Foundations of Cryptography
an Additional Fragment (Nr. 1)

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

February 9, 1996

Preface

This fragment contains the the section on *non-interactive zero-knowledge* which was missing from the chapter on *zero-knowledge* in my 1995 fragments of a book on Foundation of Cryptography. With this section completed, these fragments now contain a first draft for three major chapters and an introduction chapter. The three chapters are the chapters on computational difficulty (or one-way functions), pseudorandom generators and zero-knowledge. Unfortunately, I am forced to repeat my warning from the 1995 fragments:

> None of these chapters has been carefully proof-read and I expect them to be full of various mistakes ranging from spelling and grammatical mistakes to minor technical inaccuracies. I hope and believe that they are no fatal mistakes, but I cannot guarantee this either.

To further augment the current section I enclose at its end some bibliographical notes taken from the revised concluding section for the Chaper on Zero-Knowledge (i.e., Section 6.12).

6.10 * Non-Interactive Zero-Knowledge Proofs

In this section we consider 'non-interactive' zero-knowledge proof systems. Actually, the term non-interactive is somewhat misleading. Indeed, in the model which we will discuss the interaction between the prover and the verifier is minimal; it consists of the prover sending a single message to the verifier (as in the case of an NP-proof). Yet, both the prover and the verifier have access to a (trusted) random string, which can be thought of as a restricted trusted third party. Non-interactive zero-knowledge proof systems have various applications (e.g., to Encryption Schemes secure against Chosen Message Attacks and to Signature Schemes).

We start with basic definitions and constructions allowing to prove a single assertion of a-priori bounded length. Next we extend the treatment to proof systems in which many assertions of various lengths can be proven, as long as the total length of all assertions is a polynomial in a security parameter but the polynomial is NOT a-priori known. Jumping ahead, we note that, unlike the basic treatment, the extended treatment allows to prove assertions of total length much bigger than the length of the trusted random string. The relation between the total length of the provable assertions and the length of the trusted random string is analogous to the relation between the total length of messages that can be encrypted (resp., documents that can be signed) and the length of the encryption-key (resp., signing-key). We stress, however, that even handling the basic case is very challenging in the case of non-interactive zero-knowledge proofs.

6.10.1 Basic Definitions

In the setting of non-interactive proof systems, both the prover and verifier are ordinary probabilistic machines which, in addition to the common-input, also get a *common random-string*. We stress that both the prover and verifier may toss coins and get auxiliary inputs, in addition to the above common input and random-string. However, for sake of simplicity we present a definition for the case in which none of these machines gets an auxiliary input. The verifier also gets as input the output produced by the prover.

Definition 1 (non-interactive proof system): *A pair of probabilistic machines, (P, V), is called a* non-interactive proof system for a language *L if V is polynomial-time and the following two conditions hold*

- Completeness: *For every $x \in L$*

$$\text{Prob}\,(V(x, R, P(x, R)) = 1) \geq \frac{2}{3}$$

where R is a random variable uniformly distributed in $\{0,1\}^{\text{poly}(|x|)}$.

- Soundness: *For every $x \notin L$ and every machine B,*

$$\text{Prob}\,(V(x, R, B(x, R)) = 1) \leq \frac{1}{3}$$

where R is a random variable uniformly distributed in $\{0,1\}^{\text{poly}(|x|)}$.

The uniformly chosen string R is called the common random-string.

