Electronic Colloquium on Computational Complexity, Report No. 159 (2011)

Enhancements of Trapdoor Permutations

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

Ron Rothblum
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
ron.rothblum@weizmann.ac.il

August 3, 2011

Abstract

We take a closer look at several enhancements of the notion of trapdoor permutations. Specifically, we consider the notions of *enhanced trapdoor permutation* (Goldreich 2004) and *doubly enhanced trapdoor permutation* (Goldreich 2008) as well as intermediate notions (Rothblum 2010). These enhancements arose in the study of Oblivious Transfer and NIZK, but they address natural concerns that may arise also in other applications of trapdoor permutations. We clarify why these enhancements are needed in such applications, and show that they actually suffice for these needs.

Keywords: Trapdoor Permutations, Oblivious Transfer, Non-Interactive Zero-Knowledge

ISSN 1433-8092

Contents

1 Introduction

This article surveys and studies two enhancements of the notion of trapdoor permutations (TDP). Our exposition clarifies how these enhancements of TDP emerge in two central applications, and shows that these enhancements actually suffice for the corresponding applications. As is often the case when studying known definitions, we find it useful to start with a review of the historical roots of the various definitions.

1.1 A historical perspective

The notion of trapdoor permutations was formulated by Yao [24] as a sufficient condition for the construction of secure public-key encryption schemes (as put forward by Diffie and Hellman [7] and rigorously defined by Goldwasser and Micali [20]). Indeed, building on the ideas of [6, 20], Yao [24] showed that any collection of trapdoor permutations can be used to construct a secure public-key encryption scheme.

Loosely speaking, the notion of trapdoor permutations (TDP) refers to a collection of permutations that are easy to sample and have domains that are easy to sample from (when given the description of the permutation). The main requirements are that these permutations are easy to evaluate, easy to invert when given a suitable trapdoor, but hard to invert when only given the description of the permutation (but not the trapdoor).

The minimal requirements regarding the sampleability of permutations and their domains were glossed over when constructing secure public-key encryption schemes (both in [20, 24] and [5] (cf. [14, Sec. 5.3.4])). Consequently, in later years, researchers have tended to think of TDP in terms of the idealized case in which the corresponding sampling algorithms are trivial (i.e., they just output their random coin tosses). This tendency seems to be the source of the flaws that followed.

Specifically, trapdoor permutations were suggested as a basis for the construction of Oblivious Transfer (OT) protocols [8] and general Non-Interactive Zero-Knowledge proof (NIZK) systems [9]. In both cases, an idealized TDP suffices for these constructions, and this promoted a false belief that a general TDP would also do. (We note that the difference between idealized TDP and general TDP is crucial, because no candidate for an idealized TDP is known, whereas a general TDP can be constructed based, say, on factoring.)

The first difficulty was discovered by Bellare and Yung [2], who observed that the soundness of the NIZK construction of [9] relies on the hypothesis that the set of permutations in the collection is easily recognizable (which is trivial in the idealized case). Since this hypothesis does not hold in the known candidate TDPs, Bellare and Yung relaxed the hypothesis to requiring that membership in the aforementioned set (or actually just being almost 1-1) can be demonstrated by a special-purpose NIZK, and showed that the relaxed hypothesis can be met for all known candidates [2]. Interestingly, their presentation avoids the problem of sampling in the domains of the various permutations, by postulating that these domains consist of all strings of a specific length (which, in turn, can be easily determined by the description of the permutation). We note that all known candidates can be converted to such a form. Still, there is a difference between the latter form (of trivially sampleable domains) and the general case, and the rest of our discussion refers to that difference.

The difference between TDP with trivially sampleable domains and general TDP was first observed by Goldreich [12], when producing a detailed proof of the secure multi-party computation result of [19], which relies on the construction of OT. Specifically, he discovered that the construction of OT outlined in [8, 19], which works when the TDP has trivially sampleable domains, may not be secure when a general TDP is employed, but is secure when using an enhanced notion of a TDP. This enhancement requires that the permutation is hard to invert also when one is given the coins that were used to sample the domain element (rather than merely the domain element itself).

We note that the enhanced notion of hardness-to-invert collapses to the original notion (of hardness-to-invert) in the case that the domain-sampling algorithm is trivial (i.e., the permutation's domain consists of all strings of a specific length). More generally, this enhancement is insignificant whenever it is easy to invert the domain-sampling algorithm (i.e., given an element in the domain, find random coins that cause the domain-sampling algorithm to produce the given element). We also mention that, for the purpose of constructing OT, the enhancement may be avoided if the permutation's domain is dense (by using a more complex OT construction; cf. [22]).

Turning back to the main thread, we mention that while Goldreich [14, Apdx. C.4.1] claimed that the aforementioned enhancement of TDP suffices for constructing general NIZK, Jon Katz raised doubts regarding this claim. These justified doubts led Goldreich [16] to propose the notion of doubly-enhanced TDP, and show that this notion does suffice for constructing general NIZK.

Subsequent work by Rothblum [23] uncovered a taxonomy of notions of TDP, residing between enhanced TDP and doubly-enhanced TDP. These intermediate notions will be further discussed in Section 6.

To summarize the historical account, we note that the general formulation of TDP was envisioned as the most general and/or minimal formulation of a collection of functions that allows for the construction of secure public-key encryption schemes. This application determined the main requirements (i.e., easy to evaluate, easy to invert when given a suitable trapdoor, but hard to invert when not given the trapdoor), whereas the sampling conditions were stated in the most general form possible (i.e., merely requiring easy sampling of permutations and domain elements). However, the general (and innocent-looking) formulation of the sampling conditions turned out to be a problem when seeking to construct OT and general NIZK based on TDP. We shall review these difficulties in the next subsection.

1.2 The difficulties

The following presentation does not preserve the chronological order (which was followed in Section 1.1). Also, we shall only sketch the nature of the difficulties that arise and the way in which they are addressed by the enhancements. Corresponding detailed descriptions appear in later sections.

We shall refer to a collection of the permutations of the form $f_\alpha : D_\alpha \to D_\alpha$, where α is the index (or description) of the permutation f_α, and to two sampling algorithms: (1) an index-sampling algorithm I_1 that, on input coins s, outputs an index $\alpha = I_1(s)$; and (2) a domain-sampling algorithm S that, on input an index α and coins $r \in \{0,1\}^{|\alpha|}$, outputs an element in the corresponding domain (i.e., $S(\alpha, r) \in D_\alpha$).[1] Indeed, the index of the permutation is associated with its description, and the hardness-to-invert condition refers to inverting f_α on y, when α and y are selected by the foregoing sampling algorithms (i.e., $\alpha \leftarrow I_1(s)$ and $y \leftarrow S(\alpha, r)$, where s and r are selected uniformly in $\{0,1\}^n$). The idealized case, mentioned in Section 1.1, refers to the case that all strings are valid indices and $D_\alpha = \{0,1\}^{|\alpha|}$ for every α.

[1] Indeed, for simplicity (and without loss of generality), we assumed here that the number of coins taken by $S(\alpha, \cdot)$ is $|\alpha|$. Similarly, we assume here that $|I_1(s)| = |s|$.

The first difficulty refers to the construction of OT based on TDP. The security of the standard construction of [8] (as well as other applications) relies on the hypothesis that a party knowing $\alpha = I_1(s)$ and r (which are chosen as above), is unable to invert f_α on $S(\alpha, r)$. Note that this would follow from the hardness-to-invert condition if given (α and) y one can efficiently find a random r such that $y = S(\alpha, r)$. However, there is no reason to assume that such a "reverse-sampling" is feasible *in general*. Instead, one may define enhanced TDP as *TDP that satisfy this enhanced hardness-to-invert condition*, and note that the popular candidates for TDP actually yield enhanced TDP. Using any such enhanced TDP allows to construct an OT protocol.

