\vec{x}$ and when we write $\vec{x} \in L$ we mean that all the elements of $\vec{x}$ belong to L.

We now recall the notion of indistinguishability that is crucial to zero-knowledge. We address the reader to the original paper of [25] for motivation of this definition.

Definition 1 (Indistinguishability). Let $L \subseteq \{0, 1\}^*$ and let $\mathcal{U} = \{U(x) | x \in L\}$ and $\mathcal{V} = \{V(x) | x \in L\}$ be two families of random variables over $\{0, 1\}^*$. We say that $\mathcal{U}$ and $\mathcal{V}$ are indistinguishable (in symbols $\mathcal{U} \approx \mathcal{V}$) if, for all efficient nonuniform distinguishing algorithms $D = \{D_x\}$, for all positive constants c, and for all sufficiently long $x \in L$,

$$|\mathbf{Pr}(a \leftarrow U(x): D_x(a) = 1) - \mathbf{Pr}(a \leftarrow V(x): D_x(a) = 1)| < |x|^{-c}.$$

The following fact holds.

Fact 1. *If $\mathcal{U} \approx \mathcal{V}$ and $\mathcal{V} \approx \mathcal{Z}$, then $\mathcal{U} \approx \mathcal{Z}$.*

2.1. *Pseudorandom Collections of Functions*

In this section we briefly review the concept of a *pseudorandom collection of functions* that will play an important role in our construction of NIZKPK with preprocessing.

The concept of pseudorandom function has been introduced by Goldreich *et al.* [22]. Intuitively, we say that a collection of functions is pseudorandom if the value output of a function chosen at random from the collection on arguments chosen by a polynomial-time algorithm cannot be distinguished from the output of a truly random function. More formally, we have the following definition.

Definition 2. Let c be a positive integer constant. A *c-distinguishing* algorithm is an efficient nonuniform algorithm $D = \{D_{1^n}\}$ where each D_{1^n} has an oracle that computes an unknown (to D_{1^n}) function f: $\{0, 1\}^n \to \{0, 1\}^{n^c}$. We denote by $D_{1^n}^f$ the probability space induced by the output of D_{1^n} when given access to an oracle computing the function f.

Definition 3. Let c be a positive integer constant, let $\mathcal{R}_n$ be the set of all functions r: $\{0, 1\}^n \to \{0, 1\}^{n^c}$, and let $\mathbf{F} = \{f_s\}$ be a collection of functions where f_s: $\{0, 1\}^{|s|} \to \{0, 1\}^{|s|^c}$. We say that the collection $\mathbf{F}$ is a *c-pseudorandom* collection of functions if, for all c-distinguishing algorithms $D = \{D_{1^n}\}$ and for all constants $d > 0$,

$$|\mathbf{Pr}(r \leftarrow \mathcal{R}_n: D_{1^n}^r = 1) - \mathbf{Pr}(s \leftarrow \{0, 1\}^n: D_{1^n}^{f_s} = 1)| < n^{-d}.$$

The following theorem shows that the existence of one-way functions is sufficient for the existence of c-pseudorandom collections of functions for all positive integer constants c.

Fact 2 [22], [27], [26]. *If one-way functions exist then, for all positive integer constants c, there exist c-pseudorandom collections of functions.*

2.2. *Secure Commitment Schemes*

A secure commitment scheme is a protocol for two polynomial-time parties. It allows one party, the committer, to commit to a bit b in such a way that he can later show to the other party, the decommitter, the bit he has committed to. However, until b is decommitted, the decommitter cannot predict the bit with probability significantly better than $1/2$ and, once the bit has been committed to, the committer cannot change it. Our formal definition has been inspired by the scheme proposed by Naor [29].

Definition 4. A *secure commitment scheme* is a pair (C, D), where C is an efficient algorithm and D is a deterministic polynomial-time algorithm, such that:

1. **Meaningfullness.** For all $b \in \{0, 1\}$, for all constants $d > 0$, and for all sufficiently large n,

$$\mathbf{Pr}(\sigma \leftarrow \{0, 1\}^n; (\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma, b): D(\mathtt{com}, \mathtt{dec}, \sigma) = b) > 1 - n^{-d}.$$

2. **Uniqueness of decommitment.** For all efficient algorithms $\mathtt{Adv}$, for all constants $d > 0$, and for all sufficiently large n,

$$\mathbf{Pr}(\sigma \leftarrow \{0, 1\}^n; (\mathtt{com}, \mathtt{dec}_0, \mathtt{dec}_1) \leftarrow \mathtt{Adv}(\sigma): D(\mathtt{com}, \mathtt{dec}_0, \sigma) = 0$$
$$\text{AND } D(\mathtt{com}, \mathtt{dec}_1, \sigma) = 1) < n^{-d}.$$

3. **Indistinguishability of commitment.** The families of random variables $\{V_0(\sigma) | \sigma \in \{0, 1\}^*\}$ and $\{V_1(\sigma) | \sigma \in \{0, 1\}^*\}$, where

$$V_b(\sigma) = \{(\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma, b) : \mathtt{com}\},$$

are indistinguishable.

We now briefly comment on our definition. The mechanics of a commitment is the following. A random string σ of length n is given to both parties (as we shall see later, this string can be chosen by the decommitter). Then, to commit to a bit b, the committer runs algorithm C and obtains a pair of strings $(\mathtt{com}, \mathtt{dec})$. The string $\mathtt{com}$ is given to decommitter, while $\mathtt{dec}$ is kept secret. To decommit to the bit, the string $\mathtt{dec}$ is revealed and the decommitter can compute, by running algorithm D, the b that has been originally committed to. Property 1 says that if both parties follow the protocol, then the protocol succeeds with very high probability. Property 2 guarantees that it is very unlikely that a committer manages to compute for the same string $\mathtt{com}$ two different strings $\mathtt{dec}_0$ and $\mathtt{dec}_1$ that convince the decommitter that the bit committed is a 0 and 1, respectively. The third property says that, for all sufficiently long strings σ, no polynomially bounded algorithm can distinguish between a commitment of a 0 and a commitment of a 1. As this property holds for all sufficiently long strings σ, we can let the decommitter pick σ. In fact, he has nothing to gain by choosing σ in some special way; but a careless choice of σ might allow the committer to cheat.

The notion of a secure commitment scheme can be easily extended to any string x. If $x = b_1 \cdots b_m$ is an m-bit string, then $C(\sigma, x)$ is intended to consist of the m different commitments $C(\sigma, b_i)$, $i = 1, \ldots, m$. In this case, it can be seen that property 3 can be extended to any two strings. That is:

3. **Indistinguishability of string-commitment.** Let I_1 and I_2 be two subsets of $\{0, 1\}^*$ such that, for all n, $|I_1 \cap \{0, 1\}^n| = |I_2 \cap \{0, 1\}^n| = 1$. Then the families of random variables $\{V_1(\sigma)|\sigma \in \{0, 1\}^*\}$ and $\{V_2(\sigma)|\sigma \in \{0, 1\}^*\}$, where

$$V_i(\sigma) = \{s \leftarrow I_i \cap \{0, 1\}^{|\sigma|}; (\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma, s): \mathtt{com}\},$$

are indistinguishable.