As usual, the error probability in both conditions can be reduced (from $\frac{1}{3}$) up to $2^{-\mathrm{poly}(|x|)}$, by repeating the "protocol" sufficiently many times (using a sequence of many independently chosen random-strings). In stating the soundness condition, we have deviated from common formulations which allows $x \notin L$ to be selected after R. Namely, in many sources the soundness condition is stated as

> *for every n and every pair of functions $\chi : \{0,1\}^{\mathrm{poly}(n)} \mapsto (\{0,1\}^n - L)$ and $\pi : \{0,1\}^{\mathrm{poly}(n)} \mapsto \{0,1\}^{\mathrm{poly}(n)}$*
>
> $$\mathrm{Prob}\left(V(\chi(R), R, \pi(R)) = 1\right) \leq \frac{1}{3}$$
>
> *where R is a random variable uniformly distributed in $\{0,1\}^{\mathrm{poly}(n)}$.*

Clearly, the two formulations are equivalent; starting from the weaker soundness condition, one may first reduce the soundness error to $\frac{1}{3} \cdot 2^{-n}$ (by repetitions), and next apply a standard counting argument. Every language in $\mathcal{NP}$ has a non-interactive proof system (in which no randomness is used). However, this NP-proof system is unlikely to be zero-knowledge.

The definition of zero-knowledge for the non-interactive model gets simplified since we need only consider one verifier. Actually, we can avoid considering the verifier at all.

Definition 2 (non-interactive zero-knowledge): *A non-interactive proof system, (P, V), for a language L is* zero-knowledge *if there exists a probabilistic polynomial-time algorithm M such that the ensembles $\{(x, R_{|x|}, P(x, R_{|x|}))\}_{x \in L}$ and $\{M(x)\}_{x \in L}$ are computationally indistinguishable, where R_n is a random variable uniformly distributed in $\{0,1\}^{\mathrm{poly}(n)}$.*

6.10.2 Constructions

A fictitious abstraction which is nevertheless very helpful for constructing non-interactive zero-knowledge proof systems is the *hidden bits model*. In this model the common random-string is uniformly selected as before, but only the prover can see all of it. The 'proof' he sends the verifier consists of two parts; a 'certificate' and the specification of some bit positions in the common random-string. The verifier may only inspect the bits of the common random-string residing in the locations which have been specified by the prover. Certainly, in addition, the verifier inspects the common input and the 'certificate'.

Definition 3 (proof systems in the Hidden Bits Model): *A pair of probabilistic machines, (P, V), is called a* hidden-bits proof system for *L if V is polynomial-time and the following two conditions hold*

- Completeness: *For every* $x \in L$

$$\text{Prob}\left(V(x,R_I,I,\pi)=1\right) \geq \frac{2}{3}$$

where $(I,\pi) \stackrel{\text{def}}{=} P(x,R)$*,* R *is a random variable uniformly distributed in* $\{0,1\}^{\text{poly}(|x|)}$ *and* R_I *is the sequence of bits at positions* $I \subseteq \{1,2,...,\text{poly}(|x|)\}$*. That is,* $R_I = r_{i_1}\cdots r_{i_t}$*, where* $R = r_1\cdots r_t$ *and* $I = (i_1,...,i_t)$*.*

- Soundness: *For every* $x \notin L$ *and every machine* B*,*

$$\text{Prob}\left(V(x,R_I,I,\pi)=1\right) \leq \frac{1}{3}$$

where $(I,\pi) \stackrel{\text{def}}{=} B(x,R)$*,* R *is a random variable uniformly distributed in* $\{0,1\}^{\text{poly}(|x|)}$ *and* R_I *is the sequence of bits at positions* $I \subseteq \{1,2,...,\text{poly}(|x|)\}$*.*

In both cases, I *is called the set of* revealed bits *and* π *is called the* certificate*. Zero-knowledge is defined as before, with the exception that we need to simulate* $(x,R_I,P(x,R))$ *rather than* $(x,R,P(x,R))$*, where* $(I,\pi) = P(x,R)$*.*

As hinted above, we do not suggest the Hidden-Bits Model as a realistic model. The importance of the model stems from two facts. Firstly, it is a 'clean' model which facilitates the design of proof systems (in it), and secondly that proof systems in the Hidden-Bits Model can be easily transformed into non-interactive proof systems (i.e., the realistic model). The transformation follows.

