Basing Non-Interactive Zero-Knowledge on (Enhanced) Trapdoor Permutations: The State of the art

Oded Goldreich[*]
Department of Computer Science
Weizmann Institute of Science
Rehovot, ISRAEL.
oded@wisdom.weizmann.ac.il

October 6, 2009

Abstract

The purpose of this note is to correct the inaccurate account of this subject that is provided in our two-volume work *Foundation of Cryptography.* Specifically, as pointed out by Jonathan Katz, it seems that the construction of Non-Interactive Zero-Knowledge proofs for $\mathcal{NP}$ requires the existence of a doubly-enhanced collection of trapdoor permutations (to be defined below). We stress that the popular candidate collections of trapdoor permutations do satisfy this doubly-enhanced condition. In fact, any collection of trapdoor permutations that has dense and easily recognizable domain satisfies this condition.

1 Introduction

The purpose of this note is to correct the inaccurate account of the construction of Non-Interactive Zero-knowledge proofs (NIZK) for $\mathcal{NP}$ that is provided in [G1, Sec. 4.10.2] and modified in [G2, Apdx. C.4.1]. We briefly recall the relevant facts.

In [G1, Rem. 4.10.6], a construction of NIZK for $\mathcal{NP}$ is sketched based on a collection of trapdoor permutations in which each permutation f_α has domain $\{0,1\}^{|\alpha|}$. This description is correct, but the problem is with the unsupported claim (at the end of [G1, Rem. 4.10.6]) by which the construction can be extended to arbitrary collections of trapdoor permutations (in which the domain of the permutation f_α may be a sparse subset of $\{0,1\}^{|\alpha|}$ and may not be easy to recognize (although it is easy to sample from)).

In [G2, Apdx. C.4.1] it was claimed that such a construction (of NIZK for $\mathcal{NP}$) can be obtained based on any *enhanced* collections of trapdoor permutations, where the enhancement is as defined in [G2, Apdx. C.1]. But again, this claim was not fully supported. Furthermore, as pointed out by Jonathan Katz, it seems that this construction requires an additional enhancement. In this note we define the resulting notion of a *doubly-enhanced* collection of trapdoor permutations, and provide full details to the claim that using such permutations one can construct NIZK for $\mathcal{NP}$. We stress that the popular candidate collections of trapdoor permutations do satisfy this doubly-enhanced condition. In fact, any collection of trapdoor permutations that has dense and easily recognizable domain satisfies this condition. More generally, if the domain-sampler S' of an enhanced collection

[*]Partially supported by the Israel Science Foundation (grant No. 1041/08).

of trapdoor permutations has a "reversed sampler" (which given α, y generates a random r such that $S'(\alpha, r) = y$), then this collection is doubly-enhanced.

On the non-technical level, we believe that this unfortunate line of events demonstrates the importance of not being tempted by hand-waving arguments and working out detailed proofs. Indeed, we believe that the source of trouble is that the basic idea is presented in [G1, Rem. 4.10.6] as a patch, and further modifications are also presented as patches (see [G2, Apdx. C.4.1]).

2 Background

In this section we recall the standard definition of non-interactive zero-knowledge proof systems as well as the construction of such systems based on proof systems in the *hidden-bits model*. Since proof systems for $\mathcal{NP}$ in the hidden-bits model are known to exists (unconditionally, see [G1, Sec. 4.10.2]), our focus in this note is on transforming such systems into standard NIZK systems. We stress that intractability assumptions are used in the latter transformation.

The rest of this section is essentially reproduced from [G1, Sec. 4.10.1&4.10.2].

2.1 The Basic Definition

Recall that the model of non-interactive (zero-knowledge) proof systems consists of three entities: a prover, a verifier and a uniformly selected sequence of bits (which can be thought of as being selected by a trusted third party). Both verifier and prover can read the random sequence, and each can toss additional coins. The interaction consists of a single message sent from the prover to the verifier, who is then left with the decision (whether to accept or not). Here we present only the basic definition that supports the case of proving a single assertion of a-priori bounded length. Various extensions are presented in [G1, Sec. 4.10.3] and in [G2, Sec. 5.4.4.4]; we recall that the construction of such stronger NIZKs can be reduced to the construction of basic NIZKs (as defined below).

The model of non-interactive proofs seems closer in spirit to the model of NP-proofs than to general interactive proofs. In a sense, the NP-proof model is extended by allowing the prover and verifier to refer to a common random string, as well as toss coins by themselves. Otherwise, as in case of NP-proofs, the interaction is minimal (i.e., unidirectional: from the prover to the verifier). Thus, in the definition below both the prover and verifier are ordinary probabilistic machines that, in addition to the common-input, also get a uniformly distributed (common) *reference-string*. We stress that, in addition to the above common input and common reference-string, both the prover and verifier may toss coins and get auxiliary inputs. However, for sake of simplicity, we present a definition for the case in which none of these machines gets an auxiliary input (yet, they may both toss additional coins). 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 $x \in L$*

$$\Pr[V(x, R, P(x, R)) = 1] \geq \frac{2}{3}$$

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

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

$$\Pr[V(x, R, B(x, R)) = 1] \leq \frac{1}{3}$$

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

The uniformly chosen string R is called the common reference-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 process sufficiently many times (using a sequence of many independently chosen reference-strings). In stating the soundness condition, we have deviated from the standard formulation that allows $x \notin L$ to be adversarially selected after R is fixed; the latter "adaptive" formulation of soundness is used in [G1, Sec. 4.10.3], and it is easy to transform a system satisfying the above ("non-adaptive") soundness condition into one satisfying the adaptive soundness condition (see [G1, Sec. 4.10.3]).