In what follows we use the following equivalent property.

Lemma 1. *Let I be a subset of $\{0, 1\}^*$ such that, for all n, $|I \cap \{0, 1\}^n| = 1$ and let (C, D) be a secure commitment scheme. Then the families of random variables,*

$$V(\sigma) = \{s \leftarrow I \cap \{0, 1\}^{|\sigma|}; (\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma, s): \mathtt{com}\}$$

and

$$R(\sigma) = \{r \leftarrow \{0, 1\}^{|\sigma|}; (\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma, r): \mathtt{com}\},$$

are indistinguishable.

The following result is due to M. Naor [29].

Fact 3. *The existence of one-way functions implies the existence of secure commitment schemes.*

2.3. *Noninteractive Zero-Knowledge Proofs*

The concept of a noninteractive zero-knowledge (NIZK) proof has been put forward in [7] and further elaborated in [6]. They showed that, under the quadratic residuosity assumption, if a random reference string readable by both the prover and the verifier is available, it is possible for the prover to give any polynomial number of NIZK proofs.

The property of zero-knowledge is defined by means of a simulator which generates pairs of reference strings and proofs which cannot be told apart by any efficient nonuniform algorithm. In the definition of [6], the simulator was defined as a machine which first gets theorems to be proved and then starts the computation. For our construction we need a different kind of simulation called *on-line* simulation that has been introduced by [16]. An on-line simulator is a pair of efficient algorithms $S = (S_1, S_2)$. S_1 on input 1^n outputs a string σ of length n along with some information $\mathtt{aux}$ about σ. S_2 then receives theorems and, using $\mathtt{aux}$ and σ, computes proofs which are indistinguishable from real ones.

The proof system of [6] and [18] have an on-line simulator. In [16] it is proved that, if one-way functions exist, any NIZK proof system with efficient prover can be transformed into NIZK proof system with on-line simulator.

3. NIZKPK with Preprocessing

In this section we give the formal definition of the concept of NIZKPK with preprocessing. We start by defining the concept of a polynomial-time relation.

Definition 5. A relation is a subset of $\{0, 1\}^* \times \{0, 1\}^*$. A *polynomial-time relation* $\mathcal{P}$ is a relation such that:

1. There exists a constant a, called the *expansion constant* of $\mathcal{P}$, such that if $(x, w) \in \mathcal{P}$, then $|w| \leq |x|^a$.
2. It is possible to check in time polynomial in $|x|$ whether $(x, w) \in \mathcal{P}$.

We say that $(x, w) \in \mathcal{P}_n$ if $(x, w) \in \mathcal{P}$ and $|x| \leq n$. For each polynomial-time relation $\mathcal{P}$ the language

$$L_{\mathcal{P}} = \{x: \exists w \text{ for which } \mathcal{P}(x, w) \text{ holds}\}$$

belongs to NP. Conversely, every NP language naturally defines a polynomial-time relation.

Following [25], we model our prover and verifier as interactive Turing machines.

Definition 6. An *Interactive Turing Machine* (ITM) is a probabilistic Turing machine running in polynomial time with five tapes: a read-only *input* tape, a read/write work tape, a write-only *communication* tape, a read-only *communication* tape, and a write-only *output* tape.

We say that two ITMs A and B constitute a *pair of ITMs* if they share the input tape and the two communications tapes; that is one's write-only (read-only) communication tape is the other's read-only (write-only) communication.

The computation of a pair of ITMs (A, B) proceeds as follows. A and B take turns in becoming active with A becoming active first. When a machine is active it reads from its read-only communication tape, computes and writes a message on its write-only communication tape. That is, the ith message exchanged is a function of the public input, the private input, the coin tosses, and the previous messages.

Let (A, B) be a pair of ITMs. By the writing $\mathtt{U}_{\mathrm{A,B}}(x)$, we denote the probability space that assigns to each triplet (R, Trans, α) the probability that R is B's random coin tosses and that Trans and α are, respectively, the transcript of the conversation between A and B and the output of A, when x is the public input and B uses R as random coin tosses. Instead, writing $(\mathrm{A} \leftrightarrow \mathrm{B})(x)$ denotes the probability space that assigns to each pair (α, β) the probability that string α is written on A's output tape and string β on B's output tape, after an interaction where A and B receive as input x.

To define the notion of proof of knowledge, we need the concept of an extractor. We say that an *extractor* is a probabilistic Turing machine that can access an ITM A as an oracle; i.e., it can feed the ITM with an input, run an interactive protocol with it, and rewind the machine to a specific state, but cannot read A's private and work tapes. If A is an ITM and `Ext` is an extractor, by writing $(\mathrm{A} \leftrightarrow \mathtt{Ext})(x)$ we denote the probability space that assigns to each pair (α, β) the probability that string α is written on A's output tape and that string β is on `Ext`'s output tape, after `Ext` has computed with A as an oracle on input x.

We are now ready to introduce the concept of NIZKPK with preprocessing.

Definition 7. A pair $(\mathtt{P}, \mathtt{V})$, where $\mathtt{P} = (\mathtt{P}_1, \mathtt{P}_2)$ and $\mathtt{V} = (\mathtt{V}_1, \mathtt{V}_2)$, is a *preprocessing pair* if $\mathtt{P}_1$ and $\mathtt{V}_1$ constitutes a pair of ITMs and $\mathtt{P}_2$ is a probabilistic Turing machine running in polynomial time and $\mathtt{V}_2$ is a deterministic Turing machine running in polynomial time.

Definition 8. Let $(\mathtt{P}, \mathtt{V})$ be a preprocessing pair. We say that $(\mathtt{P}, \mathtt{V})$ constitutes an *NIZKPK with preprocessing* for the polynomial-time relation $\mathcal{P}$ if the following three conditions are satisfied:

1. **Completeness.** For all constant $d > 0$, for all $(x, w) \in \mathcal{P}$, and for all sufficiently large n,

$$\mathbf{Pr}((\alpha, \beta) \leftarrow (\mathtt{P}_1 \leftrightarrow \mathtt{V}_1)(1^n); (x, \Pi) \leftarrow \mathtt{P}_2(\alpha, x, w): \mathtt{V}_2(\beta, x, \Pi) = 1) \geq 1 - n^{-d}.$$

