SIAM J. COMPUT.
Vol. 29, No. 1, pp. 1–28

MULTIPLE NONINTERACTIVE ZERO KNOWLEDGE PROOFS UNDER GENERAL ASSUMPTIONS*

URIEL FEIGE†, DROR LAPIDOT†, AND ADI SHAMIR†

Abstract. In this paper we show how to construct noninteractive zero knowledge proofs for any NP statement under general (rather than number theoretic) assumptions, and how to enable polynomially many provers to give polynomially many such proofs based on a single random string. Our constructions can be used in cryptographic applications in which the prover is restricted to polynomial time.

Key words. Hamiltonian cycle, witness indistinguishability

AMS subject classifications. 94A60, 68Q15

PII. S0097539792230010

1. Introduction.

1.1. Background. Blum, Feldman, and Micali [BFM] suggested the intriguing concept of *noninteractive zero knowledge* (NIZK) *proofs*, aimed at eliminating the interaction between prover and verifier in zero knowledge interactive proof systems [GMR]. The prover P writes down a zero knowledge proof that an input x belongs to a prespecified language L, and any verifier V can check the validity of this written proof against a universal publicly available random string (such as the RAND string of one million random digits), called the *common reference string*. NIZK has become an important primitive for cryptographic protocols, with applications such as signature schemes [BG] and encryption schemes secure against chosen ciphertext attack [NY].

NIZK proof systems for any NP statement were constructed in [BFM] and [DMP87], under specific number theoretic assumptions (namely, that it is difficult to distinguish products of two primes from products of three primes, or that it is difficult to decide quadratic residuosity modulo products of two primes). The main disadvantage of these *bounded* NIZK proofs is that the prover can prove only one statement of size bounded by the length of the common reference string: if polynomially many proofs are given using the same reference string, the zero knowledge property breaks down.[1] In [BDMP] it was finally shown how a single prover can give polynomially many proofs using the same reference string, but the scheme is still based on a specific number theoretic assumption: deciding quadratic residuosity (modulo composite integers whose factorization is not known) is computationally hard. Moreover, their scheme cannot support polynomially many provers.

A variation of the NIZK model was suggested by De Santis, Micali, and Persiano [DMP88]. In their *noninteractive with preprocessing* model, the verifier and prover create a common reference string (which need not look like a random string) during an interactive preliminary stage. Based on this common reference string (CRS), the prover can then prove any single NP statement (of bounded length). Unlike

*Received by the editors April 13, 1992; accepted for publication (in revised form) June 18, 1997; published electronically September 14, 1999. A preliminary version of this paper appeared in *Proc. 31st Symposium on Foundations of Computer Science*, 1990, pp. 308–317.
http://www.siam.org/journals/sicomp/29-1/23001.html
†Dept. of Applied Math. and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel (feige@wisdom.weizmann.ac.il).
[1]The method suggested in [BFM] for overcoming this difficulty was found to be flawed.

the original NIZK model, in the noninteractive with preprocessing model, the proof should look convincing only to the verifier who takes part in the initial preprocessing stage, which makes this model unsuitable for applications such as signature schemes. [DMP88] showed an implementation of this idea based on the general assumption that one-way functions exist. Under the stronger cryptographic assumption that *oblivious transfer* protocols exist, [KMO] shows how after an initial preprocessing stage, the prover can noninteractively prove polynomially many NP statements, but again the proof is verifiable only by its original recipient. [BeMi] show how to do oblivious transfer without interaction (and hence NIZK proofs, by [KMO]) in a model where the verifier is first given a special public key.

1.2. Our results. In this paper we answer the two major open questions associated with the concept of NIZK, as presented by Blum, De Santis, Micali, and Persiano [BDMP]: how to construct NIZK proof systems for any NP statement under general (rather than number theoretic) assumptions and how to enable polynomially many provers to share the same random reference string in giving such proofs.

As a preliminary result leading to our solution of the first open question, we construct (under the assumption that one-way functions exist) a very simple zero knowledge *noninteractive with preprocessing* proof for Hamiltonicity, whose efficiency is comparable with the efficiency of the interactive proofs presented by Blum [Blum] and Goldreich, Micali, and Wigderson [GMW]. In contrast, all the previously known constructions of *NIZK with preprocessing* proofs [DMP88] are more complex and less efficient than their interactive counterparts. Then, under the assumption that one-way permutations exist, we show that if the prover and verifier initially share a common random string (which we call a common reference string), then the initial preprocessing stage of our protocol can be discarded, yielding a NIZK proof for any NP statement in the original noninteractive model of Blum, Feldman, and Micali. This noninteractive protocol is the only known implementation which relies on general computational assumptions and is conceptually simpler than the number-theoretic protocols[2] presented by Blum, De Santis, Feldman, Micali, and Persiano. Under the stronger assumption that certified trapdoor permutations exist (i.e., that the prover can demonstrate that his chosen function is indeed a permutation without revealing its trapdoor), our NIZK protocol can be carried out by probabilistic polynomial time provers and thus can be used in cryptographic applications which require NIZK protocols.

As a solution to the second open problem, we show how to transform any *bounded* NIZK proof system for an *NP complete* language into a *general* NIZK proof system in which polynomially many independent provers can share the same reference string and use it to prove polynomially many statements of polynomial length. The transformation is based on the general assumption that one-way functions exist.

Independent of our work, De Santis and Yung [DY] also show how to transform bounded NIZK proof systems into general ones, although their transformation produces noninteractive proofs which are longer than ours.

In order to use NIZK proof systems in cryptographic applications it is often necessary to extend the security conditions imposed on NIZKs to withstand adaptive attacks (see [BG], [NY]). The original definitions of NIZK proof systems assume that the statements to be proved are chosen independently of the CRS, whereas the

[2]This is based on the assumption that deciding quadratic residuosity (modulo composite integers whose factorization is not known) is computationally hard.

adaptive setting allows for the possibility that statements to be proven are chosen after the CRS is given, and may depend upon the CRS. In the last section of this paper we show that our constructions also satisfy the more stringent conditions imposed by the adaptive setting.

1.3. Definitions. $A(x)$ denotes the random variable describing the output of a probabilistic algorithm A on input x. Informally, $\nu(n)$ denotes functions vanishing faster than the inverse of any polynomial; i.e., $f(n) \leq \nu(n)$ is shorthand notation for

$$\forall d\ \exists N \text{ s.t. } \forall n > N \ \ 0 \leq f(n) < \frac{1}{n^d}$$

and $f(n) \geq 1 - \nu(n)$ is shorthand notation for

$$\forall d\ \exists N \text{ s.t. } \forall n > N \ \ 1 \geq f(n) > 1 - \frac{1}{n^d}.$$

DEFINITION 1.1. *A binary relation R is* polynomially bounded *if it is decidable in polynomial time and also there is a polynomial p such that for all $(x, w) \in R$ it is the case that $|w| \leq p(|x|)$. For any such relation and any x we let $w(x) = \{w : (x, w) \in R\}$ denote the witness set of x. We let $L_R = \{x \mid \exists w \ s.t. \ (x, w) \in R\}$.*

R will denote a polynomially bounded relation in what follows. Note that if R is polynomially bounded, then L_R is in NP.

A NIZK proof system for NP allows a prover P to use a publicly available random string (the *CRS*) in order to prove in writing (without interaction) any NP theorem, without revealing any knowledge besides the validity of the theorem. Any polynomial time verifier V with access to the CRS can verify the validity of the proof.

The input of P is a triple (x, ω, σ) where $(x, \omega) \in R$, R is a polynomially testable relation, and σ is the CRS. Its output $P(x, \omega, \sigma)$ is a noninteractive proof (based on the witness ω, with respect to the CRS σ) that $x \in L_R$. The initial input of V (before receiving P's proof) is the pair (x, σ). Let $|x| = n$ denote the size of the common input x. Let $V(x, \sigma, P(x, \omega, \sigma))$ denote the output of the verifier V, after receiving the noninteractive proof $P(x, \omega, \sigma)$. This output may be either "accept" or "reject." For brevity of notation, we sometimes do not explicitly specify x and σ as inputs to V, when x and σ are clear from the context.

As in the case of interactive proofs, noninteractive proofs satisfy the *completeness* and the *soundness* conditions: if $x \in L_R$ then P's proof causes V to accept, and if $x \notin L_R$ the probability that V accepts P's output is negligible. The following is the formal definition of a noninteractive proof.

DEFINITION 1.2. *A noninteractive proof system for an NP language L_R is a pair of probabilistic algorithms (P, V) (where V is polynomial time) satisfying the following conditions.*

There exist two integers $b, c \geq 1$ such that the following hold:

(1) Completeness. $\forall (x, \omega) \in R, \ \forall \sigma$ *of length* $n^b k^c \ V(x, \sigma, P(x, \omega, \sigma)) = accept.$

(2) Soundness. *If σ is a random string, then the probability of succeeding in proving a false statement is negligible, even if the theorem is chosen by P after seeing σ.*[3] *Formally, $\forall n \geq 1$ at least $(1 - \nu(k))$ of the strings σ of length $n^b k^c$ satisfy*

$$\forall x \in (\{0,1\}^n - L_R) \qquad \forall y \quad V(x, \sigma, y) = reject.$$

[3] In nonadaptive definitions of soundness, the prover could produce a false statement based on the choice of the random string, whose proof (associated with this particular string) will be accepted. Our definition of soundness disallows this possibility.

Remark. In the definition above (and in Definition 1.3 below), k is a security parameter (known to all parties) that quantifies how "sound" a noninteractive proof must be (or quantifies the "zero knowledge" property in Definition 1.3). More formally, we assume that the value of k (represented in unary notation as 1^k) is an additional input provided to all the algorithms, but we do not make this dependence explicit in order to simplify our notation. It is often convenient (and is the practice of most other papers on zero knowledge) to choose $k = n$ and require the desired properties of noninteractive proof systems to hold for "large enough n."

As in the case of interactive proofs (see [GMR]), the formal definition of NIZK proofs involves the notion of a probabilistic expected polynomial time simulator M, whose input is just the common input x of P and V (without an appropriate witness ω), and its output $M(x)$ consists of two strings: one of them simulates the common (random) reference string, and the other one simulates the real noninteractive proof (sent by P). Informally, a noninteractive proof is zero knowledge if such a pair of strings is computationally indistinguishable from what V sees in the actual noninteractive proof which is $(\sigma, P(x, \omega, \sigma))$. The following is the formal definition of NIZK.

DEFINITION 1.3. *A noninteractive proof system for an NP-language L_R is zero knowledge if there exists a probabilistic machine M (called a simulator) such that for any $x \in L_R$, $M(x)$ terminates in expected polynomial time and the two ensembles $\{(\sigma, P(x, \omega, \sigma))\}$[4] and $\{M(x)\}$[5] are computationally indistinguishable on L_R by any nonuniform polynomial time distinguisher $D = \{D_l\}_{l \geq 1}$:*

$$\exists b, c \geq 1 \quad \exists M \; \forall D = \{D_l\}_{l \geq 1} \; \forall (x, \omega) \in R \; \forall d \geq 1 \quad \exists K \geq 1 \; \forall k > \; \mathrm{Max}(K, |x|),$$

$$|Pr(D_k(M(x)) = 1) - Pr(D_k(\sigma, P(x, \omega, \sigma)) = 1)| < \frac{1}{k^d},$$

where the probability space is taken over the random choices of $\sigma \in_R \{0,1\}^{|x|^b k^c}$ and over the random tapes of P and M.