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 (as defined next). The definition of zero-knowledge for the non-interactive model is simplified by the fact that, since the verifier cannot affect the prover's actions, it suffices to consider the simulatability of the view of a single verifier (i.e., the prescribed one). Actually, we can avoid considering the verifier at all (since its view can be generated from the common reference-string and the message sent by the prover).

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

This definition too is "non-adaptive" (i.e., the common input may not depend on the common reference-string). An adaptive formulation of zero-knowledge is presented and discussed in [G1, Sec. 4.10.3].

2.2 The Hidden-Bits Model

A fictitious abstraction, which is nevertheless very helpful for the design of non-interactive zero-knowledge proof systems, is the *hidden bits model*. In this model the common reference-string is uniformly selected as before, but only the prover can see all of it. The 'proof' that the prover sends to the verifier consists of two parts; a 'certificate' and the specification of some bit positions in the common reference-string. The verifier may only inspect the bits of the common reference-string that reside in the locations that have been specified by the prover. Needless to say, 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 $x \in L$*

$$\Pr[V(x, R_I, I, \pi) = 1] \geq \frac{2}{3}$$

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 substring of R 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 algorithm B,*

$$\Pr[V(x,R_I,I,\pi)=1] \leq \frac{1}{3}$$

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 substring of R 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* $(x,R_I,P(x,R)) = (x,R_I,I,\pi)$ *rather than* $(x,R,P(x,R))$*.*

As stated 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 proof systems in the Hidden-Bits Model can be easily transformed into non-interactive proof systems (i.e., the realistic model). The transformation (which utilizes a one-way permutation f with hard-core b) follows.

Construction 4 (from Hidden Bits proof systems to non-interactive ones): *Let (P,V) be a hidden-bits proof system for L, and suppose that* $f:\{0,1\}^* \to \{0,1\}^*$ *and* $b:\{0,1\}^* \to \{0,1\}$ *are polynomial-time computable. Furthermore, let* $m = \text{poly}(n)$ *denote the length of the common reference-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 Reference-String: $s = (s_1,...,s_m)$*, where each s_i is in* $\{0,1\}^n$*.*
- Prover (denoted P'):
  1. *Computes* $r_i = b(f^{-1}(s_i))$*, for* $i = 1,2,...,m$*.*
  2. *Invokes P to obtain* $(I,\pi) = P(x, r_1 \cdots r_m)$*.*

  P' outputs (I,π,p_I)*, where* $p_I \stackrel{\text{def}}{=} (f^{-1}(s_{i_1}) \cdots f^{-1}(s_{i_t}))$ *for* $I = (i_1,...,i_t)$*.*

  That is, P' augments the proof (I,π), obtained from P, with the f-preimages of the blocks of the reference string that are indicated in I. These preimages reveal the values of the corresponding "revealed" bits in the hidden-bits model, while the values of the other bits remain essentially hidden.

- Verifier (denoted V') *given prover's output* $(I,\pi,(p_1 \cdots p_t))$*:*
  1. *Checks that* $s_{i_j} = f(p_j)$*, for each* $i_j \in I$*.*
     In case a mismatch is found, V' halts and rejects.
  2. *Computes* $r_i = b(p_i)$*, for* $i = 1,...,t$*. Let* $r = r_1,...,r_t$*.*
  3. *Invokes V on* (x,r,I,π)*, and* accepts *if and only if V accepts.*

That is, using the p_j's, the verifier V' reconstructs the the values of the corresponding "revealed" bits in the hidden-bits model, and invokes V on these values.

We comment that P' is not perfect (or statistical) zero-knowledge even in case P is. Furthermore (and more central to this note), the prover P' may not be implemented in polynomial-time even if P is (and even with the help of auxiliary inputs). See further discussion in next section.

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* $\Pr[b(U_n)=1]=\frac{1}{2}$. *Furthermore, if P is zero-knowledge and b is a hard-core of f then P' is zero-knowledge too.*

Proof: To see that (P',V') is a non-interactive proof system for L we note that uniformly chosen strings $s_i \in \{0,1\}^n$ 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 $b(f^{-1}(U_n)) \equiv b(U_n)$ is unbiased. Thus, the actions of P' and V' perfectly emulate the actions of P and V, respectively.