The second difficulty refers to the soundness of a general NIZK based on TDP. In this setting the prover is supposed to select a random permutation (in the TDP collection) and send its description to the verifier, and the soundness of the proof system relies on the hypothesis that the description sent (i.e., α) indeed refers to an almost 1-1 function (i.e., f_α is almost 1-1). In general, as observed in [2], this hypothesis needs to be tested (by the verifier), and a natural way of going so it asking the prover to provide the inverses of the function (i.e., of f_α) on a sequence of randomly selected domain elements. If we use a general TDP, then the prover is asked to provide the inverses of f_α on $S(\alpha, r_1), ..., S(\alpha, r_m)$, when given $r_1, ..., r_m$ that are randomly distributed (and are part of the common random string).

The latter proposal brings us to a third difficulty, which refers to the question of whether providing the value of $f_\alpha^{-1}(S(\alpha, r))$ for a random r is zero-knowledge. The answer would have been positive if (given α) it were feasible to generate random samples of the form (x, r) such that $S(\alpha, r) = f_\alpha(x)$, since in this case $x = f_\alpha^{-1}(S(\alpha, r))$. But, again, this condition may not hold in general, and postulating that it does hold is the contents of another enhancement.[2]

The construction of general NIZK based on TDP uses the two aforementioned enhancements: The first enhancement is used in order to argue that, when seeing α and r, the (unrevealed) value of $f_\alpha^{-1}(S(\alpha, r))$ remains secret. The second enhancement is used in order to argue that revealing $f_\alpha^{-1}(S(\alpha, r))$ for a random r is actually zero-knowledge (w.r.t a fixed α).

We note that the two difficulties that give rise to the two enhancements of TDP are natural ones, and are likely to arise also in other (sophisticated) applications of TDP. We also note that the source of both difficulties is that, in general, obtaining an element of D_α may not be computationally equivalent to obtaining the coins that are used to produce this element. This computational equivalence holds only in the special case in which there exists an efficient reversed domain-sampler (i.e., a probabilistic polynomial-time algorithm that on input (α, y) outputs a string that is uniformly distributed in $\{r : S(\alpha, r) = y\}$). (We mention that adaquate implementations and/or variants of the popular candidate collections of trapdoor permutations (e.g., the RSA and Rabin collections) do have efficient reversed domain-sampler.)

1.3 The current article

This article provides a revised account of the findings in [16, 23]. The focus of the current article is on the notion of TDP and its enhancements. In contrast, the focus of Goldreich [16] is on the application to general NIZK, whereas the starting point of Rothblum [23] is the two enhancements mentioned in Sections 1.1 and 1.2. In particular, we explore the entire range between general TDP and doubly-enhanced TDP, while Rothblum [23] focuses on the range between enhanced

[2] Again, if there exists a reverse-sampler for S, then it is easy to generate random samples of the form (x, r) such that $S(\alpha, r) = f_\alpha(x)$. This is done by uniformly selecting x, and obtaining r from the reverse-sampler.

TDP and doubly-enhanced TDP. We also present an additional application of the aforementioned enhancements in the context of public-key encryption.

We note that [16, 23] have not been published before, and the current article should be viewed as a combined journal version of these two works. We view the current article as a hybrid of a survey and a research article, and hope that it will help to clarify the confusion around the various notions of enhanced TDP.

Organization. In Section 2 we recall the definition of TDP and present its two aforementioned enhancements. These enhancements were motivated in Section 1.2, and these motivations will be detailed in the subsequent sections. In particular, Section 3 details how the first enhancement arises out of the construction of 1-out-of-2 OT, whereas Section 4 details how the second enhancement arises out of the construction of 1-out-of-3 OT and general NIZK systems. In Section 5 we discuss an application of the two enhancedments to oblivious sampling of ciphertexts in public-key encryption. In Section 6, we consider some intermediate notions of TDP that arise naturally in the foregoing applications.

Notation. We denote by $A(x)$ the output distribution of algorithm A on input x and by $A(x;r)$ the *deterministic* output of algorithm A on input x and the random string r.

2 Definitions

Recall that a collection of trapdoor permutations, as defined in [13, 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 any 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 any 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 standard hardness condition (as in [13, 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 the index 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 $A = \{A_n\}_n$), it holds that

$$\Pr_{\substack{\alpha \leftarrow I_1(1^n) \\ x \leftarrow S(\alpha)}}[A(\alpha, f_\alpha(x)) = x] \;=\; \mu(n), \tag{1}$$

where μ denotes a generic negligible function. 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 (1) is

$$\Pr_{\substack{\alpha \leftarrow I_1(1^n) \\ r \leftarrow R_n}}[A(\alpha, S(\alpha; r)) = f_\alpha^{-1}(S(\alpha; r))] \;=\; \mu(n), \tag{2}$$

where R_n denotes the distribution of the coins of S on n-bit long inputs. That is, A fails to invert f_α on $S(\alpha; r)$, where α and r are selected as above.

We note that the idealized cased mentioned in the introduction refers to the special case in which (1) $I_1(1^n)$ is uniformly distributed in $\{0,1\}^n$, and (2) $D_\alpha = \{0,1\}^{|\alpha|}$. Recall that Condition (1) seems unrealistic, and avoiding it was the contents of [2]. Our focus, instead, is on avoiding (or relaxing) Condition (2). Furthermore, we focus on the case that the domain sampler S cannot be efficiently inverted.

Before proceeding, recall that any collection of trapdoor permutations can be augmented by a hard-core predicate [6, 17], denoted h. Loosely speaking, such a predicate h is easy to compute, but given $\alpha \leftarrow I_1(1^n)$ and $x \leftarrow S(\alpha)$, it is infeasible to guess the value of $h(\alpha, f_\alpha^{-1}(x))$ non-negligibly better than by a coin toss.

Enhanced trapdoor permutations. Although the foregoing definition suffices for some applications, in other cases (further discussed in Sections 3 and 4) 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, and so we may omit it from the input given to the potential inverter.)

Definition 2.1 (enhanced trapdoor permutations [14, Def. C.1.1])**.** *Let $\{f_\alpha : D_\alpha \to D_\alpha\}$ be a collection of trapdoor permutations. We say that this collection is* enhanced (and call it an enhanced collection of trapdoor permutations) *if, for every probabilistic polynomial-time algorithm A, it holds that*

$$\Pr_{\substack{\alpha \leftarrow I_1(1^n) \\ r \leftarrow R_n}}[A(\alpha, r) = f_\alpha^{-1}(S(\alpha; r))] \;=\; \mu(n), \tag{3}$$

where R_n and μ are as above. The non-uniform version is defined analogously.

Definition 2.1 requires that it is infeasible to invert f_α on $S(\alpha; r)$, when given α and r, which are selected as above (i.e., $\alpha \leftarrow I_1(1^n)$ and $r \leftarrow R_n$). Note that any trapdoor permutations in which $D_\alpha = \{0,1\}^{|\alpha|}$ satisfies Definition 2.1 (because, without loss of generality, the sampling algorithm S may satisfy $S(\alpha; r) = r$). This implies that modified versions of the RSA and Rabin collections satisfy Definition 2.1. More natural versions of both collections can also be shown to satisfy Definition 2.1. For further discussion see Appendix B.

Any collection of enhanced trapdoor permutations can also be augmented by a hard-core predicate (or rather by an enhanced hard-core predicate). Loosely speaking, such a predicate h is easy to compute, but given $\alpha \leftarrow I_1(1^n)$ and $r \leftarrow R_n$, it is infeasible to guess the value of $h(\alpha, f_\alpha^{-1}(S(\alpha; r)))$ non-negligibly better than by a coin toss. Before presenting the actual definition, we stress that the proof of [17] extends to the current setting (cf. [13, Sec. 2.5.2] or better [15, Thm. 7.8]).

Definition 2.2 (enhanced hardcore predicate)**.** *Let $\{f_\alpha : D_\alpha \to D_\alpha\}$ be a collection of enhanced trapdoor permutations. We say that $h : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ is an* enhanced hardcore predicate *of $\{f_\alpha\}$ if h is polynomial-time computable and for every probabilistic polynomial-time algorithm A,*

$$\Pr_{\substack{\alpha \leftarrow I_1(1^n) \\ r \leftarrow R_n}}[A(\alpha, r) = h(\alpha, f_\alpha^{-1}(S(\alpha; r)))] \tag{4}$$

where R_n and μ are as above. The non-uniform version is defined analogously.