Construction 4 (from Hidden Bits proof systems to non-interactive ones): *Let* (P,V) *be a hidden-bits proof system for* L*,* $f:\{0,1\}^* \mapsto \{0,1\}^*$ *and* $b:\{0,1\}^* \mapsto \{0,1\}$*. Furthermore, let* $m = \text{poly}(n)$ *denote the length of the common random-string for common inputs of length* n *and suppose that* f *is 1-1 and length preserving. Following is a specification of a non-interactive system,* (P',V')*:*

- Common Input: $x \in \{0,1\}^n$.
- Common Random-String: $s = (s_1,...,s_m)$*, where each* s_i *is in* $\{0,1\}^n$.
- Prover (denoted P'):
  - *computes* $r_i = b(f^{-1}(s_i))$*, for* $i = 1,2,...,m$*.*
  - *invokes* P *to get* $(I,\pi) = P(x,r_1\cdots r_m)$*.*
  - *outputs* (I,π,p_I)*, where* $p_I \stackrel{\text{def}}{=} (f^{-1}(s_{i_1})\cdots f^{-1}(s_{i_t}))$ *for* $I = (i_1,...,i_t)$*.*
- Verifier (denoted V') *given prover's output* $(I,\pi,(p_1\cdots p_t))$*:*
  - *checks that* $s_{i_j} = f(p_j)$*, for each* $i_j \in I$*. In case a mismatch is found,* V' *rejects.*

- *computes* $r_i = b(p_i)$, *for* $i = 1, ..., t$. *Let* $r = r_1, ..., r_t$.
- *invokes* V *on* (x, r, I, π) *and accepts if and only if* V *accepts.*

Proposition 5 *Let* (P, V)*,* L*,* f*,* b *and* (P', V') *be as in Construction 4. Then,* (P', V') *is a non-interactive proof system for* L*, provided that* $\text{Prob}(b(U_n) = 1) = \frac{1}{2}$*. Furthermore, if* P *is zero-knowledge and* b *is a hard-core of* f *then* P' *is zero-knowledge too.*

We remark that P' is not perfect zero-knowledge even in case P is. Also, P' may not be implemented in polynomial-time (even with help of auxiliary inputs) even if P is (see Remark 6 below).

proof: To see that (P', V') is a non-interactive proof system for L we note that uniformly chosen $s_i \in \{0,1\}^n$ induce uniformly distributed bits $r_i \in \{0,1\}$. (This follows by $r_i = b(f^{-1}(s_i))$, the fact that f is one-to-one, and the fact that $b(f^{-1}(U_n)) = b(U_n)$ is unbiased.) Note that in case b is a hard-core of f, it is almost unbiased (i.e., $\text{Prob}(b(U_n)=1) = \frac{1}{2} \pm \frac{1}{\mu(n)}$, where μ is a negligible function). Thus, saying that b is a hard-core for f essentially suffices.

To see that P' is zero-knowledge note that we can convert an efficient simulator for P into an efficient simulator for P'. Specifically, for each revealed bit of value σ we uniformly select a string $r \in \{0,1\}^n$ so that $b(r) = \sigma$ and put $f(r)$ in the corresponding position in the common random-string. For each *un*revealed bit we uniformly select a string $s \in \{0,1\}^n$ and put it in the corresponding position in the common random-string. Using the fact that b is a hard-core of f, it follows that the simulator's output is computationally indistinguishable from the verifier's view. ■

Remark 6 (efficient implementation of P'): As stated above, in general P' cannot be efficiently implemented given a black-box access to P. What is needed is ability (of P') to invert f, however for P' to be zero-knowledge f must be one-way. The oblivious solution is to use a family of trapdoor permutations and let the prover know the trapdoor. Furthermore, the family should have the property that its members can be efficiently recognized (i.e., given a description of a function one can efficiently decide whether it is in the family). In other words, P' starts by selecting a permutation f over $\{0,1\}^n$ so that it knows its trapdoor, and proceeds as in Construction 4, except that it also appends f to the 'proof'. The verifier acts as in Construction 4 with respect to the function f specified in the proof. In addition it also checks that f is indeed in the family. Both the completeness and the zero-knowledge conditions follow exactly as in the proof of Proposition 5. For the soundness condition we need to consider all possible members of the family (w.l.o.g., there are at most 2^n such permutation). For each such permutation, the argument is as before and our claim thus follows by a counting argument. (Actually, we need also to repeat the (P, V) system for $O(n)$ times to make the counting argument work.)