Remark. A nonuniform algorithm $D = \{D_l\}_{l \geq 1}$ is an algorithm that for every input length l gets an auxiliary input ("advice") of length polynomial in l. These algorithms are equivalent to polynomial size circuit families.

In cryptographic applications we would like to use efficient protocols for both P and V. The term *NIZK proof systems with efficient provers* denotes NIZK proof systems in which the truthful prover (in the *completeness* condition) is probabilistic polynomial time (in n, the length of the input x, and in k, the security parameter). "Cheating" provers (in the *soundness* condition) are never required to be computationally bounded.

2. NIZKs under general cryptographic assumptions.

2.1. A NIZK proof with preprocessing. In this subsection we present a protocol for a different model which is called a NIZK proof with preprocessing. This is not the final protocol: it is presented here just as an intermediate step in order to facilitate the understanding of the final NIZK proof system.

Consider a prover who wants to prove the Hamiltonicity of an arbitrary graph G with n nodes. We assume that the prover and the verifier are allowed to execute a

[4] Every element in this ensemble is uniquely determined by the choice of $\sigma \in_R \{0,1\}^{|x|^b k^c}$ and the prover's random tape.

[5] Here, every element is uniquely determined by the value of the random tape of the probabilistic machine M.

preliminary interactive stage which is independent of G (i.e., at this stage they know that in the noninteractive stage the prover will prove the Hamiltonicity of an n node graph, but they don't know which graph it will be). Only after the termination of this interactive stage they get G and execute the noninteractive move in which the prover sends a written message to the verifier in order to convince him in zero knowledge that G is Hamiltonian. The verifier is not allowed to ask the prover any questions and should be convinced just by reading this message.

The basic step. Let H be a randomly chosen directed Hamiltonian cycle on n nodes. We call such a graph a *good graph*. Note that this is a cyclic list without any starting point. The adjacency matrix of H is a matrix in which each row and each column contains exactly one entry that is set to 1, and the locations of the entries that are 1 define a permutation with a single cycle. Let S be the adjacency matrix of a good graph in which each entry is replaced by a string that hides it (for example, by the hard bit construction of [GL] or by a probabilistic encryption), so that a polynomially bounded observer cannot determine the locations of the ones.

Assume now that S is given to both P and V, and that P wants to prove to V that some n-nodes graph G is Hamiltonian. Since P is infinitely powerful, he can recover (to himself) the hidden 0/1 values of S which define the Hamiltonian cycle H in the good graph and determine a permutation π on the nodes of G such that H is a Hamiltonian cycle in $\pi(G)$ (i.e., $H \subseteq \pi(G)$).

In order to convince V that G is Hamiltonian, P just sends it a message which consists of the permutation π and the decryptions of all entries $S_{i,j}$ in the good matrix S for which $(i,j) \notin E(\pi(G))$. V accepts the proof iff all the revealed entries are 0. The proof system is *complete* since P can carry it out and V will accept it when G is indeed Hamiltonian. The proof system is *sound* because V's acceptance implies that all the n ones that remain unrevealed in S (and define a Hamiltonian cycle) correspond to edges of $\pi(G)$, which means that $\pi(G)$ contains a Hamiltonian cycle and thus the common input graph G is Hamiltonian.

We informally argue that the proof is *zero knowledge*. All that the verifier receives is a random permutation π and a collection of random encryptions (the entries of S) along with the decryption of those $S_{i,j}$ for which $(i,j) \notin E(\pi(G))$. All these decrypted values are 0. Since the original good matrix H (which defines the 0/1 values of S) is randomly chosen with uniform distribution (among the $(n-1)!$ possibilities), then so is $\pi \in_R Sym(n)$ (the permutation group on $[1 \dots n]$). This follows from the fact that any two different Hamiltonian cycles H and H' determine two disjoint sets A_H and $A_{H'}$ of n permutations, where each permutation in A_H ($A_{H'}$) maps the Hamiltonian cycle of G onto H (H').[6] Therefore, for any permutation in $Sym(n)$, the probability that V receives it is $\frac{1}{n!}$. Therefore, this protocol can be easily simulated: the simulator (whose input is just the graph G) chooses a random permutation $\pi \in_R Sym(n)$ with uniform distribution, chooses random 0/1 values for all entries $S'_{i,j}$ for which $(i,j) \in E(\pi(G))$, fixes all the others to be 0, and produces random encryptions for all entries of S'. Then the simulator outputs π and the decryptions of all entries $S'_{i,j}$ for which $(i,j) \notin E(\pi(G))$. Since π is uniformally chosen in $Sym(n)$ and all the above encryptions are randomized, this simulation is computationally indistinguishable from the real proof.

Based on the above, we construct a NIZK proof system with preprocessing (regardless of whether P is efficient or not): In the preliminary interactive stage P sequentially sends k (= security parameter) good random matrices $S_1, S_2, \dots, S_k$ to

[6] There may be several Hamiltonian cycles in G, but we concentrate on any one of them.

V and receives k random bits $b_1, b_2, \ldots, b_k$ from V. In the noninteractive move it reveals all entries of those S_i's for which $b_i = 0$ and executes the above basic step for those S_i for which $b_i = 1$. If all the S_i with $b_i = 0$ are good (i.e., hidden adjacency matrices of Hamiltonian cycles), then V can conclude with high probability that at least one of the other S_i is also good, in which case G is guaranteed to be Hamiltonian.

In order to compare this protocol with Blum's protocol for Hamiltonicity [Blum], let us recall that in the first move of Blum's scheme P randomly permutes G and sends V the encrypted adjacency matrix of this isomorphic copy. V then sends a random bit to P and according to that bit P either reveals all the entries in the matrix and the permutation or reveals only the entries which correspond to the edges of the Hamiltonian cycle. Our protocol resembles Blum's protocol, with one major difference: in Blum's protocol all the moves depend on G, while in our protocol only the last move depends on G. As a result, Blum's protocol cannot be split into a preprocessing stage and a noninteractive proof as we did in our protocol.

Remark. This particular NIZK proof with preprocessing can be extended for *directly* proving (without reduction to the Hamiltonian problem) a variety of graph theoretic problems which are satisfied by a single minimal (or maximal) graph (under isomorphism). This family includes clique, graph coloring, graph partition into k-cliques, three-dimensional matching, etc. We don't know how to extend our proof technique directly to other NP-complete problems.

2.2. A NIZK proof with a common reference string. In this subsection we show that under the assumption that (strong) one-way permutations exist, if the prover and the verifier initially share a random string (or CRS) σ, then the initial preprocessing stage of the protocol described in section 2.1 can be discarded, yielding a NIZK proof for any NP statement in the noninteractive model of Blum, Feldman, and Micali.

DEFINITION 2.1. *A permutation f is a (strong)* one-way function *if for any x ($|x| = n$), $f(x)$ (which is also of length n) can be computed in polynomial time, but for any nonuniform polynomial time algorithm A,*

$$Prob(A(y) = f^{-1}(y)) \leq \nu(n),$$

where the probability is computed over the random choices of $y \in_R \{0,1\}^n$.

We remark that strong one-way permutations exist if "weak" one-way permutations (i.e., the probability of not inverting y is nonnegligible) exist [Yao], [GILVZ].

In our NIZK construction we want to guarantee the hardness of inverting the one-way permutation f at the single bit level. This idea is captured in the notion of *a hard-core predicate* of a one-way function. A hard-core predicate of a function f is a predicate $B : \{0,1\}^* \longrightarrow \{0,1\}$, which is efficiently computable but such that given only $f(x)$ it is hard to guess $B(x)$ with a probability significantly better than $1/2$.

DEFINITION 2.2. *We call the predicate $B : \{0,1\}^* \longrightarrow \{0,1\}$ a* hard-core predicate *of the function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ if the following conditions are satisfied:*

1. *B is computable in deterministic polynomial time.*
2. *For every nonuniform polynomial time probabilistic algorithm A, for every integer $c > 0$, and for every large enough n,*

$$Prob\{A(f(x)) = B(x)\} < \frac{1}{2} + \frac{1}{n^c},$$

where the probability is taken over the coin tosses of A and for x uniformly chosen in $\{0,1\}^n$.

The idea of hard-core predicates was introduced and first implemented (based on a specific one-way permutation) by [BM]. [Yao] presented a general transformation of any (strong) one-way function into one which has a hard-core predicate, but the transformation was impractical. Goldreich and Levin [GL] provided an alternative transformation which was much more efficient. Our NIZK construction uses such a hard-core predicate in order to extract hard-to-guess bits from any given one-way function.

2.2.1. Informal description. Assume that P and V possess a CRS σ, and P wants to send V a NIZK proof based on σ (rather than on an interactive preprocessing stage) that an arbitrary n node graph G is Hamiltonian. Basically, the proof technique consists of two stages.

1. Interpretation of the CRS as an encoding of a string of "secret" bits—a hidden random string (HRS). Only the unbounded prover can initially read the hidden bits, but he can later selectively reveal the value of some of the hidden bits to the polynomially bounded verifier without revealing any information on the other hidden bits.
2. Interpretation of the HRS as a sequence of $n \times n$ matrices in such a way that at least one of them represents a good graph with overwhelming probability. For each matrix the prover P is allowed to do one of two things: to prove to V that the matrix is not good by revealing all its entries, or to use it as if it is a good matrix in the Hamiltonicity testing protocol of section 2.1. Note that the prover can claim that a bad matrix is good, but he cannot successfully claim that a good matrix is bad, and thus he will be forced to use all the good matrices in the protocol. If the input G is non-Hamiltonian, P will fail to convince V except in the extremely unlikely case in which all the matrices defined by the HRS are bad.

The first stage is implemented by considering the CRS as a concatenation of polynomially many blocks $u_1, u_2, \ldots$, where each block contains k (= security parameter) random bits. We define a corresponding intermediate random string (IRS) by concatenating the values of $w_1, w_2, \ldots$, where each w_i is equal to $f^{-1}(u_i)$. The ith bit s_i in the HRS is a hard-core bit defined by the ith block u_i in the CRS. By revealing s_i we mean that P sends to V the bit s_i along with w_i from the IRS. By checking s_i we mean that V checks that $f(w_i) = u_i$ and $B(w_i) = s_i$. Note that $f^{-1}(u_i)$ exists and is uniquely defined by our assumption that f is a one-way permutation, and thus even the unbounded prover cannot "flip" the value of this bit without being caught by the verifier.

The second stage is implemented by interpreting the HRS as a sequence of $n \times n$ 0/1 matrices which with overwhelming probability contain at least one good matrix. Notice that if we naively interpret each block of n^2 consecutive bits from the HRS as an $n \times n$ 0/1 matrix, then the probability that even one of these polynomially many matrices is good is exponentially low. Therefore, we have to encode our matrices in a more complicated way.

Assume that the HRS can be partitioned into equal size segments $z_1, z_2, z_3, \ldots$, each of which defines (in some way) an $n^2 \times n^2$ matrix B_i of zeros and ones, such that for each i and each (j, l) the probability that the (j, l)th entry in B_i is 1 equals $\frac{1}{n^3}$ (i.e., $Pr\{B_i(j, l) = 1\} = \frac{1}{n^3}$). Therefore, for any segment z_i the expected number of 1-entries in the corresponding matrix B_i is n, and we will later prove that every B_i contains, with nonnegligible probability, exactly n rows and n columns, each of which contains a single 1-entry and the $n \times n$ permutation matrix induced by these

rows and columns is Hamiltonian. Therefore, if the length of the HRS is large enough (polynomial in n and k), then, with high probability, at least one of its segments defines a good matrix.