Note that if b is a hard-core of f, then b is almost unbiased (i.e., $\Pr[b(U_n)=1] = \frac{1}{2} \pm \mu(n)$, where μ is a negligible function), and the said emulation is only guaranteed to be almost-perfect (i.e., deviates negligibly from the original). Thus, saying that b is a hard-core for f essentially suffices for concluding that (P',V') is a non-interactive proof system for L.

To see that P' is zero-knowledge note that we can convert an efficient simulator for P into an efficient simulator for P'. Specifically, we first invoke the P-simulator and obtain the sequence of revealed bits, which correspond to the set I, as well as the simulated certificate, denoted π. Next, for each revealed bit of value σ, we uniformly select a string $r \in \{0,1\}^n$ such that $b(r)=\sigma$ and put $f(r)$ in the corresponding position in the common reference-string. That is, if the said bit corresponds to position $i \in I$, then we set the i^{th} block of the reference string to $f(r)$. For each *un*revealed bit (i.e., bit position $i \notin I$), we uniformly select a string $s \in \{0,1\}^n$ and put it in the corresponding position in the common reference-string (i.e., set the i^{th} block of the reference string to s). The output of the P'-simulator consists of the common reference-string generated as above, the sequence of all r's generated by the P'-simulator for bits revealed by the P-simulator (i.e., bit in I), and the pair (I,π) as in the output of the P-simulator. Following is a rigorous description of the P'-simulator, when invoked on input $x \in \{0,1\}^n$ and using the P-simulator, denoted M.

1. Obtain $(x,(\sigma_1,...,\sigma_t),(i_1,...,i_t),\pi) \leftarrow M(x)$.
2. For every $j=1,..,t$, select uniformly $p_j \in \{0,1\}^n$ such that $b(p_j)=\sigma_j$ and set $s_{i_j}=f(p_j)$.
3. For every $i \in [m] \setminus \{i_j : j=1,..,t\}$, select s_i uniformly in $\{0,1\}^n$.
4. Output $(x,(s_1,...,s_m),((i_1,...,i_t),\pi,(p_1,...,p_t)))$.

   That is $(s_1,...,s_m)$ is the simulated "common reference string" whereas $((i_1,...,i_t),\pi,(p_1,...,p_t))$ is the simulated proof.

Using the hypothesis that b is a hard-core of f, it follows that the output of the P'-simulator is computationally indistinguishable from the verifier's view (when receiving a proof from P'). Note that the only difference between the simulation and the real view is that in the former the values on the actual reference strings do not necessarily match the values of the corresponding hidden bits seen by P. However, this difference is computationally indistinguishable (by the hypothesis that b is a hard-core of f). ■

3 Efficient Implementations of the Prover of Construction 4

As hinted above, in general, P' may not be efficiently implemented given black-box access to P. What is needed for such an efficient implementation is the ability (of P') to invert f. On the other hand, for P' to be zero-knowledge f must be one-way. The obvious solution is to use a collection of trapdoor permutations and let the prover know the trapdoor.

The basic construction is presented based on a collection of trapdoor permutations that have simple domains (i.e., the domain of each permutation is the set of all strings of some fixed string). Furthermore, the collection 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 collection).

3.1 The basic construction

Using such a collection of trapdoor permutations, P' starts by selecting a permutation f over $\{0,1\}^n$ such that it knows its trapdoor, and proceeds as in Construction 4, except that it also appends the description of f to the 'proof'. Indeed, the knowledge of the trapdoor allows P' to invert f on any element in f's domain. The verifier acts as in Construction 4 with respect to the function f specified in the proof. In addition the verifier also checks that f is indeed in the collection.

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 collection (w.l.o.g., there are at most 2^n such permutations). For each such permutation, the argument is as before, and our soundness claim thus follows by a counting argument (as applied in [G1, Sec. 4.10.3]). Actually, we also need to repeat the (P,V) system for $O(n)$ times, so to first reduce the soundness error to $\frac{1}{3} \cdot 2^{-n}$.

The foregoing text is reproduced from [G1, Rem. 4.10.6] and is indeed valid. The only problem is that it refers to a restricted notion of a collection of trapdoor permutations. Specifically, when compared with the general definition of such collections (as provided in [G1, Def. 2.4.5]), the foregoing description corresponds to the special case in which for every index α the domain of the permutation f_α (i.e., the permutation described by α) equals $\{0,1\}^{|\alpha|}$. In contrast, in general, the domain of f_α may be an arbitrary subset of $\{0,1\}^{|\alpha|}$ (as long as this subset is easy to sample from). The focus of this note is on trying to extend the foregoing construction by using more general forms of trapdoor permutations.

3.2 Extending the basic construction

We start by recalling the definition of a collection of trapdoor permutations, and considering a couple of enhancements.