For simplicity, for both standard and enhanced hardcore predicates, we usually drop the index from the input of h and write $h(x)$ where we actually mean $h(\alpha, x)$. (This can be done without loss of generality, since the hardcore predicate of [17] does not use the index α.)

Doubly-enhanced trapdoor permutations. Although collection of enhanced trapdoor permutations suffice for the construction of Oblivious Transfer (see Section 3), it seems that they do not suffice for constructing a general NIZK proof system (see Section 4). Thus, we further enhance Definition 2.1 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 2.3 (doubly-enhanced trapdoor permutations)**.** *Let $\{f_\alpha : D_\alpha \to D_\alpha\}$ be an enhanced collection of trapdoor permutations* (as in Def. 2.1)*. 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 2.3 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 $\{r : S(\alpha; r) = y\}$). Indeed, the existence of a reversed domain-sampler eliminates the difference between producing random pairs (x, r) such that $f_\alpha(x) = S(\alpha; r)$ and producing random pairs of the form $(x, S(\alpha; r))$ such that $f_\alpha(x) = S(\alpha; r)$ (i.e., random (x, y) such that $f_\alpha(x) = y$). (Similarly, the existence of a reversed domain-sampler eliminates the difference between being given (α, r) and being given $(\alpha, S(\alpha; r))$.)

A useful relaxation of Definition 2.3 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 suitable 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 domains satisfies this condition, where $D_\alpha \subseteq \{0,1\}^{|\alpha|}$ is dense if $|D_\alpha| \geq 2^{|\alpha|}/\text{poly}(|\alpha|)$. The reason is that having such domains offer a very simple domain-sampler, which can be inverted efficiently: The sampler merely generates a sequence of $|\alpha|$-bit long strings and outputs the first string in D_α, whereas the reversed domain-sampler just generates such a sequence and replaces the first string in D_α by the element given to it.

Again, any collection of doubly-enhanced trapdoor permutations can also be augmented by a hard-core predicate (or rather by a doubly-enhanced hard-core predicate). That is, such a predicate is required to satisfy the conditions of Definition 2.2 with respect to a collection $\{f_\alpha : D_\alpha \to D_\alpha\}$ that is doubly-enhanced (rather than just enhanced).

3 Enhanced TDP and 1-out-of-2 OT

Oblivious transfer (OT) is an interactive protocol between two parties, a sender and a receiver. In the 1-out-of-2 version, introduced by Even *et al.* [8], the sender gets as input two bits σ_0 and σ_1 and the receiver gets a single bit i. The parties exchange messages and at the end of the protocol the receiver should learn the bit σ_i but gain no knowledge regarding σ_{1-i} and the sender should gain no knowledge of i. Oblivious transfer turned out to be a central cryptographic tool, especially in the context of secure multi-party computation [19].

In this section we present the standard OT protocol based on TDP, which originates in [8, 19] and is hereafter referred to as the EGL protocol. We highlight the difficulty the arises when the protocol is implemented with *general* TDP, and show that under the strengthened notion of *enhanced* TDP the protocol is actually secure.

Semi-Honest OT. We consider OT in the semi-honest model, where both parties follow the protocol but may try to learn additional information based on their view of the interaction. Recall that (using any one-way function) Goldreich *et al.* [19] showed a compiler that transforms protocols secure in the semi-honest model into protocols that are secure against malicious adversaries (which may deviate arbitrarily from the specified protocol). Informally an OT protocol should satisfy the following requirements (w.r.t the sender input $(\sigma_0, \sigma_1) \in \{0,1\}^2$ and the receiver input $i \in \{0,1\}$):

1. Correctness – At end of the protocol the receiver outputs σ_i (and the sender outputs nothing).

2. Receiver privacy – the sender does not learn the selection bit i (i.e., the view of the sender can be simulated based on σ_0, σ_1).

3. Sender privacy – the receiver does not learn the bit σ_{1-i} (i.e., the view of the receiver can be simulated based on i and σ_i).

A precise definition of OT is provided in Appendix A.

3.1 The EGL Protocol

The EGL protocol uses a TDP $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ with a hardcore predicate h. We denote the algorithms associated with the TDP by I (index/trapdoor sampler), S (domain sampler), F (forward evaluation) and B (backward evaluation). The protocol is depicted in Figure 1.

Even when implemented with general TDP (or in fact any collection of permutations), it is not hard to verify that correctness holds. That is, at the end of the protocol, the receiver outputs σ_i. Also, the receiver's privacy holds, since the the sender just sees two uniformly distributed elements $y_0, y_1 \in D_\alpha$, and therefore the selection bit i is perfectly hidden (here we use the fact that f_α is a permutation).

It is tempting to argue that the sender's privacy also holds. The misleading intuition is that the receiver does not know the preimage $x_{1-i} = f_\alpha^{-1}(z_{1-i})$ and therefore the bit σ_{1-i} is computationally hidden by the "mask" $h(x_{1-i})$. The reason this argument fails is that the receiver may be able to efficiently compute the preimage of z_{1-i} under f_α^{-1} by using the random coins that it has used to generate z_{1-i}. A general TDP does *not* guarantee that given an index α and random coins that are used to sample $z_{1-i} \in D_\alpha$ it is infeasible to obtain $f_\alpha^{-1}(z_{1-i})$ or just guess $h(f_\alpha^{-1}(z_{1-i}))$ with non-negligible advantage. Indeed, as we shall see next, such guarantees are provided by enhanced TDPs.

$$
\begin{array}{ccc}
\underline{Sender(1^n, \sigma_0, \sigma_1)} & & \underline{Receiver(1^n, i)} \\
(\alpha, \tau) \leftarrow I(1^n) & & \\
& \xrightarrow{\quad\quad\quad\quad \alpha \quad\quad\quad\quad} & \\
& & z_0, z_1 \leftarrow S(\alpha) \\
& & y_i = f_\alpha(z_i) \\
& & y_{1-i} = z_{1-i} \\
& \xleftarrow{\quad\quad\quad\quad y_0, y_1 \quad\quad\quad\quad} & \\
x_0 = f_\alpha^{-1}(y_0),\ x_1 = f_\alpha^{-1}(y_1) & & \\
c_0 = h(x_0) \oplus \sigma_0,\ c_1 = h(x_1) \oplus \sigma_1 & & \\
& \xrightarrow{\quad\quad\quad\quad c_0, c_1 \quad\quad\quad\quad} & \\
& & \text{Output } c_i \oplus h(z_i)
\end{array}
$$

Figure 1: The EGL protocol for 1-out-of-2 Oblivious Transfer

3.2 Security of the EGL protocol based on ETDP

The source of trouble (as discussed right above) is that the random coins of the sampling algorithm may reveal the preimage of the sampled element. To overcome this difficulty we use a stronger assumption about the TDP: We assume that it is actually an *enhanced* TDP (Definition 2.1). Recall that the enhancement means that even given the random coins of the domain sampler it is hard to invert a sampled element. If the EGL protocol is implemented with enhanced TDP, then intuitively the additional information that the receiver has regarding the sampled elements no longer helps and the protocol is secure. Thus, we obtain the following.

Claim 3.1. *If $\{f_\alpha\}$ is an* enhanced *TDP and h is its* enhanced *hardcore predicate, then the EGL protocol securely implements 1-out-of-2 OT in the semi-honest model.*

Proof. We first detail the correctness and receiver's privacy, which were sketched above. Correctness follows by the following syntaxtic equalities:

$$
\begin{aligned}
c_i \oplus h(z_i) &= (h(x_i) \oplus \sigma_i) \oplus h(z_i) \\
&= (h(x_i) \oplus \sigma_i) \oplus h(f_\alpha^{-1}(y_i)) \\
&= (h(x_i) \oplus \sigma_i) \oplus h(x_i) \\
&= \sigma_i.
\end{aligned}
$$

To show that sender and receiver privacy hold, we show simulators that based on the local input and output of the corresponding party simulates the party's view.