All we have to show is how to transform a given random string into a sequence of matrices, each of which has the property of B_i. Consider the given random string as a concatenation of polynomially many consecutive blocks of m bits where $m = \log(n^3)$ (w.l.o.g. we can assume that it is an integer). We interpret a block as 1 if all its m bits are 1, and 0 otherwise. Therefore, for every m-bit block, the probability of being interpreted as 1 is $\frac{1}{n^3}$ and thus we can pack each consecutive segment of $n^4 m$ random bits into the desired $n^2 \times n^2$ 0/1 matrix B_i discussed above.

Informally, the proof technique is the following: for each matrix B_i, the prover must either prove that it contains no good $n \times n$ submatrix or execute the basic step (described in section 2.1) on a good matrix derived from B_i. In order to construct a good matrix from a given matrix $B = B_i$ and to prove that the input n-node graph G is Hamiltonian, P executes the following.

1. If the number of ones in B is different from n or there exists a row or a column which contains at least two ones, then P proves this fact by revealing all entries in B. Otherwise (i.e., B contains an $n \times n$ permutation submatrix), P reveals all entries in the $n^2 - n$ rows and the $n^2 - n$ columns which contain only zeros, and removes them from B. If the resulting $n \times n$ Boolean matrix does not represent a permutation with a single cycle (i.e., it is not an adjacency matrix of some Hamiltonian cycle), then P proves this fact by revealing all entries of the remaining $n \times n$ matrix.
2. Otherwise (i.e., the remaining $n \times n$ matrix forms an adjacency matrix of some Hamiltonian cycle), P must execute the basic protocol (described in the previous section) on the resulting $n \times n$ good matrix.

In order to formally describe the scheme and prove its correctness, we introduce some notation and definitions.

2.2.2. Notation and definitions. Let

$$\sigma = r_1 \dots r_{poly(k,n)} \qquad r_i \in_R \{0,1\}$$

$$= u_1 \dots u_{poly(k,n)} \qquad u_i \in_R \{0,1\}^k$$

be the CRS, shared by P and V. Let f be a (strong) one-way permutation, and let s_i be the hard bit that corresponds to the k-bit string u_i (with respect to f).

We associate with the CRS an IRS defined by $(f^{-1}(u_1), \dots, f^{-1}(u_{poly(k,n)}))$. According to the definition of a hard bit (see [GL]) each s_i (of the HRS) is polynomially computable from the corresponding $f^{-1}(u_i)$ (of the IRS) which acts as its "witness."

For each $i \geq 1$ let

$$b_i = \bigwedge_{j=1}^{m} s_{(i-1)m+j},$$

where $m = \log(n^3)$.

Let B_i be an $n^2 \times n^2$ matrix which is defined as follows: $B_i(j,l) = b_{(i-1)n^4+(j-1)n^2+l}$ for every $1 \leq i, j, l$.

DEFINITION 2.3. *We say that B_i is a proper matrix if it contains exactly n ones and each column and row contains at most a single one.*

If B_i is a proper matrix let N_i be the $n \times n$ matrix obtained by removing all the $n^2 - n$ columns and $n^2 - n$ rows which contain only zeros. Otherwise N_i is undefined.

DEFINITION 2.4. *A Boolean $n \times n$ matrix is called good if it is the adjacency matrix of a graph which consists of a single directed cycle passing through all the n vertices.*

With some abuse of notation, we also call the large $n^2 \times n^2$ matrix B_i good if it is proper, and if the (unique) $n \times n$ submatrix N_i it defines is good.

2.2.3. The scheme. Assume that P and V have a CRS with $2n^7k^2m$ bits and fix some one-way permutation f.

P's protocol. For each $1 \le i \le kn^3$ do the following:

1. If the matrix B_i is not good then reveal all its entries.
2. Otherwise (B_i contains a good $n \times n$ submatrix N_i), reveal and remove (the entries of) all the $n^2 - n$ columns and all the $n^2 - n$ rows which contain only zeros, and execute the noninteractive stage (of the protocol described in section 2.1) on the remaining good matrix N_i.

V's protocol. For each $1 \le i \le kn^3$ do the following:

1. If P reveals all entries of B_i then check that the revealed bits are correct and that the matrix they define is not a good matrix.
2. Otherwise, P should reveal $n^2 - n$ columns and $n^2 - n$ rows: check that the revealed bits are correct and that all entries they define in these rows and columns are zeros; in addition, P should execute in this case the basic protocol (described in section 2.1) on the remaining hidden $n \times n$ matrix: check that this proof is carried out correctly.

Accept the proof iff each of these kn^3 checks is successful.

2.3. Correctness.

2.3.1. Completeness. If the input graph is indeed Hamiltonian, then the prover can execute correctly the proof with respect to each one of the good matrices (if they exist). In each $n^2 \times n^2$ matrix that does not yield a good submatrix, P is just required to reveal the entire matrix and V will accept its proof as valid. As a result, an honest prover never fails.

2.3.2. Soundness.

CLAIM 2.5. *For every i, the probability that exactly n entries of B_i are 1 is at least $\frac{1}{4\sqrt{n}}$.*

Proof. The bits of the HRS are unbiased and independent, and for each j the probability that $b_j = 1$ is $1/n^3$. Therefore, the probability that B_i has exactly n ones is

$$\binom{n^4}{n}\left(\frac{1}{n^3}\right)^n\left(1-\frac{1}{n^3}\right)^{n^4-n} > \frac{1}{4\sqrt{n}}$$

for all sufficiently large n. ∎

The size of B_i is $n^2 \times n^2$ and the 0/1 value of each entry is determined independently of the others since these bits are determined by nonoverlapping blocks from the random CRS. Assume now that B_i has exactly n ones. Its entries are independent of each other, and thus the locations of the n ones in the n^2 rows of B_i can be viewed as the result of placing n balls in n^2 buckets. The same argument holds also for the columns. By the birthday paradox, with constant probability, the n ones are located

in n distinct rows and n distinct columns, and thus with constant probability a matrix B_i with n ones is proper.

The number of permutations in $Sym(n)$ which consist of a single cycle (of length n) is $(n-1)!$. Therefore, the probability that N_i is a good matrix, given that it is a permutation matrix, is n^{-1}.

We conclude that, for every i, the probability that B_i is good is at least $\geq dn^{-3/2}$, where d is a constant. Thus if the length of the CRS is $\Omega(n^{13/2}k^2m)$ bits, then with probability $(1-e^{-nk})$ at least one of the B_i's yields a good matrix. Any such matrix will expose a cheating P, since it cannot prove that B_i is bad and cannot use it in the basic step, and these are its only two options.

Remark. If $\log(n^3)$ is not an integer, set $m = \lceil \log(n^3) \rceil$ and choose B_i as a $\lceil bn^2 \rceil \times n^2$ matrix where $b = \frac{2^m}{n^3}$ $(1 < b < 2)$.

2.3.3. Zero knowledge. We construct a random polynomial time simulator M which generates a "random string" and a "proof" of Hamiltonicity which are polynomially indistinguishable (by nonuniform distinguishers) from those generated by a real execution of the protocol.

Consider the task of M. Compared with the real prover P, it is handicapped in two respects: It cannot invert one-way functions (and thus cannot expose the HRS defined by the given CRS), and it may not know a Hamiltonian cycle in G. We solve one problem at a time and use the transitivity of indistinguishability to prove that the two solutions can be combined into one. First we construct a random polynomial time algorithm P' that cannot invert one-way functions, but does have access to the Hamiltonian cycle of G. P' uniformally generates a CRS in such a way that it can recover its associated HRS. Since the original CRS (appended to a proof of the real prover) is also uniformally chosen, these two strings are identically distributed. Next we construct a probabilistic expected polynomial time simulator M whose input is the Hamiltonian graph G (without its Hamiltonian cycle) and whose output is polynomially indistinguishable from that of P'. Therefore, these constructions imply that our scheme is zero knowledge.

Let P' be the random polynomial time algorithm whose input consists of the graph G along with its Hamiltonian cycle. The instructions of the original P and the definition of the one-way permutation f (which is fixed in the original proof) are parts of P'. P' executes the real protocol with the following exception: instead of using the given random CRS to compute its associated IRS, it chooses a truly random IRS and computes its associated CRS by applying the one-way permutation f (in the forward direction) to each consecutive block of k bits in the IRS. Namely, for each segment v_i ($v_i \in_R \{0,1\}^k$) in the IRS, P' evaluates $f(v_i)$ and set $f(v_i)$ to be the ith k-segment of the new CRS. The output of P' consists of his CRS accompanied by a noninteractive proof for the Hamiltonicity of G (which is performed exactly according to the instructions of the real P). Since both the original CRS (which is used by P) and that produced by P' are uniformally distributed random strings, and P' and P behave identically once the CRS is determined, we conclude that the output of P' is indistinguishable from the original CRS followed by the real prover's proof.

We now change P' further to get the simulator M. This simulator accepts a Hamiltonian graph G and the security parameter k as inputs but is not given any Hamiltonian cycle in G. Its output is a string σ_k of length n^7k^2m bits and a "proof" of Hamiltonicity based on this CRS. The basic idea behind the simulator is that it tries to execute the protocol and changes the CRS in an indistinguishable way whenever it encounters difficulties. To do this, it leaves unchanged all the bits of the HRS which

were part of bad matrices but changes to zero all the bits of the HRS which were part of good matrices. This would allow the simulator to successfully carry out both stages of the proof (namely, demonstrating the unsuitability of the bad matrices by revealing all their entries and revealing the required zero entries in all the presumably good remaining matrices). To change an HRS bit to zero, the simulator repeatedly tries new random IRS values until it finds one which makes the corresponding HRS bit zero and then replaces the corresponding CRS block by f applied to the new IRS value (which remains random and indistinguishable from the original value). More precisely, the simulator M performs the following steps:

1. M randomly chooses a sequence of n^7k^2m truly random bits and uses them as the IRS. Every segment in this IRS that yields a good matrix M changes all entries whose value is 1 to zeros. This is done in the following way: for each i for which N_i is a good matrix and for each j, l such that $N_i(j, l) = 1$, M executes the following trial: it replaces all bits in the IRS which give rise to this $N_i(j, l)$ by new random km bits. M repeatedly executes this trial until $N_i(j, l) = 0$ (the probability of success is $1 - \frac{1}{n^3}$).
2. M computes the CRS σ_k from the modified IRS by applying f in the forward direction and then computes the HRS from the IRS in exactly the same way as it is done by P.
3. For each i such that B_i has not been changed in the first step, M reveals all entries of B_i. For each of the other B_i's, M reveals $n^2 - n$ random rows and $n^2 - n$ random columns. Since the resulting $n \times n$ matrix contains only zeros, M can easily simulate the basic step by choosing a random permutation $\psi \in_R Sym(n)$ and revealing every $B_i(j, l)$ such that there is no edge between j and l in $\psi(G)$.

The output of M is denoted by $(\sigma_k, proof'(\sigma_k, G))$, where the first component plays the role of the CRS and the second one includes all revealed bits and permutations. Let τ_k be a string of length n^7k^2m bit which is produced by P' as a CRS and denote by $proof(\tau_k, G)$ a proof of P' based on G and τ_k.

For any nonuniform distinguisher D, let $D(x)$ denote the 0/1 output of D on input x. Let

$$\rho^D_{P',k,G} = Pr\{D((\tau_k, proof(\tau_k, G)), G) = 1\},$$

$$\rho^D_{M,k,G} = Pr\{D((\sigma_k, proof'(\sigma_k, G)), G) = 1\},$$

where the probabilities are taken over the random tapes of P' and M and thus also over the random choices of τ_k and σ_k (which are chosen by P' and M, respectively).