3.2.1 Enhanced collections of trapdoor permutations

Recall that a collection of trapdoor permutations, as defined in [G1, Def. 2.4.5], is a collection of finite permutations, denoted $\{f_\alpha : D_\alpha \to D_\alpha\}$, accompanied by four probabilistic polynomial-time algorithms, denoted I, S, F and B (for *index*, *sample*, *forward* and *backward*), such that the following (syntactic) conditions hold:

1. On input 1^n, algorithm I selects a random n-bit long index α of a permutation f_α, along with a corresponding trapdoor τ;

2. On input α, algorithm S *samples* the domain of f_α, returning an almost uniformly distributed element in it;

3. For x in the domain of f_α, given α and x, algorithm F returns $f_\alpha(x)$ (i.e., $F(\alpha, x) = f_\alpha(x)$);

4. For y in the range of f_α if (α, τ) is a possible output of $I(1^n)$ then, given τ and y, algorithm B returns $f_\alpha^{-1}(y)$ (i.e., $B(\tau, y) = f_\alpha^{-1}(y)$).

The hardness condition in [G1, Def. 2.4.5] refers to the difficulty of inverting f_α on a uniformly distributed element of its range, when given only the range-element and α. That is, letting $I_1(1^n)$ denote the first element in the output of $I(1^n)$ (i.e., the index), it is required that for every probabilistic polynomial-time algorithm A (resp., every non-uniform family of polynomial-size circuit $A = \{A_n\}_n$), every positive polynomial p and all sufficiently large n's it holds that

$$\Pr[A(I_1(1^n), f_{I_1(1^n)}(S(I_1(1^n))) = S(I_1(1^n))] \;<\; \frac{1}{p(n)}_. \tag{1}$$

Namely, A (resp., A_n) fails to invert f_α on $f_\alpha(x)$, where α and x are selected by I and S as above. An equivalent way of writing Eq. (1) is

$$\Pr[A(I_1(1^n), S'(I_1(1^n), R_n)) = f^{-1}_{I_1(1^n)}(S'(I_1(1^n), R_n))] \;<\; \frac{1}{p(n)}_. \tag{2}$$

where S' is the residual two-input (deterministic) algorithm obtained from S when treating the coins of the latter as an auxiliary input, and R_n denote the distribution of the coins of S on n-bit long inputs. That is, A fails to invert f_α on x, where α and x are selected as above.

Enhanced trapdoor permutations. Although the above definition suffices for many applications, in some cases we will need an enhanced hardness condition. Specifically, we will require that it is hard to invert f_α on a random input x (in the domain of f_α) *even when given the coins used by S in the generation of x.* (Note that given these coins (and the index α), the resulting domain element x is easily determined.)

Definition 6 (enhanced trapdoor permutations): *Let $\{f_\alpha : D_\alpha \to D_\alpha\}$ be a collection of trapdoor permutations as in* [G1, Def. 2.4.5]. *We say that this collection is* enhanced (and call it an enhanced collection of trapdoor permutations) *if for every probabilistic polynomial-time algorithm A every positive polynomial p and all sufficiently large n's*

$$\Pr[A(I_1(1^n), R_n) = f^{-1}_{I_1(1^n)}(S'(I_1(1^n)), R_n))] \;<\; \frac{1}{p(n)} \tag{3}$$

where S' is as above. The non-uniform version is defined analogously.

Note that the special case of [G1, Def. 2.4.5] in which the domain of f_α equals $\{0,1\}^{|\alpha|}$ satisfies Definition 6 (because, without loss of generality, the sampling algorithm may satisfy $S'(\alpha, r) = r$). This implies that modified versions of the RSA and Rabin collections satisfy Definition 6. More natural versions of both collections can also be shown to satisfy Definition 6. For further discussion see the Appendix.

Doubly-enhanced trapdoor permutations. Although collection of enhanced trapdoor permutations suffice for the construction of Oblivious Transfer (see [G2, Sec. 7.3.2]), it seems that they do not suffice for our current purpose of providing an efficient implementation of the prover of Construction 4.[1] Thus, we further enhance Definition 6 so to provide for such an implementation. Specifically, we will require that, given α, it is feasible to generate a random pair (x, r) such that r is uniformly distributed in $\{0,1\}^{\text{poly}(|\alpha|)}$ and x is a preimage of $S'(\alpha, r)$ under f_α; that is, we should generate a random $x \in D_\alpha$ along with coins that fit the generation of $f_\alpha(x)$ (rather than coins that fit the generation of x).

Definition 7 (doubly-enhanced trapdoor permutations): *Let* $\{f_\alpha : D_\alpha \to D_\alpha\}$ *be an enhanced collection of trapdoor permutations* (as in Def. 6). *We say that this collection is* doubly-enhanced (and call it a doubly-enhanced collection of trapdoor permutations) *if there exists a probabilistic polynomial-time algorithm that on input* α *outputs a pair* (x, r) *such that* r *is distributed identically to* $R_{|\alpha|}$ *and* $f_\alpha(x) = S'(\alpha, r)$.

We note that Definition 7 is satisfied by any collection of trapdoor permutations that has a reversed domain-sampler (i.e., a probabilistic polynomial-time algorithm that on input (α, y) outputs a string that is uniformly distributed in $\{r : S'(\alpha, r) = y\}$).