2. **Validity.** There exists an extractor algorithm $\mathtt{Ext} = (\mathtt{Ext}_1, \mathtt{Ext}_2)$ such that, for all pairs $\mathtt{Adv} = (\mathtt{Adv}_1, \mathtt{Adv}_2)$ of efficient algorithms, for all a, and for all sufficiently large n,

$$\begin{aligned}\mathbf{Pr}((\alpha, \beta) \leftarrow (\mathtt{Adv}_1 \leftrightarrow \mathtt{Ext}_1)(1^n); (x, \Pi) \leftarrow \mathtt{Adv}_2(\alpha);\\ w \leftarrow \mathtt{Ext}_2(\beta, x, \Pi): (x, w) \in \mathcal{P}) > p_n \cdot (1 - n^{-a}),\end{aligned}$$

where p_n denotes the probability

$$p_n = \mathbf{Pr}((\alpha, \beta) \leftarrow (\mathtt{Adv}_1 \leftrightarrow \mathtt{V}_1)(1^n); (x, \Pi) \leftarrow \mathtt{Adv}_2(\alpha): \mathtt{V}_2(\beta, x, \Pi) = 1).$$

3. **Zero knowledge.** For each pair $\tilde{\mathtt{V}} = (\tilde{\mathtt{V}}_1, \tilde{\mathtt{V}}_2)$ of ITMs, there exists an efficient algorithm M such that, for all $x_1, x_2, \ldots \in L_{\mathcal{P}}$, for all efficient nonuniform algorithms D, for all constants d, and sufficiently large n,

$$\begin{aligned}|\,&\mathbf{Pr}(y \leftarrow M(1^n, x_1, x_2, \ldots): D_{|^n}(y) = 1)\\ &- \mathbf{Pr}(y \leftarrow \mathtt{View}_{\tilde{\mathtt{V}}}(n, x_1, x_2, \ldots): D_{|^n}(y) = 1)| < n^{-d},\end{aligned}$$

where

$$\begin{aligned}\mathtt{View}_{\tilde{\mathtt{V}}}(n, x_1, x_2, \ldots) = \{&(\mathrm{R}, \mathrm{Trans}, \alpha) \leftarrow \mathtt{U}_{\mathtt{P}_1, \tilde{\mathtt{V}}_1}(1^n);\\ &(x_1, \Pi_1) \leftarrow \mathtt{P}_2(\alpha, x_1, w_1);\\ &(x_2, \Pi_2) \leftarrow \mathtt{P}_2(\alpha, x_2, w_2);\\ &\quad\vdots \quad\vdots \quad\vdots\\ &\qquad\qquad : (\mathrm{R}, \mathrm{Trans}, \Pi_1, \Pi_2, \ldots)\}.\end{aligned}$$

We say that an interactive pair $(\mathtt{P}, \mathtt{V})$ is a noninteractive proof system of knowledge with preprocessing if completeness and validity are satisfied.

In the definition of validity we let $\mathtt{Adv}_1$ and $\mathtt{Adv}_2$ communicate through the string α computed as output by $\mathtt{Adv}_1$. In our proof, without loss of generality, we let α be $\mathtt{Adv}_1$'s view of the interaction with $\mathtt{V}_1$ (including $\mathtt{Adv}_1$'s coin tosses).

Our definition handles any number of formulae of arbitrary size in completeness, validity, and zero-knowledge. That is, every true theorem can be proven, no matter how long. Of course longer theorems will have longer proofs and thus the verifier will have

more time to verify the proof. Similarly, we guarantee the extractor to succeed in extracting a witness, with a probability which is close (up to a factor negligible with respect to the length of the preprocessing stage) to the probability with which the verifier is convinced. The zero-knowledge property holds with respect to nonuniform distinguishing algorithms whose running time and program size are bounded by a polynomial in the length of the preprocessing. This means that if a theorem (and thus its proof) is exponentially long in the length of the preprocessing stage, the distinguishing algorithm can only compare view and output of the simulator for a polynomially long prefix.

4. NIZKPK with Preprocessing for all Polynomial-Time Relations

In this section we present an NIZKPK with preprocessing $(\mathtt{P}, \mathtt{V})$ for a polynomial-time relation $\mathcal{P}$; we denote by a the expansion constant of $\mathcal{P}$. In our proof system, after a preprocessing of size n, any polynomial number of ZKPK can be given for instances x of length at most n. However, as we shall see later, this restriction can be extended using a standard technique.

The Preprocessing Stage. The preprocessing stage is executed only once and does not depend on the choice of the theorems that will be proved.

A formal description of the protocol for the preprocessing stage is found in Fig. 1. In the description of the protocol for the preprocessing stage we denote by (C, D) a secure commitment scheme and we let COMMIT be the polynomial-time relation defined by

$$((\sigma, \mathtt{com}), (s, \mathtt{dec})) \in \text{COMMIT iff } D(\mathtt{com}, \mathtt{dec}, \sigma) = s.$$

The following fact holds.

Fact 4 [19]. *If one-way functions exist, then all polynomial-time relations have a constant-round zero-knowledge proof system of knowledge.*

In what follows, we let (Pro, Ver) be the constant-round zero-knowledge proof system of knowledge for the polynomial-time relation COMMIT.

In several points of our protocol, one of the two parties needs to check the validity of the message received from the other either explicitly (e.g., at step 5) or implicitly (e.g., at step 6 when executing the proof system (Pro, Ver)). We assume that, if the check is not passed, the party performing the check stops.

We remark that the preprocessing stage only takes a constant number of rounds.

The Proof Stage. Let $\mathbf{F} = \{f_s\}$ be a a-pseudorandom collection of functions and let the pair (A, B) be an NIZK proof system of membership for the following NP language:

$$\begin{aligned}\text{CHECK}_{\mathcal{P}} = \{&(\sigma_p, \mathtt{com}, x, \gamma): \text{ exist } \mathtt{dec} \text{ and} \\ &s \text{ s.t. } D(\mathtt{com}, \mathtt{dec}, \sigma_p) = s \text{ and } (x, \gamma \oplus f_s(x)) \in \mathcal{P}\}.\end{aligned}$$

In what follows we drop the subscript $\mathcal{P}$ and write simply CHECK. A formal description of $\mathtt{P}_2$'s and $\mathtt{V}_2$'s programs are found in Fig. 2.

Protocol for the preprocessing stage

Input: 1^n (security parameter).

Phase I

1. $\mathtt{P}_1$: Randomly choose and send $\sigma_V \in \{0,1\}^n$ to $\mathtt{V}_1$.
2. $\mathtt{V}_1$: Randomly choose $\rho \in \{0,1\}^n$.
   Compute $(\eta, \delta) \leftarrow C(\sigma_V, \rho)$.
   Send η to $\mathtt{P}_1$.
3. $\mathtt{P}_1$: Randomly choose and send $\tau \in \{0,1\}^n$ to $\mathtt{V}_1$.
4. $\mathtt{V}_1$: Compute $\sigma = \rho \oplus \tau$.
   Send σ and ρ to $\mathtt{P}_1$.
5. $\mathtt{P}_1$: Verify that $\sigma = \rho \oplus \tau$.
6. $\mathtt{V}_1 \leftrightarrow \mathtt{P}_1$: $(\mathtt{V}_1, \mathtt{P}_1)$ execute proof system (Pro, Ver), to prove in zero-knowledge that $\mathtt{V}_1$ knows δ such that $((\sigma_V, \eta)(\rho, \delta)) \in$ COMMIT.