LEMMA 2.6. *For any polynomial Q and any positive integer t, there exists a positive integer K, such that for any Hamiltonian graph G of size n, for any nonuniform distinguisher D, and for any $k \geq \max(K, n)$,*

$$|\rho^D_{P',k,G} - \rho^D_{M,k,G}| < \frac{1}{Q(k)},$$

where the running time of D is bounded by k^t.

Proof. The proof is based on the well-known hybrid argument of [GM]. We assume the existence of some efficient distinguisher D, a polynomial Q, and an infinite subset of security parameters $\mathcal{I} \subset \mathcal{N}$ such that for every $k \in \mathcal{I}$,

$$(*) \quad |\rho^D_{P',k,G} - \rho^D_{M,k,G}| \geq \frac{1}{Q(k)}.$$

Our goal is to show how to use the existence of these entities in order to successfully predict some hard-core bit, in violation of the assumption that f is a one-way permutation. More precisely, we would like to construct a probabilistic polynomial time nonuniform algorithm C which, given as input $f(x)$ for a randomly chosen x, predicts its hard-core predicate $B(x)$ with nonnegligible probability of success. This C chooses a Hamiltonian graph G and constructs a proof of Hamiltonicity whose CRS is a mixture of an initial segment corresponding to a real proof by P' and a final segment corresponding to a simulated proof by M. (Note that both P' and M are polynomial time, and thus their behavior can be replicated by C.) It is not difficult to show that for some location of this boundary, our assumption implies that the distinguisher's probability of outputting 1 jumps nonnegligibly when the CRS boundary moves by a single block. If C embeds its input $f(x)$ at this crucial location in the CRS, it can use the success of the distinguisher to predict the value of the hard-core bit of its input. This will lead to a contradiction, and thus the probability distributions generated by P' and by M are polynomially indistinguishable.

From here on we omit the superscript D and the subscript G. Let k be an element in $\mathcal{I}$. Let $\alpha = (i_1, \ldots, i_t, \psi_1, \ldots, \psi_u)$ $(1 \le i_1 < \cdots < i_t \le n^7 km$ and for each $1 \le i \le u$ $\psi_i \in Sym(n))$ and let $\rho_{\alpha,k}$ $(\rho'_{\alpha,k})$ denote the probability that the hidden bits which are revealed by P' (respectively, M) are those which are indexed (in the HRS) by $i_1, \ldots, i_t$ and $\psi_1, \ldots, \psi_u$ are the permutations associated with good matrices chosen by P' (respectively, M). Since M and P' follow exactly the same procedure for choosing the locations of the revealed bits (M may change the *value* of some revealed bits from 1 to 0 by replacing their corresponding CRS blocks but does not try to move their *location*), and both of them choose truly random permutations to apply to G, we conclude that for any α,

$$\rho_{\alpha,k} = \rho'_{\alpha,k}.$$

Let $proof(\tau_k, G, \alpha)$ and $proof'(\sigma_k, G, \alpha)$ denote proofs of P' and M based on τ_k and σ_k, respectively, in which the locations of the revealed bits and the random permutations (in $Sym(n)$) are identical and are defined by $\alpha = (i_1, \ldots, i_t, \psi_1, \ldots, \psi_u)$. Denote by $\rho_{P',\alpha,k}$ the probability that D outputs 1 on $(\tau_k, proof(\tau_k, G, \alpha))$ (produced by P') and by $\rho_{M,\alpha,k}$ the probability that D outputs 1 on $(\sigma_k, proof'(\sigma_k, G, \alpha))$ (produced by M).

It is obvious that

$$(**) \quad \rho_{P',k} = \sum_{\alpha} \rho_{\alpha,k} \rho_{P',\alpha,k}$$

and

$$(***) \quad \rho_{M,k} = \sum_{\alpha} \rho_{\alpha,k} \rho_{M,\alpha,k}.$$

We now want to show that for any fixed choice of α, D is unable to distinguish between $(\tau_k, proof(\tau_k, G, \alpha))$ and $(\sigma_k, proof'(\sigma_k, G, \alpha))$.

Claim 2.7. *For every α,*

$$|\rho_{P',\alpha,k} - \rho_{M,\alpha,k}| < \frac{1}{Q(k)}.$$

Proof. Assume that this is not true; namely, there is α for which w.l.o.g.

$$\rho_{P',\alpha,k} - \rho_{M,\alpha,k} \ge \frac{1}{Q(k)}.$$

We now use the hybrid argument by scanning across the list of locations of revealed bits and defining a sequence of associated probabilities. For every $1 \leq j \leq n^7km$, $\rho^j_{\alpha,k}$ denotes the probability that D outputs 1 on the following (string, proof): the first $k(j-1)$ bits in the string are a prefix of a CRS generated by P' from a randomly chosen IRS, while the remaining bits in the string are generated by M from a random IRS which was modified to force zeros to appear in certain HRS revealed positions. The proof following the string is the implied combination of a prefix produced by P' and a suffix produced by M, where both parts are based on the vector α. By the hybrid argument we conclude that there is $1 \leq i \leq n^7km$ for which

$$\rho^{i+1}_{\alpha,k} - \rho^i_{\alpha,k} \geq \frac{1}{Q(k)n^7km}.$$

We'll now describe the formal construction of the probabilistic polynomial time nonuniform algorithm C_k whose auxiliary input is the graph G, including the definition of a Hamiltonian cycle, α, i, and the auxiliary input needed for D. This algorithm uses the polynomial time P', M, and D as subroutines and on input $f(x)$ (where x is randomly chosen) outputs a bit b which is the hard bit of $f(x)$ with probability $\geq \frac{1}{2} + \frac{1}{poly(k)}$. This is a contradiction to the assumption that f is oneway.

The algorithm C_k executes the following steps.

1. Run P' so that the indices of the hidden bits which are revealed and the permutations associated with the Hamiltonian matrices are according to α.
2. Run M according to the same rule.
3. Erase from the output of P' all the bits coming after the $(i-1)$th block of the CRS (call this prefix S_P).
4. Erase from the output of M the first i blocks; namely, keep the last $n^7k^2m - ik$ bits of the CRS and append the revealed bits and the permutations associated with the Hamiltonian matrices (which are based on α and thus are identical for P' and M). Call this suffix S_M.
5. Feed D with $S_P \circ f(x) \circ S_M$.
6. If $D(S_P \circ f(x) \circ S_M) = 1$, then $b = 1$ else $b = 0$.

Without loss of generality, assume that the bit defined by the ith block of P' is 1, and M changes it to 0 (these are the only changes made by M, and unchanged locations cannot possibly trigger a reaction by the distinguisher). It is easy to verify that with probability $\geq \frac{1}{2} + \frac{1}{poly(k)}$, b is the hard bit of $f(x)$ and this is a contradiction to the assumption that f is oneway. □

This claim together with $(**)$ and $(***)$ contradicts $(*)$ which completes the proof of the Lemma. □

2.4. Efficient provers.

2.4.1. The scheme. If the truthful prover is restricted to be a random polynomial time machine (namely, it has the same computational power as V) then it can not invert the one-way permutation in the protocol described in the previous section. In order to overcome this difficulty we use the notion of *families of certified trapdoor permutations*. Therefore, our assumption in this section is that such families exist. Informally, a permutation is trapdoor if its values can be computed in polynomial time and it is hard to compute its inverse, but there exists an auxiliary information (which is called the trapdoor) such that there is an algorithm which gets this trapdoor information as an auxiliary input and computes the inverse of this function in polynomial time. Such a family is certified if it is easy to verify that a given function does

belong to this family. The following is the formal definition of a family of certified trapdoor permutations.

DEFINITION 2.8. *Let I be an infinite set of indices. A set of functions $F = \bigcup_{k\geq 1} F_k$ where*

$$F_k = \{f_i : D_i \longrightarrow D_i \quad : \quad i \in I \bigcap \{0,1\}^k\}$$

is a family of certified trapdoor permutations *if for every $i \in I$, f_i is a permutation over the finite domain $D_i \subseteq \{0,1\}^k$ and the following conditions are satisfied:*

1. *There exists a random polynomial time generating algorithm G that on input k (in unary representation) generates a random pair $(i, t(i))$, where $i \in I \bigcap \{0,1\}^k$ (defines the function f_i) and $t(i)$ is a trapdoor information for f_i.*
2. *There exists a probabilistic polynomial time algorithm that on input i determines whether $f_i \in F$ (in particular whether f_i is a permutation).*
3. *There exists a random polynomial time algorithm that on input $i \in I$ chooses a random element $x \in D_i$ with uniform distribution over D_i. There exists a polynomial time algorithm that for any $i \in I$ and any x checks whether $x \in D_i$.*
4. *There exists a polynomial time algorithm A such that*

$$\forall i \in I \quad \forall x \in D_i \qquad A(i,x) = f_i(x).$$

5. *For any random polynomial time algorithm B, for every constant $c > 0$, and for every large enough integer k,*

$$Prob\{B(i, f_i(x)) = x\} < \frac{1}{k^c}$$

where the probability space is taken over the random tape of B combined with the distribution of i as generated by $G(1^k)$ and a random uniform choice of $x \in D_i$.
6. *There exists a polynomial time algorithm C such that*

$$C(i, t(i), f_i(x)) = x \qquad \forall x \in D_i, \quad \forall i \in I.$$

For simplicity, we assume that there exists a family of certified trapdoor permutations in which each of the domains $D_i = \{0,1\}^{n_i}$, where n_i is some integer that depends on i. (However, we emphasize that this assumption can be relaxed in several ways, such as allowing the domains D_i to cover a nonnegligible fraction of $\{0,1\}^{n_i}$, or using one-to-one functions rather than permutations. In particular, the number theoretic assumptions used in [BDMP]'s construction of NIZKs are a special case of our relaxed assumptions.) Under this assumption, the scheme described in section 2.2 can be modified to accommodate probabilistic polynomial time provers. Efficient P randomly chooses a trapdoor permutation $f \in F_k$, sends its index i (of length k) to V, and keeps the trapdoor information $t(i)$ secret. The ability of P to invert f efficiently is due to its knowledge of the trapdoor information.

The proof of *completeness* remains unchanged. The proof of *soundness* has to be modified for the following reason. In contrast to the scheme described in section 2.2 in which the (unbounded) prover does not choose the one-way permutation, in the efficient scheme a cheating prover may choose a particularly useful trapdoor permutation after seeing the CRS. Namely, he can choose a trapdoor permutation for which

the corresponding hidden string HRS (determined by the given CRS and this particular trapdoor permutation) does not yield any Hamiltonian matrix. Such a string enables a cheating prover to "prove" the Hamiltonicity of *any* graph, in particular non-Hamiltonian graphs, without getting caught by the verifier.

To overcome this difficulty, we only have to extend the length of the CRS. Recall (see section 2.3.2) that for any fixed (either trapdoor or one-way) permutation, if the length of the CRS is $\Omega(n^{13/2}k^2m)$ bits, then the fraction of bad CRSs (those which do not yield any Hamiltonian matrix) is e^{-nk}. Since all random bits of the CRS are independent, we conclude that if the length of the CRS is twice as long, then the fraction of bad CRSs with respect to any fixed trapdoor permutation is less than e^{-2nk}. But recall that in the new scheme the prover can choose any trapdoor permutation he wishes, out of a family of at most 2^k trapdoor permutations. Therefore, $\frac{2^k}{e^{2nk}} < 2^{-k}$ is an upper bound on the fraction of random strings which can be bad relative to some trapdoor permutation.