We first show that the receiver's privacy holds. Consider the following (simple) simulation of the sender's view. On input (σ_0, σ_1), the simulator selects a random string s and uses it to sample an index α of a permutation. The simulator also selects two elements $y_0, y_1 \leftarrow S(\alpha)$ in the permutation's domain and outputs $((\sigma_0, \sigma_1, 1^n), s, (y_0, y_1))$, where the first part is the sender's input, the second part its random string, and the third part is the message that it receives. Because α is a permutation, the simulated view is distributed identically to the actual view of the sender in the protocol execution, and therefore the receiver enjoys perfect privacy.

We now turn to the sender's privacy. Recall that the simulator gets i and σ_i and needs to simulate the receiver's view in the protocol execution. The simulator proceeds as follows:

Simulator$(i, \sigma_i, 1^n)$

1. Select a random index α of a permutation.
2. Select two random strings r_1 and r_2 for the domain sampler S and set $z_j = S(\alpha; r_j)$ for $j \in \{0,1\}$. Set $y_i = f_\alpha(z_i)$ and $y_{1-i} = z_{1-i}$.
3. Set $c_i = \sigma_i \oplus h(z_i)$ and select a bit $c_{1-i} \in \{0,1\}$ uniformly at random.
4. Output $(i, (r_1, r_2), (\alpha, (c_1, c_2)))$, where the first part simulates the receiver's input, the second part its random string, and the third part simulates the two messages that it receives.

We claim that the output of the simulator is computationally indistinguishable from the actual view of the receiver. To see this observe that, except for c_{1-i}, the output of the simulator is distributed *identically* to the view of the receiver. Thus, an adversary that distinguishes between the simulation and the actual execution view, also distinguishes between c_{1-i} (which is distributed uniformly and independent of anything else) and $h(f_\alpha^{-1}(S(\alpha; r_{1-i})))$, when given random α and r_{1-i}, which contradicts the hypothesis that h is an enhanced hardcore predicate.[3] ■

4 Doubly-Enhanced TDP, 1-out-of-3 OT, and NIZK

In the previous section we showed that the pressumption that a randomly sampled element and the random coins used to sample it are computationally equivalent is false (i.e., it may be infeasible to retreive the coins from the sampled element), and that this fact may lead to the insecurity of cryptographic protocols that rely on this (false) pressumption. While enhanced TDP do bring us closer to idealized TDP (in which random coins and sampled elements are computationally equivalent), in this section we demonstrate that a significant gap exists also between the enhanced and the idealized notions of TDP.

The gap that we refer to is related to the fact that many applications of TDP use the property that for a given permutation α it is easy to generate a random pair (x, y) such that $y = f_\alpha(x)$. This can obviously be done by just sampling x at random (in the domain) and applying the permutation to obtain y (as $f_\alpha(x)$). However, in some applications a variant of this property is needed; namely, the ability to generate a random pair (x, r) such that r is the *random string* used to sample $y = f_\alpha(x)$. For idealized TDP this is easy since we can sample (x, y) and use $r = y$, but for general TDP obtaining r from y (s.t. $y = S(\alpha; r)$) may be infeasible.[4] (We mention that the ability to generate such random pairs may be used in the proof of security of a given protocol and not in the protocol execution.)

Next we show two protocols that use enhanced TDP for which the infeasibility of generating such pairs may lead to security problems. These problems (in both protocols) can be resolved by

[3] Specifically, given a random string r, we set some values of i, σ_1, σ_2 such that the adversary distinguishes the simulation from the real execution, and run the simulation with $r_{1-i} = r$. Using the distinguishing gap of the adversary we can guess whether $c_{1-i} = \sigma_{1-i} \oplus h(z_{1-i})$ (with non-negligible advantage), and thus guess the hardcore bit of the f_α-preimage of $S(\alpha; r)$.

[4] Indeed, this potential infeasiblity is the very motivation to the notion of enhanced TDP.

$$
\begin{array}{ccc}
\underline{Sender(1^n, \sigma_1, \sigma_2, \sigma_3)} & & \underline{Receiver(1^n, i)} \\
(\alpha, \tau) \leftarrow I(1^n) & & \\
 & \xrightarrow{\quad\quad \alpha \quad\quad} & \\
 & & z_1, z_2, z_3 \leftarrow S(\alpha) \\
 & & y_i = f_\alpha(z_i) \\
 & & y_j = z_j \text{ for } j \neq i \\
 & \xleftarrow{\quad y_1, y_2, y_3 \quad} & \\
x_j = f_\alpha^{-1}(y_j) \text{ and } c_j = h(x_j) \oplus \sigma_j & & \\
\text{for } j \in \{1, 2, 3\} & & \\
 & \xrightarrow{\quad c_1, c_2, c_3 \quad} & \\
 & & \text{Output } c_i \oplus h(z_i)
\end{array}
$$

Figure 2: The EGL protocol for 1-out-of-3 Oblivious Transfer

using an additional enhancement of TDP referred to as *doubly-enhanced* TDP. Recall that doubly-enhanced TDP (which were defined specifically for this purpose – see Definition 2.3) are *enhanced* TDP for which, in addition to the "standard" enhancement, it is feasible to generate pairs (x, r) as above.

4.1 1-out-of-3 OT

As a first example we consider the natural extension of the EGL protocol to 1-out-of-k OT, for any $k \geq 3$. For simplicity we consider the 1-out-of-3 case in which the sender gets as input three bits $\sigma_1, \sigma_2, \sigma_3$ and the receiver gets an index $i \in \{1, 2, 3\}$. As before, the receiver should learn σ_i but gain no knowledge on σ_1, σ_2 and the sender should gain no knowledge on i. The protocol for the case $k \equiv 3$ is depicted in Figure 2.

At first glance, it seems that the protocol is secure when using *enhanced* TDP (for similar reasons as in the 1-out-of-2 case). Nevertheless, we show that there is a subtle issue that makes it insecure. Before proceeding, we note that there are other ways to extend the EGL protocol to 1-out-of-k while preserving security (e.g., a simple generic transformation from 1-out-of-2 OT to 1-out-of-k, for any $k \geq 2$).[5] Hence 1-out-of-k OT can be constructed based on enhanced TDP, it is only that a specific natural way of doing it (i.e., Figure 2) is insecure. That is, we show the insecurity of the foregoing (natural) construction (of Figure 2) in order to demonstrate that enhanced TDP cannot be treated as an idealized TDP.

The problem with the 1-out-of-3 EGL Protocol. For sake of concreteness, consider the case $i = 1$ (i.e., the receiver wants to receive the bit σ_1). As in the case of 1-out-of-2 OT, correctness and the receiver's privacy follow from the fact that $\{f_\alpha\}_\alpha$ is a collection of permutations. Intuitively it seems that the sender's privacy should also hold. Indeed, since $\{f_\alpha\}_\alpha$ is an *enhanced* TDP the receiver does not know x_2 and x_3. and therefore can learn neither σ_2 nor σ_3 (since each is

[5]Alternatively, we mention that the protocol of Figure 2 is secure for $k = O(\log n)$, provided that the enhanced hardcore predicate that is used is the GL hardcore predicate [17]; for details, see Section 6.

"masked" by a pseudorandom bit). However, privacy requires not only that the individual bits be pseudorandom, but also that they be pseudorandom *together*. But the fact that $h(x_2)$ and $h(x_3)$ are each pseudorandom does not imply that $(h(x_2), h(x_3))$ is pseudorandom. For example, perhaps the adversary can learn $h(x_2) \oplus h(x_3)$, and thus break the security of the 1-out-of-3 EGL protocol (by obtaining the value of $\sigma_2 \oplus \sigma_3$).