A useful relaxation of Definition 7 allows r to be distributed almost-identically (rather than identically) to $R_{|\alpha|}$, where by almost-identical distributions we mean that the corresponding variation distance is negligible (i.e., the distributions are statistically close). Needless to say, in this case the definition of a reversed domain-sampler should be relaxed accordingly.

We stress that adequate implementations of the popular candidate collections of trapdoor permutations (e.g., the RSA and Rabin collections) do satisfy the foregoing doubly-enhanced condition. In fact, any collection of trapdoor permutations that has dense and easily recognizable domain satisfies this condition. For further details see the Appendix.

3.2.2 Actually implementing the prover

Recall that the basic construction presented in Section 3.1 relies on two extra properties of the collection of trapdoor permutations.

1. It was assumed that the set of possible descriptions of the possible permutations, denoted $\overline{I}$, is easily recognizable (i.e., the support of $I(1^n)$ is recognizable in poly(n)-time).

2. It was assumed that the domain of every permutation f_α equals $\{0,1\}^{|\alpha|}$.

The first assumption was waived by Bellare and Yung [BY], and we briefly sketch their underlying idea first. This relaxation is crucial since no candidate collection of trapdoor permutations that satisfies this assumption is known (i.e., for all popular candidates, the corresponding index set $\overline{I}$ is not known to be efficiently recognizable).

The problem that arises is that the prover may select (and send) a function that is not in the collection (i.e., an index α that is not in $\overline{I}$). In such a case, the function is not necessarily 1-1, and, consequently, the soundness property may be violated. This concern can be addressed by using a (simple) non-interactive (zero-knowledge) proof for convincing the verifier that the function is "typically 1-1" (or, equivalently, is "almost onto the designated range"). The proof proceeds by

[1] We note that the enhancement of Definition 6 was intended to suffice for both purposes. Furthermore, in [G2, Apdx. C.4] it was claimed that enhanced trapdoor permutations do suffice for providing an efficient implementation of the prover of Construction 4. Needless to say, we retract this claim here.

presenting preimages (under the function) of random elements that are specified in the reference string. Note that, for any fixed polynomial p, we can only prove that the function is 1-1 on at least a $1-(1/p(n))$ fraction of the designated range (i.e., $\{0,1\}^n$), yet this suffices for moderate soundness of the entire proof system (which in turn can be amplified by repetitions). For further details, consult [BY].

Note that this solution extends to the case that the collection of permutations $\{f_\alpha : D_\alpha \to D_\alpha\}_{\alpha\in\overline{I}}$ does not satisfy $D_\alpha = \{0,1\}^{|\alpha|}$, but is rather an arbitrary collection of doubly-enhanced trapdoor permutations. In this case the reference string will contain a sequence of coin-sequences to be used by the domain-sampling algorithm (rather than consisting of elements of the function's domain). By virtue of the extra condition in Definition 7, we can simulate the inverting of each domain element by generating a pair (x,r), placing r on the reference string, and providing x as the inverse of $S'(\alpha,r)$ under f_α. (See an analogous discussion in next paragraph.)

We now turn to the second aforementioned assumption; that is, the assumption that the domain of f_α equals $\{0,1\}^{|\alpha|}$ (i.e., $D_\alpha = \{0,1\}^{|\alpha|}$). We would have liked to waive this assumption completely, but are only able to do so in the case that the collection of trapdoor permutations is *doubly-enhanced.* The basic idea is letting the reference string consist of coin-sequences to be used by the domain-sampling algorithm (rather than of elements of the function's domain). The corresponding domain elements, which depend on the choice of the index α, are then obtained by applying the domain-sampling algorithm to these coin-sequences. The enhanced hardness property (stated in Def. 6) is used in order to note that the corresponding preimages under f_α is not revealed by these coin-sequences, whereas the additional enhancement (stated in Def. 7) is used for arguing that revealing such preimages does not reveal additional knowledge. That is, the two additional properties (stated in Def. 6 and Def. 7) are used in the simulation and not in the proof system itself. For sake of simplicity, in the following exposition, we again use the (problematic) assumption by which $\overline{I}$ is efficiently recognizable.

Construction 8 (Construction 4, revised): *Let (P,V) be a zero-knowledge hidden-bits proof system for L with exponentially vanishing soundness error* (i.e., soundness error at most 2^{-n-2}), *and let $m = \text{poly}(n)$ denote the length of the common reference-string for common inputs of length n. Suppose that $\{f_\alpha : D_\alpha \to D_\alpha\}_{\alpha\in\overline{I}}$ is a doubly-enhanced collection of trapdoor permutations, where $\overline{I}$ is efficiently recognizable, and $b : \{0,1\}^* \to \{0,1\}$ is a corresponding hard-core predicate (i.e., $b(f_\alpha^{-1}(S(\alpha,U_\ell)))$ is infeasible to predict when given (α,U_ℓ)).*[2] *Following is a specification of a non-interactive system, (P',V'):*

- Common Input: $x \in \{0,1\}^n$.
- Prover's auxiliary input: w.
- Common Reference-String: $s = (s_1,...,s_m)$, *where each s_i is in $\{0,1\}^\ell$ and ℓ is the number of coins used by the domain-sampler when given an n-bit long index of a permutation.*
- Prover (denoted P'):
    1. *Select at random an n-bit long index α and a corresponding trapdoor τ; i.e., $(\alpha,\tau) \leftarrow I(1^n)$.*
    2. *Using the trapdoor τ, compute $r_i = b(f_\alpha^{-1}(S'(\alpha,s_i)))$, for $i = 1,2,...,m$.*

[2]Such a hard-core predicate is obtained by applying the techniques of [GL] (see [G1, Sec. 2.5.2] or better [G3, Sec. 7.1.3]) to any (doubly-)enhanced collection of trapdoor permutations.