Phase II

7. $\mathtt{V}_1$: Randomly choose $\sigma_P \in \{0,1\}^n$ and send it to $\mathtt{P}_1$.
8. $\mathtt{P}_1$: Randomly choose $s \in \{0,1\}^n$ and compute $(\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma_P, s)$.
   Send com to $\mathtt{V}_1$.
9. $\mathtt{P}_1 \leftrightarrow \mathtt{V}_1$: $(\mathtt{P}_1, \mathtt{V}_1)$ execute proof system (Pro, Ver), to prove in zero-knowledge that $\mathtt{P}_1$ knows dec and s such that $D(\mathtt{com}, \mathtt{dec}, \sigma_P) = s$.

Output for $\mathtt{P}_1$: $\sigma, \sigma_P, (\mathtt{com}, \mathtt{dec}), s$.

Output for $\mathtt{V}_1$: $\sigma, \sigma_P, \mathtt{com}$.

Fig. 1. The protocol for the preprocessing stage.

$\mathtt{P}_2$'s program

Input from preprocessing: $(\sigma, \sigma_P, (\mathtt{com}, \mathtt{dec}), s)$ such that $D(\mathtt{com}, \mathtt{dec}, \sigma_P) = s$.

Input to $\mathtt{P}_2$: $(x_1, w_1), (x_2, w_2) \ldots \in \mathcal{P}_{|\sigma|}$.

For $i = 1, 2, \ldots$

1. Compute $f_s(x_i)$ and $\gamma_i = w_i \oplus f_s(x_i)$.
2. Run A on input σ_P, com, dec, s, x_i, γ_i using σ as the reference string. Let $\mathtt{Pf}_i$ be the output. ($\mathtt{Pf}_i$ is a "proof" that $(\sigma_P, \mathtt{com}, x_i, \gamma_i) \in$ CHECK.)
3. Send $((\sigma_P, \mathtt{com}, x_i, \gamma_i), \mathtt{Pf}_i)$.

$\mathtt{V}_2$'s program

Input from preprocessing: $\sigma, \sigma_P, \mathtt{com}$.

Input from $\mathtt{P}_2$: $((\sigma'_P, \mathtt{com}', x_i, \gamma_i), \mathtt{Pf}_i)$, for $i = 1, 2, \ldots$

For $i = 1, 2, \ldots$

- Verify that $\sigma_P = \sigma'_P$ and com=com$'$.
- Execute B's program on input $((\sigma_P, \mathtt{com}, x_i, \gamma_i), \mathtt{Pf}_i)$ and the reference string σ.

If all checks are successfully passed accept, otherwise reject.

Fig. 2. The programs for the proof stage.

The Choice of the NIZK Proof of Membership. Our protocol is quite flexible in the choice of the NIZK proof system which is used in the proof stage. If other NIZK proof systems are available, the above protocol could be modified to accommodate them. For example, if in the preprocessing step prover and verifier establish n oblivious channels, then they can use the protocol by [28] for NIZK proofs. There are few differences if we use either of the two NIZK proof systems. First, the minimal assumption on which such systems have been shown to exist: one-way trapdoor permutation in one case and oblivious transfer in the other. Second, transferability. Using the reference string the proofs are transferable to others who trust the correctness of the construction of the string *com*. This is a fundamental property for protocols for electronic cash (see [14]). Another possibility is the use of the NIZK proof systems with preprocessing of [13]. This has the advantage of being based on the sole existence of one-way functions, but has a drawback that only a small theorem (its size depends on the preprocessing stage) can be noninteractively proved.

The proof that the above protocol is indeed an NIZKPK is divided into two parts. First, we prove that it meets the completeness and soundness requirements and then, in the next section, we prove the more subtle property of zero-knowledge.

Theorem 1. *The pair* (P, V) *is a noninteractive proof system of knowledge with preprocessing.*

Proof. The *completeness* of (P, V) follows from the completeness of the interactive proof system of knowledge (Pro, Ver) and the completeness of (A, B).

To prove the *validity* property, we have to exhibit an extractor $(\mathtt{Ext}_1, \mathtt{Ext}_2)$ that, interacting with an adversary $\mathtt{Adv} = (\mathtt{Adv}_1, \mathtt{Adv}_2)$, is able to compute a witness for the theorem that is being proved.

Algorithm $\mathtt{Ext}_1$ behaves exactly like verifier $\mathtt{V}_1$ except for step 9 when $\mathtt{Adv}_1$ is supposed to give an interactive proof of knowledge. Here, $\mathtt{Ext}_1$ uses the extractor for the interactive proof system of knowledge (Pro, Ver) to obtain dec and s from $\mathtt{Adv}_1$. The probability that $\mathtt{Ext}_1$ succeeds in computing dec and s is equal (up to a negligible factor) to the probability that Ver accepts. This information is then passed onto $\mathtt{Ext}_2$ together with σ, σ_p, and dec (they correspond to string β of Definition 8). Then $\mathtt{Ext}_2$ receives from $\mathtt{Adv}_2$ a quadruple $(\sigma_p, \mathtt{com}, x, \gamma)$ along with a proof Pf. $\mathtt{Ext}_2$ runs algorithm B on input $(\sigma_p, \mathtt{com}, x, \gamma)$, the proof Pf, and reference string σ. If B accepts, it outputs w computed as $w = f_s(x) \oplus \gamma$.

We distinguish two cases depending on whether the quadruple $(\sigma_p, \mathtt{com}, x, \gamma)$ does or does not belong to CHECK. If $(\sigma_p, \mathtt{com}, x, \gamma)$ belongs to CHECK, then clearly string w computed by the extractor is a witness for x. On the other hand, if $(\sigma_p, \mathtt{com}, x, \gamma)$ does not belongs to CHECK, then $(x, w) \notin \mathcal{P}$ and we say that the extractor has failed. The probability that the extractor fails to obtain a witness w for x is equal up to a negligible factor to the probability that $\mathtt{Adv}_1$ succesfully completes the preprocessing stage and that $\mathtt{Adv}_2$ makes $\mathtt{V}_2$ accept a quadruple not belonging to CHECK. Indeed the only difference between the view received by $\mathtt{Adv}_2$ when $\mathtt{Adv}_1$ interacts with the extractor and the view received by $\mathtt{Adv}_2$ when $\mathtt{Adv}_1$ interacts with the verifier $\mathtt{V}_1$ is in the portion of transcript regarding step 9. However, by the properties of the extractor of (Pro, Ver), these two

views are indistinguishable, whence we can conclude that the two probabilities are equal up to a negligible factor.

We now show that the probability that $(\sigma_p, \mathtt{com}, x, \gamma) \notin$ CHECK and $\mathtt{V}_2$ accepts the proof provided by $\mathtt{Adv}_2$ is negligible thus proving that the extractor fails with negligible probability and concluding the proof of the validity property. To this aim, we consider the following mental experiment.