We now turn to the construction of proof systems in the Hidden Bits model. Specifically, we are going to construct a proof system for the *Hamiltonian Cycle* (HC) problem which is NP-complete (and thus get proof systems for any language in $\mathcal{NP}$). We consider directed graphs (and the existence of directed Hamiltonian cycles). Below, we present a basic zero-knowledge system in which Hamiltonian graphs are accepted with probability 1 whereas non-Hamiltonian graphs on n vertices are rejected with probability $\Omega(n^{-3/2})$.

Construction 7 (Hidden Bits systems for HC):

- Common Input: *a directed graph* $G = (V, E)$ *with* $n \stackrel{\text{def}}{=} |V|$.
- Common Random-String: *viewed as an* n^3*-by-*n^3 *Boolean matrix* M*, with each entry being 1 with probability* n^{-5}.
  (This is implemented by breaking the common random-string to blocks of length $5 \log_2 n$ and setting a matrix entry to 1 iff the corresponding block is all 1's.)
- Definitions: *A* permutation matrix *is a matrix in which each row* (resp., column) *contains a single entry of value 1. A* Hamiltonian matrix *is a permutation matrix which corresponds to a single directed cycle* (it follows that the corresponding directed graph consists of a single Hamiltonian cycle)*. An* n^3*-by-*n^3 *matrix* M *is called* useful *if it contains an* n*-by-*n *Hamiltonian submatrix and all other entries in* M *are 0.*
- Prover: *Let* C *be a Hamiltonian cycle in* G*, in case such exists.*

  case 1: M *is useful. Let* H *denote its Hamiltonian* n*-by-*n *submatrix.*

  - *the prover reveals all entries in* M *which are not in* H.
  - *the prover finds a 1-1 mapping,* π_1*, of* V *to the rows of* H *and a 1-1 mapping,* π_2*, of* V *to the columns of* H *so that the edges of* C *are mapped to the 1-entries of* H.
    (Directed pairs of vertices of G, being edges or not, are mapped in the natural manner; that is (u, v) is mapped to the matrix entry $(\pi_1(u), \pi_2(v))$. The mapping-pair (π_1, π_2) is an isomorphism of C to H. Actually, we should specify one isomorphism among the n possible ones.)
  - *the prover reveals the entries corresponding to non-edges of* G.
    (The correspondence is by the above mappings.)
  - *the prover outputs the mapping pair* (π_1, π_2) (as a certificate).

  case 2: M *is not useful. In this case the prover reveals all entries of* M.

- Verifier:

  case 1: *The prover has not revealed all entries in* M*. Let* (π_1, π_2) *be the certificate sent/output by the prover. The verifier checks that all entries in* M *which do not have an image under* (π_1, π_2) *in* E *are revealed and are indeed zero. That is, the verifier accepts if all matrix entries, except for the entries in* $\{(\pi_1(u), \pi_2(v)) : (u, v) \in E\}$*, are revealed and all revealed bits are 0.*

case 2: *The prover has revealed all of M. In this case the verifier accepts iff M is* NOT *useful.*

The following fact is instrumental for the analysis of Construction 7.