   3. Invokes P to obtain $(I,\pi) = P(x, w, r_1 \cdots r_m)$.

  P' outputs (α, I, π, p_I), *where* $p_I \overset{\text{def}}{=} (f_\alpha^{-1}(S'(\alpha, s_{i_1})) \cdots f_\alpha^{-1}(S'(\alpha, s_{i_t})))$ *for* $I = (i_1, ..., i_t)$.

- Verifier (denoted V') *given prover's output* $(\alpha, I, \pi, (p_1 \cdots p_t))$:

   1. *Check if* $\alpha \in \overline{I}$*, otherwise* halts and rejects.
   2. *Check that* $S'(\alpha, s_{i_j}) = f_\alpha(p_j)$*, for each* $i_j \in I$.
      In case a mismatch is found, V' halts and rejects.
   3. *Compute* $r_i = b(p_i)$*, for* $i = 1, ..., t$*. Let* $r = r_1, ..., r_t$.
   4. *Invoke* V *on* (x, r, I, π)*, and* accepts *if and only if* V *accepts.*

Clearly, the foregoing strategy P' is efficient, provided that so is P.

Proposition 9 (Proposition 5, revised) *Let* (P,V)*,* L*,* f*,* b *and* (P',V') *be as in Construction 8. Then,* (P',V') *is a zero-knowledge non-interactive proof system for* L.

Proof: Following the proof of Proposition 5, we note that for any fixed choice $\alpha \in \overline{I} \cap \{0,1\}^n$ the soundness error is at most 2^{-n-2}. Taking a union bound over all possible $\alpha \in \overline{I} \cap \{0,1\}^n$ and discarding all $\alpha \notin \overline{I}$ (by virtue of the explicit check), we establish that (P',V') is a non-interactive proof system for L.

To show that P' is zero-knowledge we convert any (efficient) simulator for P into an (efficient) simulator for P'. First, the new simulator selects at random an index α (of a permutation) just as P' does. We stress that although P' obtains the corresponding trapdoor (just as P' does), we will not use this fact in the simulation. Next, we proceed as in the proof of Proposition 5, modulo adequate adaptations that address the crucial difference between Construction 4 and Construction 8. Recall that the difference is that in Construction 4 the reference string is viewed as a sequence of images of the permutation, whereas in Construction 8 the reference string is viewed as a sequence of ℓ-bit long random-sequences that may be used to generate such images. Following is a rigorous description of the current P'-simulator, when invoked on input $x \in \{0,1\}^n$ and using the P-simulator, denoted M.

1. Obtain $(\alpha, \tau) \leftarrow I(1^n)$.
2. Obtain $((\sigma_1, ..., \sigma_t), (i_1, ..., i_t), \pi) \leftarrow M(x)$.
3. For every $j = 1, .., t$, generate a random pair $(p_j, s_{i_j}) \in D_\alpha \times \{0,1\}^\ell$ such that $f_\alpha(p_j) = S'(\alpha, s_{i_j})$ and $b(p_j) = \sigma_j$.

   Note that this operation can be efficiently implemented by either relying on the additional enhancement introduced in Def. 7 or by merely relying on the fact that the simulator knows the trapdoor τ and can thus invert f_α. (The "forced" use of the additional enhancement of Def. 7 arises in the proof of indistinguishabilitry provided below.)
4. For every $i \in [m] \setminus \{i_j : j = 1, .., t\}$, select s_i uniformly in $\{0,1\}^\ell$.
5. Output $(x, (s_1, ..., s_m), (\alpha, (i_1, ..., i_t), \pi, (p_1, ..., p_t)))$.

Using the hypothesis that b is a hard-core of the collection $\{f_\alpha\}$ and the doubly-enhanced hardness of this collection, it follows that the output of the P'-simulator is computationally indistinguishable from the verifier's view (when receiving a proof from P'). Again, the only difference between the simulation and the real view is that in the former the values on the actual reference strings do not necessarily match the values of the corresponding hidden bits seen by P. However, this difference is computationally indistinguishable by the hypothesis that $b(f_\alpha^{-1}(S(\alpha, U_\ell)))$ is infeasible to predict when given (α, U_ℓ). Specifically, we need to show that, for $H \stackrel{\text{def}}{=} [m] \setminus \{i_j : j = 1, .., t\}$, it is infeasible to distinguish a sequence of $|H|$ uniformly selected n-bit strings (representing the sequence $(s_i)_{i \in H}$ produced in the simulation) from a corresponding sequence of s_i's that fits a (partially) given sequence of $b(f_\alpha^{-1}(S(\alpha, s_i)))$ values (as in the real interaction). At this point, we encounter a difficulty that seems to require the doubly-enhanced hypothesis.