This gap in the security proof can actually be used to form an attack. Specifically we refer to the existence of an enhanced TDP (based on a standard intractability assumption) with an enhanced hardcore predicate for which given α, r_1, r_2 it is easy to compute the exclusive-or of the hardcore-bits of the preimages (i.e., $h(f_\alpha^{-1}(S(\alpha; r_1))) \oplus h(f_\alpha^{-1}(S(\alpha; r_2))))$. When the extended EGL protocol is invoked with such an enhanced TDP the receiver can actually learn $\sigma_2 \oplus \sigma_3$ thereby breaking (semi-honest) security. An enhanced TDP with the above property is presented in Appendix C.

Doubly-Enhanced TDP resove the problem. The essence of the problem in the 1-out-of-k EGL protocol is that in this settings (where the adversary sees random strings and not just sampled elements) hardcore bits are not necessarily pseudorandom. Recall that the standard way to prove that many hardcore bits are (simultaneously) pseudorandom is via a hybrid argument. For the hybrid argument to go through in this setting, we need the ability to generate intermediate hybrids, which boils down to generating random pairs $(h(x), r)$ such that $x = f_\alpha^{-1}(S(\alpha; r))$. For *enhanced* TDP we have no guarantee that such pairs can be efficiently generated; furthermore, as mentioned above, there exist enhanced TDP for which it is infeasible to generate such pairs. On the other hand, hardcore bits of *doubly-enhanced* trapdoor permutations *are* pseudorandom in this setting (i.e., also when the adversary sees the randomness used to sample the images). This is the case since the second enhancement guarantees the feasiblility of generating random pairs (x, r) such that $f_\alpha(x) = S(\alpha; r)$, and this allows to employ a hybrid argument in order to prove the following claim.

Claim 4.1. *Suppose that $\{f_\alpha\}$ is a* doubly-enhanced *TDP and h is its* enhanced *hardcore predicate. Then, for every polynomial $m = m(n)$, the sequences $(\alpha, r_1, ..., r_m, h(x_1), ..., h(x_m))$ and $(\alpha, r_1, ..., r_m, b_1, ..., b_m)$ are computationally indistinguishable, where the r_i's are independently drawn from R_n, each x_i is such that $f_\alpha(x_i) = S(\alpha; r_i)$, and the b_i's are independent uniformly distributed bits.*

By setting $k = m + 1$, it follows that, for every polynomial k, if the TDP is doubly-enhanced, then the EGL protocol (of Figure 2) securely implements 1-out-of-k OT in the semi-honest model.

Proof. For $m = 1$, the claim follows by the definition of enhanced hardcore predicate (i.e., Definition 2.2). For $m > 1$, we use a hybrid argument. Given α and (r, z), where $r \leftarrow R_n$ and $z \in \{0, 1\}$, we select uniformly $i \in [m]$, generate $i - 1$ pairs $(r_1, x_1), ..., (r_{i-1}, x_{i-1})$ such that $f_\alpha(x_j) = S(\alpha; r_j)$ for every $j \in [i - 1]$, compute $b_j = h(x_j)$ for every $j \in [i - 1]$, select $b_{i+1}, ..., b_m$ uniformly in $\{0, 1\}$ (and $r_{i+1}, ..., r_m$ from R_n), and produce the sequence

$$(\alpha, r_1, ..., r_{i-1}, r, r_{i+1}, ..., r_m, b_1, ..., b_{i-1}, z, b_{i+1}, ..., b_m).$$

Now, the indistinguishability of neighboring hybrids follows from the hypothesis that h is an enhanced hardcore predicate, wheraes the extreme hybrids correspond to the desired conclusion. ∎

4.2 Noninteractive Zero-Knowledge Proofs

As a second example, we consider a construction of non-interactive zero-knowledge proofs for any $\mathcal{NP}$ language. Recall that zero-knowledge proofs allow a prover to convince a verifier that a given statement is valid without disclosing any additional information other than the validity of the statement [21]. Non-interactive zero-knowledge proof systems (NIZK), introduced by Blum, Feldman, and Micali [4], are zero-knowledge proofs in which there is no actual interaction; that is, a single message is sent from the prover to the verifier, which either accepts or rejects. Instead of bi-directional interaction, a setup assumption is used; specifically, the existence of a common random string, to which both parties have (read-only) access. For a definition of NIZK proofs see Appendix A.

Assuming the existence of one-way permutations, Feige, Lapidot, and Shamir [9] constructed NIZK proof-systems for any $\mathcal{NP}$ language. They also offer an efficient implementation of the prescribed prover, by using an idealized TDP. We refer to this construction as the FLS protocol, and consider what happens when it is implemented when using a general TDP (and the two enhancements of this notion).

There are two gaps when trying to replace idealized TDP in the FLS protocol with general TDP. The first gap (discovered by Bellare and Yung [2]) is that the soundness of the FLS construction relies on the feasibility of recognizing permutations in the collection. We start by elaborating on this gap, while noting that the solution will lead us to the second gap.

Proving that a function is 1-1. In the FLS protocol the prover provides the verifier an index α of a permutation in the collection, and the soundness is based on the assumption that α does indeed describe a permutation. This assumption always holds in the case of idealized TDP (where any index describes a permutation), but for all popular candidate TDPs it is unknown how to efficiently check whether a given string describes a valid permutation (cf. [2]). Therefore, when the FLS protocol is implemented with general TDP, a cheating prover may provide a string that does not correspond to any permutation (but rather describes a many-to-one function, which in turn may be used to violate the soundness condition). Bellare and Yung suggested to resolve this problem by augmenting the main NIZK, with a (non-interactive zero-knowledge) proof that the given index α does indeed describe a function that is practically a permutation. This is done by presenting sufficiently many domain elements (described as part of the common random string) and expecting the prover to provide the inverses of these elements (where the validity of these preimages can be checked by applying the function f_α in the forward direction). Soundness follows from the fact that if the function is not (almost) 1-1, then, with high probability, a cheating prover will not be able to supply preimages for random domain elements. Unfortunately, it turns out that this protocol is not necessarily zero-knowledge when using general (or singly enhanced) TDP, but it is zero-knowledge in case the domain of f_α equals $\{0,1\}^{|\alpha|}$.

The reason that the foregoing protocol may not be zero-knowledge is the essence of the second aforementioned gap, and we shall discuss this gap now, while first detailing the foregoing proof system. In this proof system, both parties are given an index α (which allegablly describes a permutation in the collection), and the prover is also given a corresponding trapdoor. Both parties have access to a common random string that is partitioned into ℓ strings, denoted $r_1, \ldots, r_\ell$, each of length that fits the number of coins used by $S(\alpha)$. The prover uses the random strings to obtain ℓ elements, $y_1, \ldots, y_\ell$ such that $y_i = S(\alpha; r_i)$, inverts them (using the trapdoor) to obtain $x_1, \ldots, x_\ell$ such that $x_i = f_\alpha^{-1}(y_i)$, and sends $x_1, \cdots, x_\ell$ to the verifier. The verifier computes by itself $y_i = S(\alpha; r_i)$ and verifies that $y_i = f_\alpha(x_i)$, for all $i \in \{1, \ldots, \ell\}$. Although this may convince the verifier that f_α is almost 1-1 (i.e., if n/ℓ fraction of D_α has no preimage under f_α, then the verifier will reject with overwhelming high probability), but it may not be zero-knowledge in general. The point is that the verifier obtains a random pair (x, r) such that $f_\alpha(x) = S(\alpha; r)$ (actually it gets many such pairs), and it is not clear that the verifier could have generated such a pair (let alone many such pairs) by itself. This concern remains valid if the collection is an enhanced TDP, but it disappears by the assumption that the collection is doubly-enhanced (which indeed is tailored for such applications).