A Mental Experiment. We have Adv interact with an efficient machine G defined as follows. During Phase I, G follows $\mathtt{V}_1$'s program but instead of sending string ρ he has committed to, he sends a random string ρ'. Then, at step 6 he uses the simulator of proof system (Pro, Ver) to produce a transcript of a proof that convinces $\mathtt{Adv}_1$ that ρ' is the string he has committed to. In Phase II, G behaves exactly as $\mathtt{Ext}_1$ (i.e., it computes s and dec using the extractor of (Pro, Ver)). Then $\mathtt{Adv}_2$ receives from $\mathtt{Adv}_1$ the transcript of the preprocessing stage (along with $\mathtt{Adv}_1$'s own random coin tosses), chooses strings x, γ and sends $(\sigma_p, \mathtt{com}, x, \gamma)$ along with a proof string Pf. G checks if $(\sigma_p, \mathtt{com}, x, \gamma) \notin$ CHECK (remember that G knows s and dec) and $\mathtt{V}_2$ accepts. If this is the case G outputs 1 otherwise it outputs 0.

We now compare the view received by $\mathtt{Adv}_2$ from $\mathtt{Adv}_1$ when interacting with the extractor and the view received by $\mathtt{Adv}_2$ in the experiment. The differences between the two views are the following:

- In the view of $\mathtt{Adv}_1$ interacting with the extractor, string ρ is the string committed to at step 2, while in the experiment ρ' is chosen independently from the commitment of step 2.
- In the view of $\mathtt{Adv}_1$ interacting with the extractor, the proof at step 6 is obtained by running Ver's program while in the experiment it is produced by the simulator.

Therefore, by the indistinguishability of the string commitments and by the zero-knowledgeness of the proof system (Pro, Ver), the two views are indistinguishable and this implies that the probability that the extractor fails and the probability that G outputs 1 are equal up to a negligible factor.

Next, we argue that the probability that G outputs 1 is equal (up to a negligible factor) to the probability that $\mathtt{Adv}_2$ makes $\mathtt{V}_2$ accept a quadruple not in CHECK when the reference string is chosen at random. Indeed we have the following two observations:

- As ρ' is chosen at random, reference string σ with respect to which $\mathtt{Adv}_2$ has to produce a noninteractive proof is random.
- The view that $\mathtt{Adv}_2$ receives in the second experiment is indistinguishable from the view that $\mathtt{Adv}_2$ receives from $\mathtt{Adv}_1$ interacting with $\mathtt{V}_1$. This can be shown in a way similar to what was done before.

Whence we conclude that the probability that the extractor fails to obtain a witness and the verifier accepts is equal up to a negligible factor to the probability that $\mathtt{Adv}_2$ makes $\mathtt{V}_2$ accept a quadruple not in CHECK when the reference string is chosen at random. The latter probability is, by the soundness of (A, B), negligible thus proving the validity property. □

4.1. (P, V) *Is Zero-Knowledge*

In this section we prove the following theorem.

Theorem 2. *The pair* (P, V) *is an NIZKPK with preprocessing.*

Intuitively (P, V) is zero-knowledge because all the verifier sees is just commitments and zero-knowledge proofs on those commitments which would not give additional information.

Let c be a constant. Without loss of generality, we consider for each pair of ITM $\mathrm{V}' = (\mathrm{V}'_1, \mathrm{V}'_2)$, the family of random variables $\mathtt{View}_{\mathrm{V}'} = \{\mathtt{View}_{\mathrm{V}'}(n, \vec{x}) | \vec{x} \in L_{\mathcal{P}} \text{ and } |\vec{x}| = n^c\}$. $\mathtt{View}_{\mathrm{V}'}(n, \vec{x})$ represents the view of what a verifier $(\mathrm{V}'_1, \mathrm{V}'_2)$ sees when 1^n is given as input in the preprocessing stage and $\vec{x}$ are the inputs of the noninteractive phase. $\mathtt{View}_{\mathrm{V}'}$ consists of the following components:

- R, the random tape of V'_1.
- the n-bit strings $\sigma, \sigma_{\mathrm{V}}, \sigma_{\mathrm{P}}, \eta, \tau, \rho$.
- $\mathtt{Log}_1$, the transcript of the interactive ZKPK that ρ is the string committed to.
- com, the commitment of a random n-bit string s (unknown to V'_1) computed using σ_{P}.
- $\mathtt{Log}_2$, the transcript of the interactive ZKPK that P knows the string committed to by com.
- $\Gamma = (\gamma_1, \ldots, \gamma_{n^c})$, where $\gamma_i = f_s(x_i) \oplus w_i$ and w_i is a witness for x_i.
- $\mathtt{PF} = (\mathtt{Pf}_1, \ldots, \mathtt{Pf}_{n^c})$, where $\mathtt{Pf}_i$ is an NIZK proof of "correctness" of γ_i.

For the sake of compact notation, in what follows we drop the subscript V'_1. We present a simulator M such that the family of random variables $\mathcal{M} = \{M(1^n, \vec{x})\colon \vec{x} \in L_{\mathcal{P}}\}$ cannot be distinguished from View by any efficient nonuniform algorithm.

Here is a sketch of the simulator M. A formal description can be found in Fig. 3. M receives as inputs 1^n and an n^c-tuple $\vec{x}$ and we let $S = (S_1, S_2)$ be the on-line simulator for the NIZK proof system (A, B). M starts by running S_1 and obtains a string σ along with some information aux that allows S_2 to simulate convincing noninteractive "proofs" of any statement. Then M starts an interactive protocol with V'_1 at the end of which both V'_1 and M would have agreed on σ as the string to use for the noninteractive proofs. This is accomplished by running the protocol twice in the following way. The first run is intended for M to learn the random string ρ to which V'_1 has committed and, once ρ is known, M rewinds V'_1 and chooses his string so to obtain the string σ. Then M executes step 9 of the preprocessing stage just as P would do.

In the simulation of the proof stage, M does not compute γ_i as $f_s(x_i) \oplus w_i$ (he does not know w_i!) but instead assigns to γ_i a truly random string. Then M runs the simulator S_2 for the NIZK on input σ, aux, and the quadruple $(\sigma_{\mathrm{P}}, \mathrm{com}, x_i, \gamma_i)$ to obtain $\mathtt{Pf}_i$.

First, clearly M runs in expected polynomial time. Indeed all computations can be performed efficiently and the probability that at step 2.4 M has to restart is negligible (this follows from the properties of the secure committment scheme).

We now compare the output of M on input $(1^n, \vec{x})$ with an element chosen from $\mathtt{View}(n, \vec{x})$. We see that some of the components have the same distribution; for example,

The program of M

Input: 1^n and a tuple $\vec{x} \in L_{\mathcal{P}}$.