The proof of the zero knowledge property of our new scheme resembles its counterpart for the original scheme (with an unbounded prover), except that we have to consider a family of certified trapdoor permutations rather than a fixed one-way permutation. Namely, the probability distribution over all pairs (CRS , proof) includes now also the uniform distribution over all trapdoor permutations in this family, rather than just the uniform distribution over all random strings. Now, the intermediate simulator P' (which has an access to a Hamiltonian cycle in G) should uniformly choose a random trapdoor permutation f (without its trapdoor information) in order to have an output's distribution identical to that of the real prover. The main simulator M (which does not know any Hamiltonian cycle in G) uses the same random f, and both P' and M should apply f in the forward direction. The rest of the proof (regarding the indistinguishability between the outputs of M and P') goes exactly as introduced in section 2.3.3.

We remark that the certification assumption (part 2 of Definition 2.8) can be relaxed. It states that *there exists a polynomial time algorithm that on input i determines whether* $f_i \in F$. This item is included to guarantee that a cheating prover cannot choose a function which does not belong to the prespecified family F of trapdoor permutations. In particular, if P sends to V a definition of a trapdoor function which is not a permutation at all, *soundness* does not necessarily hold, as the opening of the hard-core bits is not unique. Recently, Bellare and Yung [BY] constructed a NIZK proof system for proving that a given function (whose description is of polynomial size) is "almost permutation" (permutation on all but a small fraction of the domain). They showed that this implies that part 2 of the definition is not necessary for the construction of efficient NIZK proof systems for NP.

2.4.2. Public-key cryptosystems secure against chosen ciphertext attacks. The existence of public-key cryptosystems which are secure against passive eavesdropping under the assumption that certified trapdoor permutations exist is well known [GM, Yao, GL]. Naor and Yung [NY] show how to construct a public-key cryptosystem which is provably secure against chosen ciphertext attacks (CCS-PKC), given a public-key cryptosystem which is secure against passive eavesdropping and a NIZK proof system in the shared string model. Using their result together with our construction (for polynomial time provers) we have the following corollary.

COROLLARY 2.9. *CCS-PKC exist under the general assumption that certified trapdoor permutations exist.*

This is the first known CCS-PKC which does not rely on the hardness of specific

computational number-theoretic problems.

3. Multiple NIZK proofs based on a single random string.

3.1. Introduction. In the previous section we constructed a *bounded* NIZK proof system, in which the prover can prove a single statement. In this section we construct a *general* NIZK proof system, in which polynomially many statements can be proven by polynomially many provers independently. Our main concern will be to control the length of the common random string σ. Recall that in the case of bounded NIZKs, the length of σ is some explicit polynomial in n (the length of the statement to be proved) and k (a security parameter). One may attempt to transform a bounded NIZK proof system into a general one by reusing the bits of σ over and over again each time a new statement is proved. Unfortunately, it is not known whether the zero knowledge property is preserved if the number of statements proven exceeds $O(\log n)$. If there is an a-priori bound m on the number of statements to be proved, then an alternative approach may be to extend σ by a factor of m. However, this solution is extremely wasteful in the length of σ (and even more if m turns out to be an overestimate). Thus throughout this paper we make the requirement that *the length of σ depends on n and k alone, but not on the number of statements to be proven, which can be an arbitrary polynomial in n.*

Notation and conventions. Throughout the following sections, n denotes the size of a single input statement, whereas m denotes the number of input statements to be proved, each of length n. The value of m is a function of n (typically, some polynomial in n), though we do not introduce special notation to denote this fact. To avoid excessive use of parameters, we identify the security parameter k with the length n of a single input statement (see the remark following Definition 1.2). We use the $\nu(n)$ notation of section 1.3 whenever it is not a source for confusion. We assume that outputs of provers (denoted by $P(x, w, \sigma)$) include explicitly the input x and the CRS σ. Recall our nonstandard use of *ensembles* (see section 1.3) in which the ensemble for the simulator is indexed by inputs $x \in L_R$, whereas the ensemble for the truthful prover is indexed by inputs $x \in L_R$, together with a valid witness w for each input. The two ensembles were said to be indistinguishable if, for any (large enough) input $x \in L_R$, the corresponding distributions (the prover's proofs and the simulator's simulated proofs) were indistinguishable, regardless of the witness w used by the truthful prover. We extend this concept of ensembles to accommodate multiple noninteractive proofs. The ensemble for the simulator is indexed by sequences of equal length inputs in L_R, where, for each sequence, its length m is bounded by some polynomial in the length n of single input statements in the sequence. The ensemble for the truthful prover is indexed by a sequence of equal length inputs in L_R, together with a sequence of corresponding valid witnesses. The two ensembles are indistinguishable if for any sequence of equal length inputs in L_R the corresponding distributions (the prover's sequence of proofs and the simulator's simulated proofs) are indistinguishable, regardless of the sequence of witnesses used by the truthful prover. For computational indistinguishability, the probability of distinguishing between the two ensembles must decrease as fast as $\nu(n)$, which is equivalent to $\nu(mn)$, by our requirement that m is polynomial in n. The distinguishing algorithm is denoted by D and runs in nonuniform polynomial time. Hence $\forall D$ quantifies over all nonuniform polynomial time algorithms. D will typically receive the output of the prover as input, which by our convention regarding $P(x, w, \sigma)$ implies that D also sees x and σ.

DEFINITION 3.1. *A noninteractive proof systems for the language L_R is* general zero knowledge *if there exists a random polynomial time simulator M such that for*

any positive constant c, for any $m \leq n^c$, the two ensembles $\{(\sigma, P(x_1, w_1, \sigma), \ldots, P(x_m, w_m, \sigma))\}$ and $\{M(x_1, x_2, \ldots, x_m)\}$ are computationally indistinguishable. In any sequence of instances that indexes the ensembles, all x_i are of the same length (denoted by n), and for all $1 \leq i \leq m$, $(x_i, w_i) \in R$.

Remark. An important feature of our definition of general zero knowledge noninteractive proof systems is that each of the statements x_j is proven independently. Consequently, polynomially many provers can share the same random reference string σ and prove polynomially many statements independently. A somewhat weaker definition, in which the proof of statement x_j may depend on the proof of previous statements, is given in [BDMP]. Their definition applies only to the case that a single prover uses σ to prove polynomially many statements. The construction that they propose does not support polynomially many independent provers (i.e., our stronger definition).

In this section we show how to transform any bounded NIZK proof system for an NP complete language L_R into a general NIZK proof system for the same language L_R. Our transformation uses the NP-completeness of L_R in an essential way, and does not handle cases in which L_R is not NP-complete. Our transformation applies only to NIZK proof systems with efficient provers (and does not apply to NIZK proof systems such as that of section 2.2 in which P inverts one-way permutations).

We now give a quick overview of our construction. It is based on the concept of *witness indistinguishability* [FS], which informally means that it is intractable to distinguish which of two possible witnesses P is using in his proof of an NP statement. We prove that any NIZK proof system with efficient provers is also witness indistinguishable. Furthermore, the witness indistinguishability property is preserved even if polynomially many noninteractive witness indistinguishable proofs are given using the same reference string (again, provided that the prover in each individual proof is efficient).

If one could argue that any sequence of noninteractive witness indistinguishable proofs is also zero knowledge then we would be done. It is not true that in general witness indistinguishability implies zero knowledge, but there are special cases where this implication holds. We show, under the assumption that one-way functions exist, that any NIZK proof system for any NP-complete language can be modified to a new noninteractive proof system for which witness indistinguishability always implies zero knowledge.

3.2. Noninteractive witness indistinguishability. In this subsection we define the concept of noninteractive witness indistinguishability and prove some of its important properties.

DEFINITION 3.2. *A noninteractive proof system (P, V) is* bounded witness indistinguishable *over R if for any large enough input x, any $w_1, w_2 \in w(x)$, and for a randomly chosen reference string σ, the ensembles which differ only in the witness that P is using, but not in x or σ, are computationally indistinguishable. In more detail,*

$\forall D\ \exists N\ \forall n > N\ \forall x \in L_R \bigcap \{0,1\}^n\ \forall w_1, w_2 \in w(x)$,

$$\sum_{\sigma} 2^{-|\sigma|} \cdot |\ \mathrm{Prob}(D(P(x, w_1, \sigma)) = 1), - \mathrm{Prob}(D(P(x, w_2, \sigma)) = 1)| \ < \ \nu(n).$$

The probability space is that of P's random coin tosses.

We remark that there are two plausible ways of defining noninteractive witness indistinguishability. In the first alternative, the proofs that use different witnesses

need to look similar when both use the same CRS σ. In the second alternative, each proof may use a different σ (that is, for each witness we first average the output of the distinguisher over all choices of σ, and only then compare between the use of different witnesses). The first alternative is stronger, and we adopted it in Definition 3.2. The second alternative is also useful, as it relates more naturally to noninteractive zero knowledge, where σ produced by the simulator M is not required to be identical to σ used by the prover. For the case of efficient provers, the following lemma shows the equivalence of the two alternatives.

LEMMA 3.3. *Noninteractive proof system (P, V) with efficient provers is* bounded witness indistinguishable *over R if and only if*

$\forall D\ \exists N\ \forall n > N\ \forall x \in L_R \bigcap \{0,1\}^n\ \forall w_1, w_2 \in w(x)$,

$$\left| \sum_{\sigma} 2^{-|\sigma|} (Prob(D(P(x, w_1, \sigma)) = 1) - Prob(D(P(x, w_2, \sigma)) = 1)) \right| < \nu(n).$$

Proof. The "only if" direction is obvious. We prove only the "if" direction.

Assume that for some infinite sequence $\mathcal{I}$ of triplets (x, w_1, w_2) of inputs together with their respective witnesses, some nonuniform polynomial time algorithm D can distinguish between P using witness w_1 and P using witness w_2. Formally, for some $k > 0$,

$$\sum_{\sigma} 2^{-|\sigma|} \cdot |Prob(D(P(x, w_1, \sigma)) = 1) - Prob(D(P(x, w_2, \sigma)) = 1)| > \frac{1}{(|x|)^k},$$

where $(x, w_1, w_2) \in \mathcal{I}$, and the probabilities are taken over the random choices of P. We construct a new nonuniform *random* polynomial time distinguisher D', which uses "knowledge" of both w_1 and w_2 and contradicts the condition of the lemma. (D' can be transformed into a *deterministic* nonuniform distinguisher by standard averaging techniques.) The distinguisher D' essentially simulates the behavior of D but with the following modification: on public random strings σ for which $Prob(D(P(x, w_1, \sigma)) = 1) < Prob(D(P(x, w_2, \sigma)) = 1)$, algorithm D' inverts the output of D so as to prevent a cancelation effect between σ with the above property and σ for which $Prob(D(P(x, w_1, \sigma)) = 1) > Prob(D(P(x, w_2, \sigma)) = 1)$.

On input z, a noninteractive proof for x using the public random string σ, D' operates as follows. First, ignoring z, algorithm D' performs the following *bias test* for σ, obtaining a "bias indicator" b. Algorithm D' generates $|x|^{k+1}$ independent strings from each of the distributions $P(x, w_1, \sigma)$ and $P(x, w_2, \sigma)$ (we note that this is possible because P is polynomial time, and D' can simulate P with the relevant auxiliary input). D' feeds these strings to D, obtaining from D two sequences of output bits, each of length $|x|^{k+1}$. If the first such sequence (corresponding to w_1) contains less 1 entries than the second (corresponding to w_2), then b is set to 1. Otherwise b is set to 0. Then D' feeds D with z and flips the output of D if and only if $b = 1$.