Fact 6.10: Prob(M is useful) $= \Omega(n^{-3/2})$.

proof: With probability $\Omega(1/\sqrt{n})$, the matrix M contains exactly n entries of value 1. Considering any row of M, observe that with probability at most $\binom{n^3}{2} \cdot (n^{-5})^2 < n^{-4}$ this row contains more than a single 1-entry. Thus, with probability $\Omega(1/\sqrt{n})$, the matrix M contains an n-by-n permutation matrix and all its other entries are 0. The fact follows by observing that there are $n!$ (n-by-n) permutation matrices and $(n-1)!$ of them are Hamiltonian matrices. ■

Proposition 8 *There exists a* (perfect) *zero-knowledge Hidden Bits proof system for Graph Hamiltonicity. Furthermore, the prover may be implemented by a polynomial-time machine which gets an Hamiltonian cycle as auxiliary input.*

proof: We start by demonstrating a gap in the acceptance probability of the verifier of Construction 7. Firstly, we claim that in case G is Hamiltonian and the prover follows the program then the verifier accepts no matter which matrix M appears as common random-string. The claim follows easily by observing that in Case 1 the mapping-pair maps the Hamiltonian cycle of G to the Hamiltonian cycle of H and, since the latter contains the only 1-entries in M, all non-edges of G are mapped to 0-entries of M. (In Case 2 the claim is trivial.) We remark that the prover's actions can be implemented in polynomial-time when given an Hamiltonian cycle of G as auxiliary input. Specifically, all that the prover needs to do is check if M is useful and find an isomorphism between two given n-vertex cycles.

Next, suppose that G is non-Hamiltonian. By Fact 6.10, with probability at least $\Omega(n^{-3/2})$, the matrix M is useful and let H denote its n-by-n Hamiltonian submatrix. In this case the prover must reveal all entries not in the submatrix H since mapping $V \times V$ to any other n-by-n submatrix of M will reveal 1-entries (in the rest of M). Thus, the prover must output a mapping pair (π_1, π_2) so that $\pi_1(V) \times \pi_2(V) = H$. Also, each non-edge of G must be mapped to a 0-entry of H (or else the verifier will reject). It follows that the preimage of each 1-entry in H must be an edge in G, which implies that G has a Hamiltonian cycle (in contradiction to our hypothesis). We conclude that in case G is non-Hamiltonian, it is rejected with probability $\Omega(n^{-3/2})$.

Finally, we show that the above prover is zero-knowledge. This is done by constructing a simulator that on input a graph G randomly selects an n^3-by-n^3 matrix, denoted M, with distribution as in the common random-string (i.e., each entry being 1 with probability n^{-5}). If M is not useful then the simulator outputs $(G, M, \{1, ..., n^3\}^2)$ (i.e., all bits are revealed with values as in M and no certificate is given). Otherwise, the prover selects uniformly a pair of 1-1 mappings (π_1, π_2) so that $\pi_i : V \mapsto \{1, ..., n^3\}$, for $i = 1, 2$. The prover outputs $(G, 0^{n^6 - |E|}, I, (\pi_1, \pi_2))$, where $I \stackrel{\rm def}{=} \{1, ..., n^3\}^2 - \{(\pi_1(u), \pi_2(v)) : (u, v) \in E\}$. The reader can easily verify that the output distribution of the simulator is identical to the distribution seen by the verifier. ■

Using Propositions 8 and 5 and Remark 6, we conclude

Theorem 9 *Assuming the existence of one-way permutations, each language in $\mathcal{NP}$ has a zero-knowledge non-interactive proof system. Furthermore, assuming the existence of families of trapdoor permutations for which membership in the family can be decided in $\mathcal{BPP}$, each language in $\mathcal{NP}$ has a zero-knowledge non-interactive proof system in which the prover can be implemented by a probabilistic polynomial-time machine which gets an NP-witness as auxiliary input.*

6.10.3 Extensions: many assertions of varying length

The definitions presented in Section 6.10.1 are restricted in two ways. Firstly, they consider only the proving of one assertion relative to the common random-string, and furthermore the common random-string is allowed to be longer than the assertion (though polynomial in length of the assertion). A stronger definition provided below allows proving $\mathrm{poly}(n)$-many assertions, each of $\mathrm{poly}(n)$-length, using the same n-bit long common random-string.