0. Set $(\sigma, \mathtt{aux}) \leftarrow S_1(1^n)$.
   Randomly choose random bits R for V_1'.
1. **First Trial.**
   - 1.1 Randomly choose $\sigma_v \in \{0,1\}^n$ and send it to V_1'.
   - 1.2 Receive commitment η from V_1'.
   - 1.3 Randomly choose $\tau \in \{0,1\}^n$ and send it to V_1'.
   - 1.4 Receive $\hat{\sigma}$ and ρ from V_1'.
   - 1.5 Check that $\hat{\sigma}$ has been properly computed. If not, **Output:** $(R, \sigma_v, \eta, \tau, \hat{\sigma}, \rho)$.
   - 1.6 Execute interactively with V_1' Ver's program to verify that V_1' knows δ such that $((\sigma_v, \eta), (\rho, \delta)) \in$ COMMIT and let $\mathtt{Log}_1$ be the transcript of the interaction.
     If Ver does not accept, then **Output:** $(R, \sigma_v, \eta, \tau, \hat{\sigma}, \rho, \mathtt{Log}_1)$.
2. **Obtaining** σ.
   - 2.1 Rewind V_1' to state after step 1.2.
   - 2.2 Send string $\hat{\tau} = \sigma \oplus \rho$ to V_1'.
   - 2.3 Receive σ' and ρ' from V_1'.
     Check that σ' has been properly computed. If not **Output:** $(R, \sigma_v, \eta, \hat{\tau}, \sigma', \rho')$.
   - 2.4 Execute interactively with V_1' Ver's program to verify that V_1' knows δ' such that $((\sigma_v, \eta'), (\rho', \delta')) \in$ COMMIT and let $\mathtt{Log}_1$ be the transcript of the interaction.
     If Ver does not accept, then **Output:** $(R, \sigma_v, \eta, \hat{\tau}, \sigma', \rho', \mathtt{Log}_1)$.
     If Ver accepts but $\sigma \neq \sigma'$, then goto 0.
3. Receive σ_p from V_1'.
   Randomly choose $s \in \{0,1\}^n$, compute $(\mathtt{com}, \mathtt{dec}) \leftarrow C(\sigma_p, s)$, and send com to V_1'.
4. Prove in zero-knowledge to V_1' knowledge of s and dec such that $D(\mathtt{com}, \mathtt{dec}, \sigma_p) = s$.
   Let $\mathtt{Log}_2$ be a transcript of the protocol.
5. For $i = 1, \ldots, n^c$
   - 5.1 Randomly choose $\gamma_i \in \{0,1\}^n$ and compute $\mathtt{Pf}_i \leftarrow S_2(\sigma_p, \mathtt{com}, x_i, \gamma_i, \sigma, \mathtt{aux})$.
6. Set $\Gamma = (\gamma_1, \ldots, \gamma_{n^c})$ and $\mathtt{PF} = (\mathtt{Pf}_1, \ldots, \mathtt{Pf}_{n^c})$.

Output: $(R, \sigma_v, \eta, \hat{\tau}, \sigma, \rho', \mathtt{Log}_1, \sigma_p, \mathtt{com}, \mathtt{Log}_2, \Gamma, \mathtt{PF})$.

Fig. 3. The program of the simulator M.

σ_v and σ_p are random strings just as in the view of V', com is the commitment of a random n-bit string, and $\mathtt{Log}_1$ is the transcript of a "conversation" between P_1 and V_1'. Nonetheless, the two distributions are very different. First, the γ_i's are not computed correctly (that is, by xoring a witness of x_i's with $f_s(x_i)$); moreover, string σ is not truly random, as it should be. Though we shall prove that the two distributions cannot be told apart by a polynomially bounded machine.

We assume for the sake of contradiction that there exists a nonuniform efficient algorithm $\mathcal{H} = \{H_{1^n}\}$ that violates the zero-knowledge property.

Consider the following family of random variables $\mathcal{Z} = \{\mathcal{Z}(n, \vec{x}) | \vec{x} \in L_{\mathcal{P}} \text{ and } |\vec{x}| = n^c\}$. For each $\vec{x} \in L_{\mathcal{P}}$, the random variable $\mathcal{Z}(n, \vec{x})$ consists of the output of algorithm $Z_{\vec{x}}$ on input n. $Z_{\vec{x}}$ has "wired-in" $w_1, \ldots, w_{n^c}$ such that, for $i = 1, \ldots, n^c$, $(x_i, w_i) \in \mathcal{P}$ and executes the same instructions of M on input $(1^n, \vec{x})$ with one exception; at step 5.1, γ_i is computed as $\gamma_i = f_s(x_i) \oplus w_i$. We have the following lemma.

Lemma 2. *The families of random variables $\mathcal{Z}$ and* `View` *are indistinguishable.*

Proof. Suppose that $\mathcal{Z}$ and `View` are distinguished by an efficient nonuniform algorithm $\mathcal{D} = \{D_{1^n, \vec{x}}\}$. That is, there exists a constant a such that, for an infinite set $\mathcal{I}$ of tuples $\vec{x}$,

$$|\mathbf{Pr}_{\mathtt{View}}(n, \vec{x}) - \mathbf{Pr}_{\mathcal{Z}}(n, \vec{x})| > n^{-a},$$

where

$$\mathbf{Pr}_{\mathtt{View}}(\vec{x}) = \mathbf{Pr}(\alpha \leftarrow \mathtt{View}(n, \vec{x}) \colon D_{\vec{x}}(\alpha) = 1)$$

and

$$\mathbf{Pr}_{\mathcal{Z}}(\vec{x}) = \mathbf{Pr}(\alpha \leftarrow \mathcal{Z}(n, \vec{x}) \colon D_{\vec{x}}(\alpha) = 1).$$

We now describe a nonuniform algorithm $\mathcal{C} = \{C_{1^n, \vec{X}} | \vec{X} \in \text{CHECK}\}$ which distinguishes the output of S from the view of B in the proof system (A, B). This would contradict the zero-knowledgeness of (A, B) and prove the lemma.