General NIZK. We mention that the difficulty encountered in the foregoing protocol (for proving that a function is 1-1) also presents itself in the basic FLS protocol. Specifically, the FLS verifier sees random pairs of the form (x, r) such that $f_\alpha(x) = S(\alpha; r)$ also in the basic FLS protocol (i.e., before its augmentation by [2]). Again, as above (and as in the case of the EGL OT protocol for $k \geq 3$), the difficulty is resolved by using doubly-enhanced TDP. In such a case, by definition, the ability to efficiently generate random (x, r) such that $f_\alpha(x) = S(\alpha; r)$ is guaranteed, and the zero-knowledge property of the procotol follows.

5 Obliviously Sampling Ciphertexts

In this section we consider the standard construction of public-key encryption based on trapdoor permutations [20, 24]. This construction is indeed secure given *any* TDP (i.e., no enhancement is necessary), but there are natural properties that are guaranteed only when the TDP is singly or doubly enhanced. Specifically, we refer to the ability to generate ciphertexts *obliviously* of the plaintext. We focus on public-key schemes for which this property means that, given the encryption-key, it is feasible to sample an encryption of a random message such that even the generator itself does not know the message (assuming that it does not have the decryption-key). In contrast, the trivial sampler that chooses a random message and encrypts it is inherently non-oblivious.

We use the standard definition of a public-key encryption scheme, except that we allow a restriction of the message space. Recall that such a scheme is described in terms of three algorithms (i.e., key-generation, encryption and decryption). For sake of simplicity, we assume that the message space consists of all strings of length $\ell(n)$, where n is the security parameter and ℓ is a polynomially-bounded function. Typical cases are $\ell \equiv 1$ (i.e., bit encryption) and $\ell(n) = n$.

Definition 5.1 (oblivious ciphertext sampleablity)**.** *We say that a public-key encryption scheme* (G, E, D) *is an oblivious ciphertext sampleable (OCS) scheme if there exists a probabilistic polynomial-time algorithm* O (called the oblivious ciphertext sampling algorithm) *such that the following holds:*

1. *For any encryption-key* e *the output of* $O(e)$ *is distributed identically to a random encryption of a random message; that is,* $O(e)$ *is distributed identically to* $E_e(U)$ *where* U *is distributed uniformly in the message space.*[6]

2. *Given the encryption-key* e *and the random coins of the sampler* r*, the value* $D_d(O(e; r))$ *is pseudorandom; that is,* $(e, r, D_d(O(e; r)))$ *and* (e, r, U) *are computationally indistinguishable, when* e, r *and* U *are random.*

[6] A natural relaxation would require the distributions to be statistically close or even just computationally indistinguishable to an adversary that has the decryption-key.

Note that the essence of the obliviousness condition is captured in the second item, which asserts that the plaintext looks random even when the coins used to produce the ciphertext are known. We mention that OCS encryption schemes were considered by Gertner *et al.* [10],[7] who showed that they can be used to construct a 3-round OT protocol.

5.1 On constructing public-key OCS schemes

As a concrete example of an OCS scheme consider the standard construction of public-key bit encryption from TDP. Recall that in this construction the encryption-key is an index α of a TDP, the decryption-key is the corresponding trapdoor τ, and the message space is $\{0,1\}$. Using the encryption-key α, the bit $\sigma \in \{0,1\}$ is encrypted by selecting a random domain element $x \leftarrow S(\alpha)$, and outputting $(f_\alpha(x), h(x) \oplus \sigma)$, where h is a hardcore predicate of the TDP. The ciphertext (y, b) is decrypted via the decryption-key τ by outputting $h(f_\alpha^{-1}(y)) \oplus b$ (where f_α is inverted using τ).

At first glance, it seems that, in this scheme, it is easy to sample ciphertexts obliviously of the plaintexts, since an encryption of a random message is uniformly distributed in $D_\alpha \times \{0,1\}$. Specifically, consider the algorithm $O(\alpha)$ that outputs (y, b) such that $y \leftarrow S(\alpha)$ and b is a random bit. While the sampler O clearly outputs the right distribution (and so satisfies the first item of Definition 5.1), it is not necessarily oblivious (i.e., it does not necessarily satisfy the second item). The random coins of O include the random coins that are used to produce the sample $y \in D_\alpha$, and therefore, as shown in Section 3, when using a *general* TDP, it may be possible (using these random coins) to invert f_α on y, and so retreive the plaintext. Hence the suggested sampling algorithm may not satisfy the conditions of Definition 5.1.

To resolve this issue we yet again use enhanced TDP (and assume that h is an enhanced hardcore predicate). Indeed, if the TDP is enhanced, then the foregoing O is an oblivious ciphertext sampler (i.e., it satisfies the conditions of Definition 5.1).[8]

5.2 Sampling Multiple Ciphertexts Obliviously

Consider extending the notion of obliviously sampling a *single* ciphertext to sampling *multiple* ciphertexts obliviously. Informally a k-OCS public-key encryption scheme is one in which it is feasible given the encryption-key e to sample from the joint distribution $E_e(m_1) \times \cdots \times E_e(m_k)$ such that $m_1, \ldots, m_k$ are (1) uniformly distributed in the message space and (2) pseudorandom even given the random coins of the sampler (although they are information-theoretically determined).

Intuitively, it may seem that any regular OCS scheme (i.e., a 1-OCS scheme) directly yields a k-OCS scheme, by merely invoking the oblivious sampling algorithm k times. Clearly, these k samples will be distributed correctly, but these samples may not be pseudorandom given the random coins of the sampler. That is, while each individual message m_i is guaranteed to be pseudorandom, the joint distribution $(m_1, \ldots, m_k)$ is not necessarily pseudorandom. To see this we return to the construction of public-key bit-encryption based on enhanced TDP discussed above. Even for $k = 2$ the suggested sampler is not necessarily oblivious. This follows from reasons similar to those

[7] Gertner *et al.* [10] refer to such schemes as to having Property B.

[8] To prove this it suffices to show that given $\alpha, (r, b)$ it is infeasible to predict $D_\tau(O(\alpha; (r, b))$ with non-negligible advantage. Note that $D_\tau(O(\alpha; (r, b))) = D_\tau(y, b) = h(f_\alpha^{-1}(y)) \oplus b$ where $y = S(\alpha; r)$ so an adversary that predicts $D_\tau(O(\alpha; (r, b)))$ can be easily converted to an adversary for h the enhanced hardcore predicate that on input α, r predicts $h(f_\alpha^{-1}(y))$.

discussed in Section 4 and the difficulty can be resolved similarly by using a *doubly-enhanced* TDP. Details follow.

Consider the standard TDP based encryption scheme and the corresponding oblivious sampler O (outlined in Section 5.1). Let O^2 denote the direct product of O; that is, $O^2(\alpha)$ selects two random elements $y_1, y_2 \in D_\alpha$ and two random bits $b_0, b_1 \in \{0,1\}$ and outputs $((y_1, b_1), (y_2, b_2))$. The output of O^2 is indeed distributed identically to a pair of encryptions of independent random bits. However, the random string used by O^2 is (r_1, b_1, r_2, b_2) where r_1 and r_2 are the random strings that respectively sample y_1 and y_2 (i.e., $y_i = S(\alpha; r_i)$). In Section 4 we showed that for an enhanced TDP, given α, r_1 and r_2, it may be feasible to compute $h(x_1) \oplus h(x_2)$ where $x_i = f_\alpha^{-1}(y_i)$. Since the two plaintext bits (corresponding to ciphertexts (y_1, b_1) and $y_2, b_2)$ are respectively "masked" by $h(x_1)$ and $h(x_2)$, the random string used by O^2 may reveal whether the two plaintexts are equal or not.

As mentioned above, the difficulty can be resolved by using a *doubly-enhanced* TDP. When using a doubly-enhanced TDP, for any polynomial $k = k(n)$, the sampler that outputs $(y_1, b_1), \ldots, (y_k, b_k)$, where $y_1, \ldots, y_k$ are random domain elements and $b_1, \ldots, b_k$ are random bits, is a k-oblivious sampler for the TDP based public-key encryption scheme. This fact follows from Claim 4.1, which states that even given the random strings $r_1, \ldots, r_k$, which are used to sample $y_1, \ldots, y_k$ respectively, the bits $h(x_1), \ldots, h(x_k)$, where $x_j = f_\alpha^{-1}(y_j)$, are pseudorandom. Thus, given $r_1, \ldots, r_k$, the k plaintext bits that correspond to the k ciphertexts $(y_1, b_1), \ldots, (y_k, b_k)$, are also pseudorandom since they are respectively "masked" by the pseudorandom bits $h(x_1), \ldots, h(x_k)$.