The point is that the indistinguishability of the two sequences is demonstrated by showing that, given a prefix of the second sequence, it is infeasible to predict the $b(f_\alpha^{-1}(S(\alpha, \cdot)))$-value of the next element. That is, we wish to show that, for every i, given a randomly selected α and a uniformly selected sequence $s_1, ..., s_{i-1}, s_i$ along with the values $b(f_\alpha^{-1}(S(\alpha, s_1))), ..., b(f_\alpha^{-1}(S(\alpha, s_{i-1})))$, it is infeasible to predict the value of $b(f_\alpha^{-1}(S(\alpha, s_i)))$. Recall that the standard approach toward this task is to use a reducibility argument in order to derive a contradiction to the hard-core hypothesis (which refers to a single $s = s_i$ for which $b(f_\alpha^{-1}(S(\alpha, s)))$ is unpredictable), by generating the auxiliary prefix $s_1, ..., s_{i-1}$ along with $b(f_\alpha^{-1}(S(\alpha, s_1))), ..., b(f_\alpha^{-1}(S(\alpha, s_{i-1})))$. Thus, given only α (and $s = s_i$), we need to be able to generate a random sequence $s_1, ..., s_{i-1}$ along with the corresponding $b(f_\alpha^{-1}(S(\alpha, s_j)))$'s. But this is easy to do given the doubly-enhanced hypothesis. ■

Open Problem: *Under what intractability assumptions is it possible to construct non-interactive zero-knowledge proofs* (NIZKs) *with efficient prover strategies for any set in $\mathcal{NP}$? In particular, does the existence of arbitrary collections of trapdoor permutations suffice?* We mention that the assumption used in constructing such NIZKs effects the assumption used in (general) constructions of public-key encryption schemes that are secure under chosen ciphertext attacks (see, e.g., [G2, Thm. 5.4.31]).

Acknowledgments

We are grateful to Jonathan Katz for pointing out the gap in [G2, Apdx. C.4.1]. While being embarrassed about such flaws, we feel deeply indebted to those discovering them and bringing them to our attention.

We thank Ron Rothblum for pointing out that a previous version of this write-up failed to deliver the crucial point, which is currently spelled out at the end of the proof of Proposition 9.

Appendix: On the RSA and Rabin Collections

In this appendix we show that adequate versions of the RSA and Rabin collections satisfy the two aforementioned enhancements (presented in Definitions 6 and 7, respectively). Establishing this claim is quite straightforward for the RSA collection, whereas for the Rabin collection some modifications (of the straightforward version) seem necessary. In order to establish this claim we will consider a variant of the Rabin collection in which the corresponding domains are dense and easy to recognize, and will show that having such domains suffices for establishing the claim.

A.1 The RSA collection satisfies both enhancements

We start our treatment by considering the RSA collection (as presented in [G1, Sec. 2.4.3.1] and further discussed in [G1, Sec. 2.4.3.2]). Note that in order to discuss the enhanced hardness condition (of Def. 6) it is necessary to specify the domain sampler, which is not entirely trivial (since sampling Z_N^* (or even Z_N) by using a sequence of unbiased coins is not that trivial).

A natural sampler for Z_N^* (or Z_N) generates random elements in the domain by using a regular mapping from a set of sufficiently long strings to Z_N^* (or to Z_N). Specifically, the sampler uses $\ell \stackrel{\rm def}{=} 2\lfloor \log_2 N \rfloor$ random bits, views them as an integer in $i \in \{0, 1, ..., 2^\ell - 1\}$, and outputs $i \bmod N$. This yields an almost uniform sample in Z_N, and an almost uniform sample in Z_N^* can be obtained by discarding the few elements in $Z_N \setminus Z_N^*$.

The fact that the foregoing implementation of the RSA collection satisfies Definition 6 (as well as Definition 7) follows from the fact that it has an efficient reversed-sample (which eliminates the potential gap between having a domain element and having a random sequence of coins that makes the domain-sample output this element). Specifically, given an element $e \in Z_N$, the reversed-sampler outputs an almost uniformly distributed element of $\{i \in \{0, 1, ..., 2^\ell - 1\} : i \equiv e \pmod N\}$ by selecting uniformly $j \in \{0, 1, ..., \lfloor 2^\ell / N \rfloor - 1\}$ and outputting $i \leftarrow j \cdot N + e$.

A.2 Versions of the Rabin collection that satisfy both enhancements

In contrast to the case of the RSA, the Rabin Collection (as defined in [G1, Sec. 2.4.3.3]), does not satisfy Definition 6 (because the coins of the sampling algorithm give away a modular square root of the domain element). Still, the Rabin Collection can be easily modify to yield an *doubly-enhanced* collection of trapdoor permutations, provided that factoring is hard (in the same sense as assumed in [G1, Sec. 2.4.3]).