We first note that it suffices to treat the case in which the number of assertions is unbounded but the length of each assertion is a-priori bounded. Specifically, for any $\varepsilon > 0$, it suffices to consider the case where $\mathrm{poly}(n)$-many assertions, each of length n^ε, need to be proven relative to the same n-bit long common random-string. The reason for this is that we can reduce, in a "zero-knowledge manner", any NP-assertion of length $\mathrm{poly}(n)$ into a sequence of $\mathrm{poly}(n)$-many NP-assertions, each of length n^ε. For example, first we reduce the original NP-assertion to an assertion regarding the 3-colorability of a $\mathrm{poly}(n)$-vertex graph. Next, we use a commitment scheme with commitments of length n^ε, in order to commit to the coloring of each vertex. Finally, for each edge, we (invoke the proof system to) prove that the corresponding two commitments are to two different values in $\{1,2,3\}$.

We now turn to the actual definitions. First we note that nothing needs to be changed regarding the definition of non-interactive proof systems (Definition 1). We still require ability to be convinced by valid assertions and "protection" from false assertions. Alas a minor technical difference is that, while in Definition 1 we have denoted by n the length of the assertion and considered a common random-string of length $\mathrm{poly}(n)$, here we let n denote the length of the common random-string used for assertions of length n^ε. We call ε the *fundamental constant* of the proof system. In contrast, the definition of zero-knowledge has to be extended to handle a sequence of proofs.

Definition 10 (non-interactive zero-knowledge – extended): *A non-interactive proof system, (P, V), with fundamental constant ε, for a language L is* strongly zero-knowledge *if there exists a probabilistic polynomial-time algorithm M such that the ensembles*

$$\{((x_1, ..., x_m), U_n, (P(x_1, U_n), ..., P(x_m, U_n)))\}_{x_1,...,x_m \in L_{n^\varepsilon}} \quad \text{and} \quad \{M(x_1, ..., x_m)\}_{x_1,...,x_m \in L_{n^\varepsilon}}$$

are computationally indistinguishable, where $m = \mathrm{poly}(n)$ and $L_\ell \stackrel{\text{def}}{=} L \cap \{0,1\}^\ell$.

We now turn to the construction of strong zero-knowledge (non-interactive) proof systems. The underlying idea is to facilitate the simulation by potentially proving a fictitious assertion regarding a portion of the common random-string. The assertion that will be potentially proven about this portion will have the following properties

1. The assertion holds for a negligible fraction of the strings of the same length. Thus, adding this potential ability does not significantly effect the soundness condition.
2. Strings satisfying the assertion are computationally indistinguishable from uniformly distributed strings of the same length. Thus, it will be OK for the simulator to use such strings rather than uniformly chosen ones (used in the real proof system).
3. The decision problem for the assertion is in $\mathcal{NP}$. This will allow a reduction to an NP-complete problem.

An immediate assertion, concerning strings, which comes to mind is being produced by a pseudorandom generator.

Construction 11 (strong zero-knowledge non-interactive proof systems): *Let $G : \{0,1\}^\ell \mapsto \{0,1\}^{2\ell}$, L_1 be an NP-complete language, and (P,V) be a non-interactive proof system for L_1. Furthermore, suppose that (P,V) uses a common random-string of length $n-2\ell$ for assertions of length* $\mathrm{poly}(\ell)$ *and that P takes as auxiliary input an NP-witness for membership in L_1. Following is a specification of a non-interactive system for $L \in \mathcal{NP}$:*