6 Intermediate Notions

So far we have mainly considered general TDP, enhanced TDP, doubly-enhanced TDP and idealized TDP. In this section we present a few intermediate notions. We first consider the realm between doubly-enhanced TDP and idealized TDP and then an intermediate notion between enhanced and doubly-enhanced TDP.

6.1 Between Doubly-Enhanced and Idealized TDP

Recall that idealized TDP are TDP which have domain $\{0,1\}^{|\alpha|}$ (and therefore have a trivial sampler) *and* for which the set of indices with respect to security parameter n are $\{0,1\}^n$. The first relaxation that we discuss refers to dropping the latter requirement:

Definition 6.1. *A TDP is called a* full-domain *TDP if for every index $\alpha \leftarrow I_1(1^n)$ it holds that $D_\alpha = \{0,1\}^{|\alpha|}$.*

This definition as the subsequent three definitions were mentioned in passing in Section 2. Assuming, for simplicity, that $D_\alpha \subseteq \{0,1\}^{|\alpha|}$, we consider a relaxation of Definition 6.1 by allowing domains that are either dense and/or efficiently recognizable.

Definition 6.2. *A TDP is* dense *if for every index $\alpha \leftarrow I_1(1^n)$ it holds that $|D_\alpha| \geq \frac{2^{|\alpha|}}{\mathrm{poly}(\alpha)}$.*

Definition 6.3. *A TDP has an* efficiently recognizable domain *if it is possible to efficiently check, given an index α and a string $x \in \{0,1\}^{|\alpha|}$, whether $x \in D_\alpha$.*

We note that given a TDP with dense *and* efficiently recognizable domain, one can construct a full-domain TDP. This is done in two steps: First, we construct a full-domain *weak* TDP (in the sense of weak one-way functions), and then we apply the transformation from weak one-way functions to strong one-way functions (see [13, Theorem 2.3.2]) to obtain a full-domain (strong) TDP.[9] Lastly we recall an additional relaxation discussed in Section 2:

Definition 6.4. *A TDP is said to have a* reversed domain-sampler *if there exists a probabilistic polynomial-time algorithm that on input an index α and a domain element $y \in D_\alpha$ outputs a string that is uniformly distributed in $\{r : S(\alpha; r) = y\}$).*

As mentioned in Section 2, any TDP that has a reversed domain-sampler is doubly-enhanced. The first enhancement follows by using the reversed domain-sampler to reduce the standard inverting task to the enhanced-inverting task (i.e., given (α, y), we invoke the enhanced-inverter on input (α, r) where r is random subject to $S(\alpha; r) = y$). For the second enhancement, we can sample a random (x, r) such that $f_\alpha(x) = S(\alpha; r)$ by selecting a random element $x \in D_\alpha$, computing $y = f_\alpha(x)$, and using the reversed domain-sampler to obtain r.

6.2 Between Enhanced and Doubly-Enhanced TDP

In some of the protocols discussed above we used the doubly-enhanced property to argue that many hardcore bits of a *doubly-enhanced* TDP are pseudorandom in the enhanced settings. Although we have an example for an enhanced TDP whose enhanced hardcore bits are not pseudorandom it may be possible to *transform* any enhanced TDP to one whose hardcore bits are pseudorandom. The following theorem takes a step in this direction by showing that upto *logarithmically* many hardcore bits of a specific enhanced hardcore predicate are pseudorandom.

Theorem 6.5. *Let $\{f_\alpha \colon D_\alpha \to D_\alpha\}_\alpha$ be an enhanced TDP where $D_\alpha \subseteq \{0,1\}^{|\alpha|}$ and let $\{g_\alpha \colon D_\alpha \times \{0,1\}^{|\alpha|} \to D_\alpha \times \{0,1\}^{|\alpha|}\}_\alpha$ be the enhanced TDP defined as $g_\alpha(x, s) = (f_\alpha(x), s)$ where $|x| = |s| = |\alpha|$. Then, for the* GL enhanced hardcore predicate *of $\{g_\alpha\}_\alpha$, defined as $h(x, s) \stackrel{\text{def}}{=} \langle x, s\rangle = \sum_{i=1}^{n} x_i s_i \bmod 2$, logarithmically many hardcore bits are pseudorandom. That is, for any $k = O(\log|\alpha|)$ the following ensembles are computationally indistinguishable:*

- *$\{(h(x_1, s_1), \ldots, h(x_k, s_k)), ((r_1, s_1), \ldots, (r_k, s_k))\}_\alpha$ where each pair (r_i, s_i) is a uniform random strings of the domain sampler of g and $x_i = f_\alpha^{-1}(S(\alpha; r_i))$.*
- *$\{(\sigma_1, \ldots, \sigma_k), ((r_1, s_1), \ldots, (r_k, s_k))\}_\alpha$ where each pair (r_i, s_i) is a uniform random strings of the domain sampler of g and each σ_i is a uniformly random bit.*

To proof Theorem 6.5 we show that given an index α of a permutation and $k = O(\log n)$ random strings $(r_1, s_1), \ldots, (r_k, s_k)$ of the domain sampler S_g (the sampling algorithm of $\{g_\alpha\}_\alpha$), it is infeasible to approximate $\bigoplus_{j\in U} b_j$, for any non-empty set $U \subseteq [k]$ (where $b_j = h(x_j, s_j)$ and $x_j = f_\alpha^{-1}(S(\alpha; r_j))$). The theorem follows by applying the computational XOR lemma for hardcore functions [13, Lem. 2.5.8]. (This XOR lemma asserts that if it is infeasible to approximate the parity of a random subset of logarithmically many hardcore bits, then these bits are pseudorandom.)

[9] For the first step suppose that we have a TDP $\{f_\alpha\}_\alpha$ with dense and efficiently recongizable domains. We consider a new TDP $\{f'_\alpha\}_\alpha$ that is defined by letting $f'_\alpha(x) = f_\alpha(x)$ if $x \in D_\alpha$ and $f'_\alpha(x) = x$ otherwise (i.e., if $x \notin D_\alpha$). Note that $\{f'_\alpha\}_\alpha$ is an efficiently computable permutation (since the domains are efficiently recongizable), and that it is weakly one-way due to the density of the domains and the one-wayness of $\{f_\alpha\}_\alpha$.

Proposition 6.6. *Let $k = k(n)$ be $O(\log n)$. For any probabilistic polynomial-time algorithm A and any non-empty set $U \subseteq [k]$, it holds that*

$$\Pr_{\substack{(\alpha,\tau)\leftarrow I(1^n) \\ (r_1,s_1),\ldots,(r_k,s_k)\leftarrow\{0,1\}^{\mathrm{poly}(|\alpha|)}\times\{0,1\}^{\mathrm{poly}(|\alpha|)}}}\left[A\left(\alpha,(r_1,s_1),\ldots,(r_k,s_k),U\right) = \bigoplus_{j\in U} b_j\right] = \frac{1}{2} + \mu(n) \tag{5}$$

where $b_j \stackrel{\text{def}}{=} h(x_j, s_j) = \langle x_j, s_j\rangle$ and $x_j \stackrel{\text{def}}{=} f_\alpha^{-1}(S(\alpha; r_j))$ (and μ is a generic negligible function).