The modification is based on modifying the domain of these permutations (following [ACGS]). Specifically, rather than considering the permutation induced (by the modular squaring function) on the set Q_N of the quadratic residues modulo N, we consider the permutations induced on the set M_N, where M_N contains all integers in $\{1, ..., N/2\}$ that have Jacobi symbol modulo N that equals 1. Note that, as in case of Q_N, each quadratic residue has a unique square root in M_N (because exactly two square roots have Jacobi symbol that equals 1 and their sum equals N; indeed, as in case of Q_N, we use the fact that -1 has Jacobi symbol 1). However, unlike Q_N, membership in M_N can be determined in polynomial-time (when given N without its factorization). Actually, squaring modulo N is a 1-1 mapping of M_N to Q_N. In order to obtain a permutation over M_N, we modify the function a little such that if the result of modular squaring is bigger than $N/2$ then we use its additive inverse (i.e., rather than outputting $y > N/2$, we output $N - y$).

Using the fact that M_N is dense (w.r.t $\{0,1\}^{\lfloor \log_2 N \rfloor + 1}$) and easy to recognize, we may proceed in one of two ways, which are actually generic. Thus, let us assume that we are given an arbitrary collection of trapdoor permutations, denoted $\{f_\alpha : D_\alpha \to D_\alpha\}_{\alpha \in \overline{I}}$, such that $D_\alpha \subseteq \{0,1\}^{|\alpha|}$ is dense (i.e., $|D_\alpha| > 2^{|\alpha|}/{\rm poly}(|\alpha|)$)[3] and easy to recognize (i.e., there exists an efficient algorithm that given (α, x) decides whether or not $x \in D_\alpha$).

1. The natural way to proceed is showing that the collection $\{f_\alpha\}$ itself is *doubly-enhanced*. This is shown by presenting a rather straightforward domain-sampler that satisfies the enhanced hardness condition (of Def. 6), and noting that this sampler has an efficient reversed sampler (which implies that Def. 7 is satisfied).

[3]Actually, a more general case, which is used for the Rabin collection, is one in which $D_\alpha \subseteq \{0,1\}^{\ell(|\alpha|)}$ satisfies $|D_\alpha| > 2^{\ell(|\alpha|)}/{\rm poly}(|\alpha|)$, where $\ell : \mathbb{N} \to \mathbb{N}$ is a fixed function.

The domain-sampler that we have in mind repeatedly selects random (i.e., uniformly distributed) $|\alpha|$-bit long strings and output the first such string that resides in D_α (and a special failure symbols if $|\alpha| \cdot 2^{|\alpha|}/|D_\alpha|$ attempts have failed). This sampler has an efficient reversed-sampler that, given $x \in D_\alpha$, generates a random sequence of $|\alpha|$-bit long strings and replaces the first string that resides in D_α by x.

2. An alternative way of obtaining a doubly-enhanced collection is to first define a (rather artificial) collection of *weak* trapdoor permutations, $\{f'_\alpha : \{0,1\}^{|\alpha|} \to \{0,1\}^{|\alpha|}\}_{\alpha \in \overline{I}}$, such that $f'_\alpha(x) = f_\alpha(x)$ if $x \in D_\alpha$ and $f'_\alpha(x) = x$ otherwise. Using the amplification of a weak one-way property to a standard one-way property (as in [G1, Sec. 2.3&2.6]), we are done.

(Indeed, in the first alternative we amplified the trivial domain-sampler that succeeds with noticeable probability, whereas in the second alternative we amplified the one-way property of the trivial extension of f_α to the domain $\{0,1\}^{|\alpha|}$.) Either way we obtain a *doubly-enhanced* collection of trapdoor permutations, provided that $\{f_\alpha\}$ is an ordinary collection of trapdoor permutations.

We mention that the foregoing modifications of the Rabin collection follows the outline of the second modification that is presented in [G2, Apdx. C.1]. In contrast, as pointed out by Jonathan Katz, the first implementation (of an enhanced trapdoor permutation based on factoring) that is presented in [G2, Apdx. C.1] is not doubly-enhanced.

References

[ACGS] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr. RSA/Rabin Functions: Certain Parts are As Hard As the Whole. *SIAM Jour. on Comput.*, Vol. 17, April 1988, pages 194–209.

[BY] M. Bellare and M. Yung. Certifying Permutations: Noninteractive Zero-Knowledge Based on Any Trapdoor Permutation. *Journal of Cryptology*, Vol. 9, pages 149-166, 1996.

[G1] O. Goldreich. *Foundation of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[G2] O. Goldreich. *Foundation of Cryptography: Basic Applications.* Cambridge University Press, 2004.

[G3] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[GL] O. Goldreich and L.A. Levin. Hard-core Predicates for any One-Way Function. In *21st STOC*, pages 25–32, 1989.