It is a simple matter to show that

$$\left| \sum_{\sigma} 2^{-|\sigma|} (Prob(D'(P(x, w_1, \sigma)) = 1) - Prob(D'(P(x, w_2, \sigma)) = 1)) \right| > \frac{1}{2(|x|)^k}. \qquad \square$$

Remark. The above proof uses the fact that P is efficient (i.e., polynomial time). The equivalence between definitions might not hold if the prover is nonpolynomial.

Consider for example the zero knowledge proof system of section 2.2 in which the prover inverts one-way functions. It is witness indistinguishable in the sense of Lemma 3.3—this can be proved in a way similar to the proof of Lemma 3.4 below. However, it is not witness indistinguishable in the sense of Definition 3.2: the prover is deterministic, and hence for any particular σ, the use of different witnesses by the prover is distinguishable by examining a single bit location (that may depend on σ) of the prover's output. An averaging argument shows that there is some bit location that (for a polynomial fraction of the possible choices of σ) distinguishes between the two witnesses that the prover may be using.

LEMMA 3.4. *Any bounded NIZK proof system with polynomial time prover is also a bounded noninteractive witness indistinguishable proof system.*

Proof. Assume that the proof system is not witness indistinguishable. By Lemma 3.3, for some constant k and an infinite sequence of inputs there exists a distinguisher D that satisfies

$$\left| \sum_{\sigma} 2^{-|\sigma|} (Prob(D(P(x, w_1, \sigma)) = 1) - Prob(D(P(x, w_2, \sigma)) = 1)) \right| > \frac{1}{(|x|)^k}.$$

Now the proof system cannot be zero knowledge. For consider any proposed simulator M. No matter what value $Prob(D(M(x)) = 1)$ takes on, it differs either from $Prob(D(P(x, w_1)) = 1)$ or from $Prob(D(P(x, w_1)) = 1)$ by at least $\frac{1}{2(|x|)^k}$. Since both the latter cases are valid distributions of noninteractive proofs for x, we conclude that D is a distinguisher which fails any simulator. ∎

DEFINITION 3.5. *A noninteractive proof system is* general witness indistinguishable *over R if for any positive constant c, for any $m \leq n^c$, we have that the two ensembles $\{(P(x_1, w_1^1, \sigma), P(x_2, w_2^1, \sigma), \ldots, P(x_m, w_m^1, \sigma))\}$ and $\{(P(x_1, w_1^2, \sigma), P(x_2, w_2^2, \sigma), \ldots, P(x_m, w_m^2, \sigma))\}$ are computationally indistinguishable (in the sense of Definition 3.2, where σ is identical in the two ensembles). In more detail,*

$$\forall D\ \forall c\ \exists N\ \forall n > N\ \forall m < n^c$$

whenever $x_i \in L_R \bigcap \{0,1\}^n$ and $w_i^1, w_i^2 \in w(x_i)$ for all $1 \leq i \leq m$, then

$$\sum_{\sigma} 2^{-|\sigma|} \cdot |Prob(D(P(x_1, w_1^1, \sigma), \ldots, P(x_m, w_m^1, \sigma)) = 1)$$

$$-Prob(D(P(x_1, w_1^2, \sigma), \ldots, P(x_m, w_m^2, \sigma)) = 1)| < \nu(n).$$

The probability space is that of P's random coin tosses.

LEMMA 3.6. *Any bounded noninteractive witness indistinguishable proof system with efficient provers is also general witness indistinguishable.*

Proof. Assume that for some constant c and infinitely many n the following holds: there exists a distinguisher D of size at most n^c, a positive integer m, where $m \leq n^c$, a sequence $\mathcal{X} = (x_1, x_2, \ldots, x_m)$ of inputs (each of size n), and two sequences, $\mathcal{W}^1 = (w_1^1, \ldots, w_m^1)$ and $\mathcal{W}^2 = (w_1^2, \ldots w_m^2)$, of witnesses for the respective $x_i \in \mathcal{X}$, such that

$$\sum_{\sigma} 2^{-|\sigma|} \cdot |Prob(D(P(x_1, w_1^1, \sigma), \ldots, P(x_m, w_m^1, \sigma)) = 1)$$

$$-Prob(D(P(x_1, w_1^2, \sigma), \ldots, P(x_m, w_m^2, \sigma)) = 1)| > n^{-c},$$

where probabilities are taken over the random coin tosses of P. Then by the "hybrid" argument of [GM] (also known as "probability walk" argument), there must be a "polynomial jump" somewhere in the execution: there exists k, where $1 \leq k \leq m$, such that

$$\sum_{\sigma} 2^{-|\sigma|} \cdot |Prob(D(P(x_1, w_1^1, \sigma), \ldots, P(x_k, w_k^1, \sigma), \ldots, P(x_m, w_m^2, \sigma)) = 1)$$

$$-Prob(D(P(x_1, w_1^1, \sigma), \ldots, P(x_k, w_k^2, \sigma), \ldots, P(x_m, w_m^2, \sigma)) = 1)| \; > \; \frac{1}{mn^c}.$$

We now use the nonuniformity of the distinguishers to derive a contradiction. Since the proof system has efficient provers, the whole set of proofs $(P(x_1, w_1^1, \sigma)$, $\ldots, P(x_{k-1}, w_{k-1}^1, \sigma)$, $P(x_{k+1}, w_{k+1}^2, \sigma), \ldots, P(x_m, w_m^2, \sigma))$ can be simulated by a modified D', who has as auxiliary input $((x_1, w_1^1), \ldots, (x_{k_1}, w_{k-1}^1), (x_{k+1}, w_{k+1}^2), \ldots, (x_m, w_m^2))$. This random nonuniform polynomial time D' can now distinguish between proofs for x_k in which the prover uses w_k^1 and proofs in which the prover uses w_k^2. This contradicts our assumption that the original protocol was witness indistinguishable. □

3.3. The transformation. We assume the existence of pseudorandom bit generators (see [BM], [Yao]), which extend n-bit random seeds to $2n$-bit pseudorandom strings, computationally indistinguishable from strings of truly random $2n$ bits. The existence of pseudorandom generators follows from the assumption that one-way functions exist [ILL], [Ha].

Let (P, V) be any bounded NIZK proof system with polynomial time prover for the NP-complete language L_R. We construct $(\bar{\mathrm{P}}, \bar{\mathrm{V}})$, a general NIZK proof system for L_R, under the sole assumption that one-way functions exist.

Let $g : \{0,1\}^n \longrightarrow \{0,1\}^{2n}$ be a pseudorandom bit generator. We introduce two new NP languages. L_{R_g} is the NP language corresponding to the relation $R_g(y, s)$ iff $g(s) = y$. $L_{R_\#}$ is the NP language corresponding to the relation $R_\#(x\#y, w)$ iff *either* $R(x, w)$ *or* $R_g(y, w)$, where $\#$ is used as a special delimiting character in the alphabet of the inputs to $L_{R_\#}$.

We now describe the algorithm of $\bar{\mathrm{P}}$ on input $(x, w) \in R$ and reference string σ.

1. Divide the CRS σ into two segments: the first $2n$ bits, denoted by y, and called the *reference statement*; the rest of the CRS is denoted by σ'.
2. Construct the instance $x\#y \in L_{R_\#}$. Observe that w, the witness that $\bar{\mathrm{P}}$ has for $x \in L_R$, is a witness for $x\#y \in L_{R_\#}$.
3. Reduce $x\#y$ to an instance X of the NP-complete language L_R, using a publicly known reduction with efficient transformation of witnesses. (That is, any witness for the original instance can be efficiently transformed into a witness for the target instance, and vice versa. Known reductions to NP-complete languages have this property.) Reduce w, the witness for $x\#y \in L_{R_\#}$, to W, a witness for $X \in L_R$.
4. Send x, X, and $P(X, W, \sigma')$. (The last term, $P(X, W, \sigma')$, is a random variable that denotes the NIZK proof that the prover P from the system (P, V) would produce on input X, witness W, and reference string σ', depending on P's private random string.)

The verifier $\bar{\mathrm{V}}$ accepts if the publicly known reduction (from $L_{R_\#}$ to L_R) gives X when applied to $x\#y$, and if furthermore V would have accepted $P(X, W, \sigma')$.

THEOREM 3.7. *Under the assumptions that (P, V) is a bounded NIZK proof system with efficient provers for L_R, that L_R is NP-complete, and that g is a pseudo-*

random generator, the above transformed scheme is a general NIZK proof system for L_R.

Proof. We first give an intuitive introduction to the full proof. The completeness, soundness, and zero knowledge properties of $(\bar{\mathrm{P}}, \bar{\mathrm{V}})$ are based on the corresponding properties of (P, V). Efficiency is preserved in the completeness property since from a witness to x, prover $\bar{\mathrm{P}}$ can derive a witness to X, and thus execute the bounded NIZK proof system (P, V). The soundness property follows from the fact that y is chosen as a truly random (rather than pseudorandom) string. Thus, for almost all possible choices of y, it is not in the range of g, and consequently $X \in L_R$ if and only if $x \in L_R$. The zero knowledge property requires more subtle analysis. The simulation of $(\bar{\mathrm{P}}, \bar{\mathrm{V}})$ is done by replacing the reference statement y by a pseudorandom string y'. This y' is generated by selecting at random an n bit seed s and computing $y' = g(s)$. Since g is a pseudorandom bit generator, this replacement is indistinguishable to polynomial time observers. Now any statement x with witness w is transformed into a statement X which also has s, the seed of y', as its witness. The simulator uses s instead of w in order to prove X. The concept of witness indistinguishability can now be used to show that this change of witnesses in the proof of X is indistinguishable to polynomial time observers, even if it is done polynomially many times. We now give a more detailed proof.

Completeness (while preserving efficiency). The reduction of $L_{R_\#}$ to L_R allows efficient transformation of witnesses. Thus $\bar{\mathrm{P}}$, who knows a witness for $x \in L_R$, and hence for $x\#y \in L_{R_\#}$, can also compute in polynomial time a witness for X and use it in order to perform the protocol. The reduction is also publicly known, and so $\bar{\mathrm{V}}$ can check that it was followed correctly. The completeness property then follows from the completeness property of (P, V).

Soundness. From the soundness property of (P, V) it follows that either $x \in L_R$ or y is in the range of the generator g (i.e., there is an s such that $y = g(s)$). But simple counting shows that the probability that the random string y of length $2n$ is in the range of g (i.e., has a seed of length n) is at most 2^{-n}, and thus with overwhelming probability indeed $x \in L_R$.

The completeness and soundness property trivially continue to hold even if polynomially many statements are proved.

Zero knowledge. For any large enough value of n, consider any sequence (x_1, w_1), (x_2, w_2), $\ldots$, (x_m, w_m), of inputs together with their respective witnesses, where m is polynomial in n. Assume that $\bar{\mathrm{P}}$ proves for these inputs membership in L_R (by using private coin tosses, the associated witnesses, and the CRS σ). We construct a simulator M which creates an ensemble indistinguishable from the ensemble that $\bar{\mathrm{P}}$ produces. M receives as input only the sequence of instances $\{x_i\}$, without their respective witnesses.