- Common Input: $x \in \{0,1\}^\ell$.
- Common Random-String: $r = (p,s)$, *where* $p \in \{0,1\}^{2\ell}$ *and* $s \in \{0,1\}^{n-2\ell}$.
- Prover:
  - *Using a standard reduction of L_2 to L_1, reduces (x,p) to $y \in \{0,1\}^{\mathrm{poly}(\ell)}$, where*

    $$L_2 \stackrel{\text{def}}{=} \{(x,p) : x \in L \bigvee \exists w \in \{0,1\}^{|x|} \;\; s.t. \;\; G(w) = p\}$$

    In case the prover is given a witness u for $x \in L$, it reduces u to a witness, denoted w, for $y \in L_1$.
  - *Invokes P with common input y, auxiliary input w and common random-string s, getting output π which it outputs/sends.*
- Verifier:
  - *Reduces (x,p) into y using the same standard reduction of L_2 to L_1.*
  - *Invokes V with common input y, common random-string s and prover's output π, and decide as V does.*

Proposition 12 *Let (P, V) be as above, G be a pseudorandom generator, and* $\text{poly}(\ell) = n^{\varepsilon}$*. Furthermore, suppose that P is zero-knowledge and that when given an NP-witness as auxiliary input it can be implemented in probabilistic polynomial-time. Then, Construction 11 constitutes a zero-knowledge non-interactive proof system for L, with fundamental constant ε. Furthermore, the prover may be implemented by a probabilistic polynomial-time machine which gets an NP-witness as auxiliary input.*

proof sketch: The completeness and efficiency claims for the new prover follow immediately from the hypothesis concerning (P, V). The soundness condition follows by observing that the probability that p is in the range of G is at most $2^{-\ell}$. To prove the zero-knowledge property, we construct a simulator as follows. The simulator uniformly selects $u' \in \{0,1\}^{\ell}$ and $s \in \{0,1\}^{n-2\ell}$, sets $p = G(u')$, and follows the prover's program except that it uses u' as the NP-witness for $(x, p) \in L_2$. Namely, the simulator reduces $(x, p) \in L_1$ to $y \in L_1$ along with reducing the NP-witness u' to a witness w' (for y). Next, the simulator invokes P with common input y, auxiliary input w' and common random-string s. Note that the efficiency of the simulator relies on the efficient implementation of P. To prove that the simulator's output is computationally indistinguishable from the verifier's view we combine the following two observations:

1. The distribution of the common random-string is very different in the two cases. Yet, by the pseudorandomness of G this difference is computationally indistinguishable. Thus, we may consider the verifier's view in case the common random-string is selected exactly as in the simulation (but the prover acts as in Construction 11).

2. The zero-knowledge property of P implies that P is witness-indistinguishable (see Section **??**). Thus, one cannot distinguish the case P uses a witness for $x \in L$ (as in Construction 11) from the case P uses as witness a seed for the pseudorandom sequence p (as done in the simulator). The same holds when repeating the process polynomially-many times.

■

Using Theorem 9 and Proposition 12, we obtain

Theorem 13 *Assuming the existence of families of trapdoor permutations for which membership in the family can be decided in $\mathcal{BPP}$, each language in $\mathcal{NP}$ has a* strong *zero-knowledge non-interactive proof system. Furthermore, the prover can be implemented by a probabilistic polynomial-time machine which gets an NP-witness as auxiliary input.*

Bibliographical Notes

Non-interactive zero-knowledge proof systems were introduced by Blum, Feldman and Micali. The constructions presented in Section 6.10 are due to Feige, Lapidot and Shamir [FLSnizk].

Author's Note: *BFM has appeared in STOC88, and FLSnizk in FOCS90*

Multi-prover interactive proofs were introduced by Ben-Or, Goldwasser, A much more efficient construction of non-interactive proof systems for $\mathcal{NP}$, based on the same assumptions as [FLSnizk], has appeared in a paper of Kilian and Petrnak [KP96].

Author's Note: *KP96 has appeared in ECCC and will appear in Jour of Cryptology.*