Let $X = ((x_1, \sigma_p, \mathtt{com}, \alpha_1), \ldots, (x_{n^c}, \sigma_p, \mathtt{com}, \alpha_{n^c})) \in \text{CHECK}$ with $\vec{x} = (x_1, \ldots, x_{n^c}) \in \mathcal{I}$. $C_{1^n, \vec{X}}$ has "wired-in" witness w_i for x_i, $i = 1, \ldots, n^c$, and a string $\tilde{s}$ such that `com` is a commitment of $\tilde{s}$ computed using string σ_p. $C_{1^n, \vec{X}}$ receives as input a pair $(\tilde{\sigma}, \Pi)$, with $|\tilde{\sigma}| = n$, which is either the view of B on input X when an n-bit random string is shared with A, or the output of S on input $\vec{X}$ and 1^n. $C_{1^n, \vec{X}}$ executes M's program with the following exceptions: at step 0, $C_{1^n, \vec{X}}$ sets $\sigma = \tilde{\sigma}$; at step 3, the string s is set equal to $\tilde{s}$; at step 5.1, it sets $\gamma_i = \alpha_i$; and at step 6, it sets $\mathtt{PF} = \Pi$. Then $C_{1^n, \vec{X}}$ feeds $D_{1^n, \vec{x}}$ with $\text{Test} = (R, \sigma_v, \eta, \tau, \sigma, \rho, \mathtt{Log}_1, \sigma_p, \mathtt{com}, \mathtt{Log}_2, \Gamma, \Pi)$. Now, if $(\tilde{\sigma}, \Pi)$ is distributed according to $S(1^n, \vec{X})$, then Test is distributed according to $\mathcal{Z}(n, x)$; on the other hand, if $(\tilde{\sigma}, \Pi)$ is distributed according to B's view, then Test has the same distribution of `View`. Since $\mathcal{D}$ distinguishes between the two random variables, $\mathcal{C}$ is able to distinguish the output of $S(1^n, \vec{X})$ from B's view. Contradiction. □

We now prove that $\mathcal{Z}$ and $\mathcal{M}$ are indistinguishable. By Fact 1, this will prove that $\mathcal{M}$ and `View` are indistinguishable and conclude the proof of the zero-knowledgeness of $(\mathtt{P}, \mathtt{V})$.

The main difference between the two random variables is in the nature of Γ. In fact, in $\mathcal{Z}$ the ith component of Γ consists of the xor of $f_s(x_i)$ and a witness for x_i, where s is the string committed to by `com`, while in the output of M it is just a random string. However, as (C, D) is a secure commitment scheme and $\mathbf{F}$ is a pseudorandom family of functions the two distributions are indistinguishable.

We use the following lemma.

Lemma 3. *Let c be a constant. The families of random variables $\mathcal{U} = \{U_{1^n,\vec{x}} \mid |\vec{x}| = n^c \text{ and } \vec{x} \in L_{\mathcal{P}}\}$ and $\mathcal{V} = \{V_{1^n,\vec{x}} \mid |\vec{x}| = n^c \text{ and } \vec{x} \in L_{\mathcal{P}}\}$ defined as*

$$U_{1^n,\vec{x}} = \{\sigma, s \leftarrow \{0,1\}^n; (\mathrm{com}, \mathrm{dec}) \leftarrow C(\sigma, s);\ \eta_1 = f_s(x_1);\ \vdots\ \vdots\ \vdots\ \eta_{n^c} = f_s(x_{n^c}):\ (\sigma, \mathrm{com}, \eta_1, \ldots, \eta_{n^c})\}$$

and

$$V_{1^n,\vec{x}} = \{\sigma, s \leftarrow \{0,1\}^n; (\mathrm{com}, \mathrm{dec}) \leftarrow C(\sigma, s);\ \eta_1 \leftarrow \{0,1\}^n;\ \vdots\ \vdots\ \vdots\ \eta_{n^c} \leftarrow \{0,1\}^n:\ (\sigma, \mathrm{com}, \eta_1, \ldots, \eta_{n^c})\}$$

are indistinguishable.

Proof. Consider the family of probability distributions $\mathcal{T} = \{T_{1^n,\vec{x}} \mid |\vec{x}| = n^c \text{ and } \vec{x} \in L_{\mathcal{P}}\}$ where

$$T_{1^n,\vec{x}} = \{\sigma, s, \tilde{s} \leftarrow \{0,1\}^n; (\mathrm{com}, \mathrm{dec}) \leftarrow C(\sigma, \tilde{s});\ \eta_1 = f_s(x_1);\ \vdots\ \vdots\ \eta_{n^c} = f_s(x_{n^c}):\ (\sigma, \mathrm{com}, \eta_1, \ldots, \eta_{n^c})\}.$$

1. $\mathcal{T}$ is indistinguishable from $\mathcal{V}$.

   Suppose they are not and let C be an efficient nonuniform algorithm that distinguishes $\mathcal{T}$ and $\mathcal{V}$. That is, there exist a constant d and an infinite set $\mathcal{I}$ of $(n, \vec{x})$ for which the probability that $C_{1^n,\vec{x}}$ outputs 1 on input a string chosen according to $T_{1^n,\vec{x}}$ differs by more than $|\vec{x}|^{-d}$ from the probability that it outputs 1 on input a string chosen according to $V_{1^n,\vec{x}}$.

   Then we construct an algorithm $D = \{D_{1^n}\}$ that distinguishes a randomly chosen function from $\mathbf{F}$ from a completely random function.

   Let n be such that there exists a $\vec{x}$ for which $(n, \vec{x}) \in \mathcal{I}$. D_{1^n} has $\vec{x}$ "wired-in" and proceeds as follows. It randomly chooses $\sigma, s \in \{0,1\}^n$ and computes a commitment com of s using σ; then it asks for the value of the function at $x_1, \ldots, x_{n^c}$ receiving answers $\eta_1, \ldots, \eta_{n^c}$. Finally, it gives as output the result of the computation of $C_{1^n,\vec{x}}$ on input TEST $= (\sigma, \mathrm{com}, \eta_1, \ldots, \eta_{n^c})$. Now, if the function D_{1^n} has access to is randomly chosen from $\mathbf{F}$, then TEST is distributed according to $T_{1^n,\vec{x}}$. On the other hand, if the function D_{1^n} has access to is a truly random one, then TEST is distributed according to $V_{1^n,\vec{x}}$. Thus, D is a distinguishing algorithm for $\mathbf{F}$ which contradicts the pseudorandomness of $\mathbf{F}$.

2. $\mathcal{T}$ is indistinguishable from $\mathcal{U}$.

   Again, suppose they are not and let C be an efficient nonuniform algorithm such that, for a constant d and an infinite set $\mathcal{I}$ of $(n, \vec{x})$, the probability that $C_{1^n,\vec{x}}$ outputs 1 on input a string chosen according to $T_{1^n,\vec{x}}$ differs by more than $|\vec{x}|^{-d}$ from the probability that it outputs 1 on input a string chosen according to $U_{1^n,\vec{x}}$.

   Consider now the following nonuniform algorithm $G_{\sigma,\tilde{s}}$ that receives as input

a commitment com, computed using σ, of a string of length $n = |\hat{s}| = |\sigma|$ and outputs a guess to whether com is the commitment of $\hat{s}$. The algorithm knows $\vec{x} \in \mathcal{I}$ of size n^c. Such a tuple exists for infinitely many values of n. $G_{\sigma,\hat{s}}$ computes $\eta_1 = f_{\hat{s}}(x_1), \ldots, \eta_{n^c} = f_{\hat{s}}(x_{n^c})$. Then it gives in output the result of the computation of $C_{1^n,\vec{x}}$ on input TEST $= (\sigma, \text{com}, \eta_1, \ldots, \eta_{n^c})$.

If com is a commitment of $\hat{\sigma}$, then TEST is distributed according to $U_{1^n,\vec{x}}$. While if com is a commitment of a random string of length n, then TEST is distributed according to $T_{1^n,\vec{x}}$. Thus, algorithm B contradicts Lemma 1.