M randomly selects an n bit seed s and computes the $2n$-bit string $y' = g(s)$, to be used as the reference statement (instead of a random y used in reality). M generates a truly random reference string σ'. For each $1 \le j \le m$, M reduces $x_j\#y'$ to an instance X_j of L_R and derives from s a witness w' for this instance. Then M uses the proof system (P, V) and the reference string σ' to simulate a proof that $X \in L_R$, by using its knowledge of the seed s (rather than a witnesses it does not have to the statements x_j).

In order to prove that M's simulation is indistinguishable from $\bar{\mathrm{P}}$'s proofs, we construct a hybrid $\bar{\mathrm{M}}$, which constructs y' pseudorandomly as M does, but simulates the proofs using w_1, w_2, $\ldots$ as $\bar{\mathrm{P}}$ does.

LEMMA 3.8. *For any positive constant c, for any $m \leq n^c$, the two ensembles $\{(\sigma, \bar{P}(x_1, w_1, \sigma), \ldots, \bar{P}(x_m, w_m, \sigma))\}$ and $\{\bar{M}((x_1, w_1), (x_2, w_2), \ldots, (x_m, w_m))\}$ are computationally indistinguishable. In any sequence of instances that indexes the ensembles, all x_i are of the same length (denoted by n), and for all $1 \leq i \leq m$, $(x_i, w_i) \in R$.*

Proof. Assume otherwise, and let D be a distinguisher that distinguishes between $\bar{\mathrm{M}}$'s output and $\bar{\mathrm{P}}$'s output. We construct a distinguisher D' that distinguishes between truly random strings, and outputs of the generator g, contradicting its pseudorandomness. For infinitely many values of n, nonuniform algorithm D' has as auxiliary input the corresponding sequence (x_1, w_1), (x_2, w_2), $\ldots$, (x_m, w_m) for which D distinguishes between the two ensembles. In order to test whether a string y of length $2n$ was generated from the distribution of outputs of the generator g, D' generates a random reference string σ' and simulates $\bar{\mathrm{P}}$'s action on the sequence (x_1, w_1), (x_2, w_2), $\ldots$, (x_m, w_m), with respect to the reference string composed of y and σ'. Now the verdict of D of whether the output was produced by $\bar{\mathrm{P}}$ (which uses truly random y) or by $\bar{\mathrm{M}}$ (which uses pseudorandom y) forms a statistical test as to whether y was truly random or generated by g. ∎

The proof of the following lemma is the heart of our argument that the transformed scheme is general zero knowledge.

LEMMA 3.9. *For any positive constant c, for any $m \leq n^c$, the two ensembles $\{\bar{M}((x_1, w_1), (x_2, w_2), \ldots, (x_m, w_m))\}$ and $\{M(x_1, x_2, \ldots, x_m)\}$ are computationally indistinguishable. In any sequence of instances that indexes the ensembles, all x_i are of the same length (denoted by n), and for all $1 \leq i \leq m$, $(x_i, w_i) \in R$.*

Proof. Consider what the simulators $\bar{\mathrm{M}}$ and M actually do. They are giving NIZK proofs for a sequence of statements $X_1, \ldots, X_m$, derived from the sequence $x_1, \ldots, x_m$, by taking into account a reference statement y'. In more detail, they first generate a reference statement y', and thereafter each of them uses the truly random reference string σ' to execute NIZK protocols for the sequence $X_1, \ldots, X_m$, in the same way as a true prover in (P, V) would execute such protocols. The only difference between $\bar{\mathrm{M}}$ and M is in the sequence of witnesses that they are using, where $\bar{\mathrm{M}}$ uses the sequence $w_1, \ldots, w_m$, and M repeatedly uses s as a witness for each of the X_i (recall that s is the seed that was used in order to generate y').

Protocol (P, V) is zero knowledge. By Lemma 3.4 it is witness indistinguishable. By Lemma 3.6, even polynomially many executions of (P, V) on the same reference string σ' are witness indistinguishable. Hence the ensembles that $\bar{\mathrm{M}}$ and M create are indistinguishable. ∎

From the two lemmas above it follows that the outputs of M and $\bar{\mathrm{P}}$ are computationally indistinguishable, which completes the proof that the protocol is zero knowledge. ∎

Remarks.

1. In giving multiple NIZK proofs, the prover reuses public randomness (the CRS) over and over again. However, the prover must use "fresh" private randomness (internal random coin tosses) in each execution of the protocol. Otherwise, zero knowledge might not be preserved. In particular, when applying our transformation to the efficient bounded NIZK described in section 2.4, the truthful prover is expected to use a different trapdoor permutation for each input statement being proved.
2. The complexity of the pseudorandom bit generator g has major impact on the overall complexity of our transformation. The choice of g (and, in par-

ticular, the time required for a nondeterministic Turing machine to accept the language L_{R_g}) influences the size of X, the statement that the truthful prover eventually proves (which may be much larger than the size of x, the original statement that the prover wants to prove). There is a spectrum of constructions of pseudorandom bit generators, where the complexity of the construction typically depends on the strength of the underlying computational complexity assumptions (e.g., compare [BM] with [ILL, Ha]). In applying our transformation, one should first determine the computational complexity assumptions that were made in the construction of the particular bounded NIZK proof system, and based on them, select the simplest pseudorandom bit generator.

3. The term *bounded* NIZK proof systems indicates that the length of the CRS bounds the size of the (single) NP statement that can be proven. We have seen that using *general* NIZK proof systems, polynomially many NP statements, each of bounded length, can be proved. [BDMP] show that under the assumption that one-way functions (and hence encryption schemes) exist, general NIZK proof systems can be used in order to prove statements that are longer than the bound implied by the length of the CRS. For completeness, we sketch how this is done.
   On input of a satisfiable 3-SAT formula Φ (any other NP-statement can be reduced to 3-SAT, if necessary), the prover encrypts separately the satisfying value of each variable and for each clause $C \in \Phi$ constructs the string composed of the concatenation of the encrypted values of the three variables in C. Observe that the length of each such string does not depend on the length of Φ. Each string is treated as an NP-statement: "there exists a decryption of the encrypted values that would show that clause C is satisfiable." The prover gives a NIZK for each such statement separately, and the verifier verifies that each of the NIZK proofs is acceptable.

4. Security against adaptive attacks.

4.1. Definitions. NIZK proof systems are useful design primitives in the construction of cryptographic schemes, such as signature schemes [BG] and encryption schemes [NY]. It is often required that the cryptographic scheme will be robust against attacks of adaptive nature, which are the strongest types of attack. For example, a standard security requirement of signature schemes is that even after requesting signatures of polynomially many messages of his choice, the adversary is not able to forge a signature to any new message. In order to treat adaptive attacks we extend the security requirements of NIZK proof systems.

In a typical adaptive scenario, a polynomial time adversary A repeatedly selects statements and observes their noninteractive proofs. His goal is to come up with a statement x on which one of the three basic properties of NIZK proof systems is violated: either x is true but P cannot produce a noninteractive proof for it (violating the *completeness* condition) or x is false but there exists a noninteractive "proof" that convinces V to accept x (violating the *soundness* condition), or x is true and the adversary can extract useful information from the noninteractive proof that P gives (violating the *zero knowledge* property).

Definition 1.2 is strong enough to serve as the definition of *completeness* and *soundness* for the adaptive scenario. However, Definition 3.1 of *zero knowledge* needs to be modified. We define the concept of *adaptive zero knowledge* in a way similar to Bellare and Goldwasser [BG] (see also [GGM]'s test for pseudorandom functions).

We call this test *adaptive indistinguishability*, or the AI test (partially because of its origins as Turing's test for artificial intelligence). In our AI test, an adversary A is confronted with a blackbox B. His goal is to determine whether B contains a real prover P or whether it contains a simulator M. A first requests the random reference string σ from B. If P is inside the box, it replies with a truly random string. If M is inside the box, it replies with a string of its choice. Now, possibly based on σ, A generates a pair $(x, w) \in R$ and sends it to B. If P is inside the box, it produces a noninteractive proof $P(x, w, \sigma)$. If M is inside the box, w is "magically" filtered away, and M must simulate a noninteractive proof for x. This procedure of adaptively choosing theorems and receiving noninteractive proofs for them is repeated polynomially many times until A is ready to pass a decision: $'0'$ or $'1'$. (M, P) are said to pass the AI test if the probabilities that A outputs $'1'$ when M is inside the box and when P is inside the box are equal up to negligible additive terms.

DEFINITION 4.1. *A noninteractive adaptive proof system* (P, V) *is* adaptive zero knowledge *if there exists a random polynomial time simulator* M *such that* (M, P) *pass the AI test for any nonuniform polynomial time adversary* A.

We remark that in [BG]'s definition, A is not required to supply witnesses for $x \in L_R$ to the blackbox. This leaves open the question of how the real prover P (which is assumed to be efficient) comes up with a witness w for $x \in L_R$, to be used in producing a NIZK proof for this fact. One possibility is that some computationally unbounded agent produces this witness for P. Another is that A has to produce the witness. In our definition we choose the latter possibility, with the intention of using it only in applications where all parties are (nonuniform) polynomial time. We do not know if the theorems to follow (and, in particular, Theorem 4.4) hold also with respect to the stronger definition of zero knowledge, in which a computationally unbounded agent produces the witnesses.

PROPOSITION 4.2. *Any noninteractive proof system which is adaptive zero knowledge (Definition* 4.1*) is also general zero knowledge (Definition* 3.1*).*

Proof. The proof follows directly from the nonuniformity of the adversary in Definition 4.1. If there is a sequence of inputs with respective witnesses that serves to defeat the general zero knowledge property (according to Definition 3.1), then the adversary A (of Definition 4.1) can hold this same sequence as auxiliary input and defeat the adaptive zero knowledge property. □

A somewhat weaker condition than adaptive zero knowledge is *single statement adaptive zero knowledge*. For such proof systems, the adversary of the AI test is allowed to request only one noninteractive proof from B before passing his judgment as to what is inside the box.

PROPOSITION 4.3. *Any noninteractive proof system which is single statement adaptive zero knowledge is also bounded zero knowledge (Definition* 1.3*).*

Proof. The proof follows directly from the nonuniformity of the adversary. □

Remark. The converse of the above proposition is probably not true. For example, consider the NIZK proposed in [BDMP] for the language NQR. [BDMP] prove that the noninteractive proof system for this language is zero knowledge by producing a simulator M that constructs a CRS only after it sees the input x. In contrast, adaptive zero knowledge postulates that σ is generated first, and x is chosen only later. Despite the fact that [BDMP]'s NIZK proof system for the language NQR is bounded zero knowledge, it is not known to be single statement adaptive zero knowledge.

4.2. Robustness of our protocols. All theorems and lemmas that deal with the adaptive zero knowledge scenario are straightforward modifications of their coun-

terparts that dealt with the nonadaptive scenario. For this reason, we only state our theorems, and give some "hints" to help the reader in modifying the proofs in previous sections so that they also apply to the adaptive case.

THEOREM 4.4. *The NIZK of section* 2 *is single statement adaptive zero knowledge.*

Proof. There are two parts to a proof that a certain protocol is zero knowledge. One part is to construct the simulator M. The other part is to prove that its output is indistinguishable from the output of the real prover.

We first address the problem of constructing the "adaptive" simulation. Consider the simulator M described in section 2. This simulator generates a reference string σ' independently of the common input x. This σ' can be used to simulate a noninteractive proof for any input x. Consequently, the same simulator can be used even if the input statement is chosen adaptively after the CRS is chosen.