Using the transitivity of the indistinguishability relation, the lemma follows. □

We have thus proved Theorem 2. The following theorem also holds.

Theorem 3. *If one-way functions exist and all NP languages have an NIZK proof system of membership with efficient prover, then all polynomial-time relations have an NIZKPK with constant-round preprocessing.*

Proof. The theorem is proved using Theorems 1 and 2 and the facts that one-way functions imply the existence of pseudorandom collections of functions (Fact 3), secure commitment schemes (Fact 3), and constant-round zero-knowledge proof system of knowledge for all polynomial-time relations (Fact 4). □

Remark. As we have observed, in the proof system (P, V) all instances have length bounded by the length n of the preprocessing stage (actually, by the length of the string s to which the prover committed in the preprocessing). The extension to instances of length polynomial in n can be achieved by breaking (see [6]) a long instance into a polynomial number of instances of length at most n and proving each of them individually.

Applications. In our proof system it is not necessary that the verifier is the same in the preprocessing and in the proof stage. Instead, it is enough that the verifier of the proof stage trusts the correctness of the preprocessing stage. This suggests replacing the interactive preprocessing stage (that now has to be performed with each potential verifier) with a preprocessing stage that is performed with a trusted center. In a sense, this preprocessing is some sort of a registration of the prover and is conceptually similar to the interaction needed in establishing a new key in a public-key cryptosystem. This makes our results of potential applicability in the traditional public-key setting.

Acknowledgments

We thank an anonymous referee for carefully reading the manuscript. The second author would like to thank Michael O. Rabin for his constant support and encouragment.

References

[1] J. Bar-Ilan and D. Beaver, Non-Cryptographic Fault-Tolerant Computation in a Constant Number of Rounds of Interaction, *Proceedings of the 8th PODC*, 1989, pp. 201–209.

[2] D. Beaver, S. Micali, and P. Rogaway, The Round Complexity of Secure Protocols, *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, 1990, pp. 503–513.

[3] M. Bellare and O. Goldreich, On Defining Proofs of Knowledge, *Proceedings of CRYPTO '92*, Lecture Notes in Computer Science, vol. 740, Springer-Verlag, Berlin, pp. 390–420.

[4] M. Bellare, S. Micali, and R. Ostrowsky, Perfect Zero Knowledge in Constant Rounds, *Proceedings of the 22nd Annual Symposium on the Theory of Computing*, 1990, pp. 482–493.

[5] M. Bellare and M. Yung, Certifying Cryptographic Tools: the Case of Trapdoor Permutations, *Proceedings of CRYPTO '92*, Lecture Notes in Computer Science, vol. 740, Springer-Verlag, Berlin, pp. 442–460.

[6] M. Blum, A. De Santis, S. Micali, and G. Persiano, Non-Interactive Zero Knowledge, *SIAM Journal on Computing*, vol. 20, December 1991, pp. 1084–1118.

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

[8] G. Brassard, Constant-Round Perfect Zero Knowledge Made Easy (and Efficient), manuscript available from the author, 1990.

[9] G. Brassard, C. Crépeau, S. Laplante, and C. Léger, Computational Convincing Proofs of Knowledge, *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science (STACS 91)*, Lecture Notes in Computer Science, vol. 480, Springer Verlag, Berlin, pp. 251–262.

[10] G. Brassard, C. Crépeau, and M. Yung, Constant-Round Perfect Zero-Knowledge Computationally Convincing Protocols, *Theoretical Computer Science*, vol. 84, 1991, pp. 23–52.

[11] A. De Santis, G. Di Crescenzo, and G. Persiano, Statistical Zero-Knowledge Arguments and Public-Key Cryptography, *Information and Computation*, vol. 121, No. 1, August 1995, pp. 23–40.

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

[13] A. De Santis, S. Micali, and G. Persiano, Non-Interactive Zero-Knowledge Proof-Systems with Preprocessing, in *Advances in Cryptology—CRYPTO 88*, ed. S. Goldwasser, Lecture Notes in Computer Science, vol. 403, Springer-Verlag, Berlin, pp. 269–282.

[14] A. De Santis and G. Persiano, Communication Efficient Zero-Knowledge Proofs of Knowledge (with Applications to Electronic Cash), *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science (STACS '92)*, ed. A. Finkel and M. Jantzen, Lecture Notes in Computer Science, vol. 577, Springer-Verlag, Berlin, pp. 449–460.

[15] A. De Santis and G. Persiano, Zero-Knowledge Proofs of Knowledge Without Interaction, *Proceedings of the 33rd Symposium on Foundations of Computer Science*, 1992, pp. 427–437.

[16] A. De Santis and M. Yung, Cryptographic Applications of the Non-Interactive Metaproof and Many-Prover Systems, in *Advances in Cryptology—CRYPTO '90*, eds. A. J. Menezes and S. A. Vanstone, Lecture Notes in Computer Science, vol. 537, Springer-Verlag, Berlin, pp. 366–377.

[17] U. Feige, A. Fiat, and A. Shamir, Zero-Knowledge Proofs of Identity, *Journal of Cryptology*, vol. 1, 1988, pp. 77–94. (Preliminary version in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, 1987, pp. 210–217.)

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

[19] U. Feige and A. Shamir, Zero-Knowledge Proof of Knowledge in Two Rounds, *Advances in Cryptology—CRYPTO 89*, Lecture Notes in Computer Science, vol. 435, Springer-Verlag, Berlin, pp. 526–544.

[20] P. Feldman and S. Micali, Optimal Algorithms for Byzantine Agreement, *Proceedings of the 20th Annual Symposium on the Theory of Computing*, 1988, pp. 148–161.

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

[22] O. Goldreich, S. Goldwasser, and S. Micali, How To Construct Random Functions, *Journal of the Association for Computing Machinery*, vol. 33, no. 4, 1986, pp. 792–807.

[23] O. Goldreich and H. Krawczyk, On the Composition of Zero-Knowledge Proof Systems, *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, 1986, pp. 174–187.

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

[25] S. Goldwasser, S. Micali, and C. Rackoff, The Knowledge Complexity of Interactive Proof-Systems, *SIAM Journal on Computing*, vol. 18, no. 1, February 1989, pp. 281–308.

[26] J. Håstad, Pseudorandom Generators Under Uniform Assumptions, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990, pp. 395–404.

[27] R. Impagliazzo, L. Levin, and M. Luby, Pseudo-Random Generation from One-Way Functions, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 1989, pp. 12–24.

[28] J. Kilian, S. Micali, and R. Ostrowsky, Minimum-Resource Zero-Knowledge Proofs, *Proceedings of the 30th IEEE Symposium on Foundation of Computer Science*, 1989, pp. 474–479.

[29] M. Naor, Bit Commitment Using Pseudorandomness, *Journal of Cryptology*, vol. 4, no. 2, pp. 151–158.

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