The proof of indistinguishability follows the same arguments as those which are used in the proof of subsection 2.3.3. Recall that it was shown that if the proof system is not zero knowledge on input $(x, w) \in R$, then one could construct an efficient algorithm (denoted by C) that predicts the hard bits of the one-way permutation (or trapdoor permutation, if the prover is required to be efficient). This algorithm had (x, w) as auxiliary input and constructed a reference string σ in the course of its operation. For the case of adaptive zero knowledge, the end result would again be an efficient algorithm C that predicts the hard bits of the one-way permutation. However, this algorithm would not explicitly receive any $(x, w) \in R$ as auxiliary input. The reason for this is that in the adaptive scenario, the adversary A need not have a prespecified (x, w) that it uses in foiling the zero knowledge property. Instead, A may generate (x, w) only after observing σ. Likewise, we must allow C to generate (x, w) only after constructing σ, in a way similar to A. Hence, instead of supplying C with explicit (x, w) as auxiliary input, we supply it with the auxiliary input of A, which C can later use to generate (x, w) that depend on σ. □

In the rest of this section we consider only NIZK proofs with efficient provers.

THEOREM 4.5. *The transformation of section* 3.3 *transforms efficient noninteractive single statement adaptive zero knowledge proof systems into efficient (general) noninteractive adaptive zero knowledge proof systems.*

Proof. Theorem 4.5 is proved in the same way as Theorem 3.7, which is its nonadaptive counterpart. We only sketch the modifications that are necessary.

The proof of the *completeness* and *soundness* conditions is straightforward. The main emphasis is on the proof of the *adaptive zero knowledge* property. Recall that in order to prove Theorem 3.7, we used the concept of *witness indistinguishability*. In order to prove Theorem 4.5, we define a corresponding notion of *adaptive witness indistinguishability*. Once this concept is defined, and its main properties are established (see Lemmas 4.7 and 4.8 below), the proof of Theorem 4.5 can be carried out in a way similar to the proof of Theorem 3.7.

In the AI test for witness indistinguishability a nonuniform polynomial time adversary A is confronted with a blackbox B that may contain one of two possible provers, denoted by P_1 and P_2. The provers differ in the witnesses that they select to use in producing NIZK proofs, and the goal of A is to determine whether B contains P_1 or P_2. A first requests the random reference string σ from B. Now, possibly based on σ, A generates a triplet (x, w^1, w^2), where $(x, w^1) \in L_R$ and $(x, w^2) \in L_R$ and sends the triplet to B. If P_1 is inside B, it uses only the first of the given witnesses for x to generate the noninteractive proof $P(x, w^1, \sigma)$. If P_2 is inside B, it uses only the

second of the given witnesses to generate $P(x, w^2, \sigma)$. This procedure of adaptively choosing theorems and receiving noninteractive proofs for them is repeated polynomially many times until A is ready to pass a decision: $'0'$ or $'1'$. (P_1, P_2) are said to pass the AI test for witness indistinguishability if the probabilities that A outputs $'1'$ when P_1 is inside the box and when P_2 is inside the box are equal up to negligible additive terms.

DEFINITION 4.6. *A noninteractive adaptive proof system* (P, V) *is* adaptive witness indistinguishable *if* (P_1, P_2) *pass the AI test for witness indistinguishability.*

In *single statement adaptive witness indistinguishability*, the adversary A is allowed to request only one noninteractive proof from B before passing his judgment of whether P_1 or P_2 is inside the box.

LEMMA 4.7. *Any noninteractive proof system which is single statement adaptive zero knowledge is also single statement adaptive witness indistinguishable.*

The proof of Lemma 4.7 is similar to the proof Lemma 3.4 and is omitted.

The following lemma shows that adaptive witness indistinguishability is preserved under repeated applications of the noninteractive proof system with the same random reference string.

LEMMA 4.8. *Any noninteractive proof system which is single statement adaptive witness indistinguishable is also adaptive witness indistinguishable.*

Proof. Assume that there exists an adversary A which adaptively generates triplets (x_i, w_i^1, w_i^2) (for $i \geq 1$) and can distinguish between P_1 and P_2. By the "hybrid" argument of [GM], there must be a "polynomial jump" somewhere in the execution: there exists k, such that if for $i < k$ the adversary uses the first of the two generated witnesses of each instance to produce $P(x_i, w_i^1, \sigma)$ by itself, then generates (x_k, w_k^1, w_k^2) and gives it to the blackbox, and finally (for $i > k$) uses the second of the two generated witnesses of each instance to produce $P(x_i, w_i^2, \sigma)$ by itself, then A can distinguish between the case that P_1 is inside the box and the case that P_2 is inside the box. This contradicts our assumption that the original protocol was single statement adaptive witness indistinguishable. □

The rest of the proof of Theorem 4.5 can be carried out in a way similar to the proof of Theorem 3.7. The details are omitted. □

5. Conclusions. We show how one can construct general NIZK proof systems under general computational complexity assumptions. Theoretically, NIZK proof systems have numerous cryptographic applications ([BG], [NY], and we are confident that more will follow). However, to be useful in practice, the efficiency of NIZK proof systems must be greatly improved. One parameter that deserves special attention is the length of the CRS. To prove Hamiltonicity of n-node graphs, our NIZK proof system requires $|\sigma| = \Omega(n^{11/2})$. Recently, Kilian [K94] presented considerably more efficient NIZK proof systems for circuit satisfiability, and this was further simplified and improved by Kilian and Petrank [KP], to a point where the complexity of NIZK proof systems for NP statements almost matches that of the most efficient known *interactive* zero knowledge proof systems.

The following question remains open: what are the minimal computational complexity assumptions that support bounded NIZK proof systems?

Recall that once bounded NIZK proof systems with efficient provers are constructed, the transformation to general NIZK proof systems requires only the assumption that one-way functions exist and are relatively efficient. Since the transformation to general NIZK proof systems requires only bounded noninteractive witness indistinguishable proof systems as a starting point, the above question can be reformulated

with NIZK replaced by noninteractive witness indistinguishability.

Acknowledgments. We thank Moni Naor for helpful discussions, Oded Goldreich for his useful remarks on an early version of this manuscript, and the referees of this paper for greatly improving its readability.

REFERENCES

[BG] M. Bellare and S. Goldwasser, *New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs*, in Advances in Cryptology-CRYPTO 89, Lecture Notes in Computer Science 435, Springer-Verlag, New York, 1989, pp. 194–211.

[BeMi] M. Bellare and S. Micali, *Non-interactive oblivious transfer and applications*, in Advances in Cryptology-CRYPTO 89, Lecture Notes in Computer Science 435, Springer-Verlag, New York, pp. 547–557.

[BY] M. Bellare and M. Yung, *Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutations*, J. Cryptology, 9 (1996), pp. 149–166.

[Blum] M. Blum, *How to prove a theorem so no one else can claim it*, in Proc. of the International Congress of Mathematicians, Berkeley, CA, 1986, pp. 1444–1451.

[BDMP] M. Blum, A. De Santis, S. Micali, and G. Persiano, *Noninteractive zero-knowledge*, SIAM J. Comput., 20 (1991), pp. 1084–1118.

[BFM] M. Blum, P. Feldman, and S. Micali, *Noninteractive zero-knowledge and its applications*, in Proc. of 20th ACM Symposium on Theory of Computing, ACM Press, NY, 1988, pp. 103–112.

[BM] M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM J. Comput., 13 (1984), pp. 850–864.

[DMP87] A. De Santis, S. Micali, and G. Persiano, *Non-interactive zero-knowledge proof systems*, in Advances in Cryptology-CRYPTO 87, Lecture Notes in Computer Science 293, Springer-Verlag, New York, 1987, pp. 52–72.

[DMP88] A. De Santis, S. Micali, and G. Persiano, *Non-interactive zero-knowledge with preprocessing*, in Advances in Cryptology-CRYPTO 88, Lecture Notes in Computer Science 403, Springer-Verlag, New York, 1988, pp. 269–283.

[DP] A. De Santis and G. Persiano, *Zero-knowledge proofs of knowledge without interaction*, in Proc. of 33rd IEEE Annual Symposium on Foundations of Computer Scienc, 1992, pp. 427–436.

[DY] A. De Santis and M. Yung, *Cryptographic applications of the non-interactive metaproof and many-prover systems*, in Advances in Cryptology-CRYPTO 90, Lecture Notes in Computer Science, 537, Springer-Verlag, New York, 1990, pp. 366–377.

[FS] U. Feige and A. Shamir, *Witness indistinguishable and witness hiding protocols*, in Proc. of 22nd ACM Symposium on Theory of Computing, ACM Press, NY, 1990, pp. 416–426.

[G] O. Goldreich, *A Uniform-Complexity Treatment of Encryption and Zero-Knowledge*, TR-568, Computer Science Dept., Technion, Haifa, Israel, 1989.

[GGM] O. Goldreich, S. Goldwasser, and S. Micali, *How to construct random functions*, J. Assoc. Comput. Mach., 33 (1986), pp. 792–807.

[GILVZ] O. Goldreich, R. Impagliazzo, L. Levin, R. Venkatesan, and D. Zuckerman, *Security preserving amplification of hardness*, in Proc. of 31st IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 318–326.

[GL] O. Goldreich and L. Levin, *A hard-core predicate for all oneway functions*, in Proc. of 21st ACM Symposium on Theory of Computing, ACM Press, NY, 1989, pp. 25–32.

[GMW] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems*, J. Assoc. Comput. Mach., 38 (1991), pp. 691–729.

[GM] S. Goldwasser and S. Micali, *Probabilistic Encryption*, J. Comput. System Sci., 28 (1984), pp. 270–299.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.

[Ha] J. Hastad, *Pseudo-random generators under uniform sssumptions*, in Proc. of 22nd ACM Symposium on Theory of Computing, ACM Press, NY, 1990, pp. 395–404.

[ILL] R. Impagliazzo, L. Levin, and M. Luby, *Pseudorandom generation from oneway functions*, in Proc. of 21st ACM Symposium on Theory of Computing, ACM Press, NY, 1989, pp. 12–24.

[K94] J. Kilian, *On the complexity of bounded-interaction and moninteractive zero-knowledge proofs*, in Proc. 35th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 466–477.

[KMO] J. Kilian, S. Micali, and R. Ostrovsky, *Minimum resource zero-knowledge proofs*, in Proc. 30th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1989, pp. 474–479.

[KP] J. Kilian and E. Petrank, *An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions*, J. Cryptology, 11 (1998), pp. 1–27.

[LS] D. Lapidot and A. Shamir, *Publicly verifiable non-interactive zero-knowledge proofs*, in Advances in Cryptology-CRYPTO 90, Lecture Notes in Computer Science, 537, Springer-Verlag, New York, 1990, pp. 353–365.

[Naor] M. Naor, *Bit commitment using pseudorandomness*, J. Cryptology, 4 (1991), pp. 151–158.

[NY] M. Naor and M. Yung, *Public-key cryptosystems provably secure against chosen ciphertext attacks*, in Proc. of 22nd ACM Symposium on Theory of Computing, ACM Press, NY, 1990, pp. 427–437.

[RS] C. Rackoff and D. Simon, *Non-interactive zero-knowledge proofs of knowledge and chosen ciphertext attacks*, in Advances in Cryptology-CRYPTO 91, Lecture Notes in Computer Science, 576, Springer-Verlag, New York, 1991, pp. 433–444.

[Yao] A. C. Yao, *Theory and applications of trapdoor functions*, in Proc. 23rd IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1982, pp. 80–91.