Proof. Assume toward a contradiction that this is not the case. That is, there exists a non-empty set $U \subseteq [k]$ and an algorithm A that has a non-negligible advantage in approximating $\bigoplus_{j\in U} b_j$ based on α, U and $(r_1,s_1),\ldots,(r_k,s_k)$. Furthermore, such a set U can be found in probabilistic polynomial-time by experimenting with all possible sets (while generating random samples of $I(1^n)$). Fixing such a set $U = \{j_1,\ldots,j_{k'}\}$, we observe that $\bigoplus_{j\in U} b_j = \bigoplus_{j\in U}\langle x_j, s_j\rangle$ equals $\langle x_{j_1}\circ\cdots\circ x_{j_{k'}}, s_{j_1}\circ\cdots\circ s_{j_{k'}}\rangle$, which is the GL hardcore predicate of g'_α that is defined by the direct produce of k' values of g_α (i.e., $g'_\alpha((x_1,s_1),\ldots,(x_k,s_k)) = (g_\alpha(x_1,r_1),\ldots,g_\alpha(x_k,s_k))$). Thus, it suffices to note that g'_α is an enhanced trapdoor permutation,[10] and the result of [17] (cf. [13, Sec. 2.5.2] or better [15, Thm. 7.8]) implies that the said predicate is indeed an enhanced hardcore. ■

Corollary. Theorem 6.5 implies that, when using the GL hardcore predicate, the extended EGL protocol is secure for any logarithmically bounded k.

Further Notions. Theorem 6.5 refers to the pseudorandomness of the sequence $h(x_1,s_1),...,h(x_k,s_k)$ relative to $(\alpha,(r_1,s_1),...,(r_k,s_k))$, where $x_i \stackrel{\text{def}}{=} f_\alpha^{-1}(S(\alpha;r_i))$. Alternative notions that refer to the unpredictability of related sequences arise naturally. The interested reader is referred to [23] for a taxonomy of notions of TDP that lie between enhanced and doubly-enhanced.

[10] Note that here we merely claim that direct product preserves (rather than amplifies) hardness-to-invert. Indeed, if given $\alpha,(r_1,s_1),\ldots,(r_k,s_k)$ it is feasible to compute $x_1,\ldots,x_k$, then it particular it is feasible to compute x_1 from $\alpha,(r_1,s_1)$.

References

[1] W. Alexi, B. Chor, O. Goldreich and C.P. Schnorr. RSA/Rabin Functions: Certain Parts are As Hard As the Whole. *SIAM Journal on Computing*, Vol. 17, April 1988, pages 194–209. Preliminary version in *25th FOCS*, 1984.

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

[3] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 20, No. 6, pages 1084–1118, 1991. (Considered the journal version of [4].)

[4] M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and its Applications. In *20th ACM Symposium on the Theory of Computing*, pages 103–112, 1988. See [3].

[5] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme which hides all partial information. In *Crypto84*, Lecture Notes in Computer Science (Vol. 196), Springer-Verlag, pages 289–302.

[6] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

[7] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22 (Nov. 1976), pages 644–654.

[8] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, Vol. 28, No. 6, 1985, pages 637–647. Extended abstract in *Crypto'82*.

[9] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, Vol. 29 (1), pages 1–28, 1999. Preliminary version in *31st FOCS*, 1990.

[10] Y. Gertner, S. Kannan, T. Malkin, O. Reingold and M. Viswanathan. The Relationship between Public Key Encryption and Oblivious Transfer. Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS), 2000.

[11] O. Goldreich. A Uniform Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, Vol. 6, No. 1, pages 21–53, 1993.

[12] O. Goldreich. *Secure Multi-Party Computation.* Available from the author's homepage, 1998 (revised 2001).

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

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

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

[16] O. Goldreich. Basing Non-Interactive Zero-Knowledge on (Enhanced) Trapdoor Permutations: The State of the Art. To appear in Lecture Notes in Computer Science (Vol. 6650), Springer. Available from the author's homepage, Nov. 2008.

[17] O. Goldreich and L.A. Levin. Hard-core Predicates for any One-Way Function. In *21st ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[18] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[19] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[20] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.

[21] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[22] I. Haitner. Implementing Oblivious Transfer using a Collection of Dense Trapdoor Permutations. In *1st Theory of Cryptography Conference*, Lecture Notes in Computer Science (Vol. 2951), Springer, 2004.

[23] R. Rothblum. A Taxonomy of Enhanced Trapdoor Permutations. *ECCC*, TR10-145, September 2010.

[24] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

A Definitions of OT and NIZK

For sake of simplicity, we present a non-uniform formulation of all definitions; that is, the inputs to the protocols are quantified over all possibilities. Thus, constructing such protocols may require non-uniformly hard TDP. Uniform-complexity formulations can be derived by considering only polynomial-time sampleable inputs (cf., [11] or [14, Sec. 5.2.5]).

A.1 Oblivious Transfer

Here $k = k(n)$ is a polynomially bounded function. A pair of probabilistic polynomial-time strategies (S, R) constitute a 1-out-of-k OT protocol (in the semi-honest model) if the following conditions hold.

- **Correctness:** For every $\sigma_1, \ldots, \sigma_k \in \{0, 1\}$ and $i \in \{1, \ldots, k\}$, when $S(1^n, \sigma_1, \ldots, \sigma_k)$ interacts with $R(1^n, i)$ the receiver R outputs σ_i and the sender S outputs nothing.
- **Receiver security:** There exists a probabilistic polynomial-time simulator S_1 such that the following two probability ensembles are computationally indistinguishable.
  1. $\{S_1(\sigma_1, \ldots, \sigma_k, 1^n)\}_{n\in\mathsf{N},\sigma_1,\ldots,\sigma_k,i}$ and
  2. $\{\mathtt{view}_1(\sigma_1, \ldots, \sigma_k, i, 1^n)\}_{n\in\mathsf{N},\sigma_1,\ldots,\sigma_k,i}$, where $\mathtt{view}_1$ is a random variable that consists of the sender's view in the interaction (i.e. its input, randomness and received messages).
- **Sender security:** There exists a probabilistic polynomial-time simulator S_2 such that the following two probability ensembles are computationally indistinguishable.
  1. $\{S_2(i, 1^n, \sigma_i)\}_{n\in\mathsf{N},\sigma_1,\ldots,\sigma_k,i}$ and
  2. $\{\mathtt{view}_2(\sigma_1, \ldots, \sigma_k, i, 1^n)\}_{n\in\mathsf{N},\sigma_1,\ldots,\sigma_k,i}$, where $\mathtt{view}_2$ is a random variable that consists of the receiver's view in the interaction (i.e. its input, randomness and received messages).

A.2 Non-Interactive Zero-Knowledge Proofs

A pair of probabilistic polynomial-time algorithms (P, V) constitute an (efficient-prover) non-interactive zero-knowledge proof for an $\mathcal{NP}$ language L with the witness relation R_L if the following conditions hold.

- *Completeness:* For every $x \in L$ and every witness w such that $(x, w) \in R_L$,
$$\Pr_{r\in\{0,1\}^{\mathrm{poly}(|x|)}}[V(x, r, P(x, w, r)) = 1] \geq \frac{2}{3}.$$
- *Soundness:* For every $x \notin L$ and every cheating strategy P^*,
$$\Pr_{r\in\{0,1\}^{\mathrm{poly}(|x|)}}[V(x, r, P^*(x, r)) = 1] \leq \frac{1}{3}.$$
- *Zero-Knowledge:* There exists a probabilistic polynomial-time simulator M such that the following two probability ensembles are computationally indistinguishable.
  1. $\{M(x)\}_{x\in L,w\in R_L(x)}$, where $R_L(x) = \{w : (x, w) \in R_L\}$, and
  2. $\{(x, R_{|x|}, P(x, w, R_{|x|}))\}_{x\in L,w\in R_L(x)}$, where R_n denotes a random variable uniformly distributed over $\mathrm{poly}(n)$.

B On the RSA and Rabin Collections

In this appendix we show that suitable versions of the RSA and Rabin collections satisfy the two aforementioned enhancements (presented in Definitions 2.1 and 2.3, 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.

B.1 The RSA collection satisfies both enhancements

We start our treatment by considering the RSA collection (as presented in [13, Sec. 2.4.3.1] and further discussed in [13, Sec. 2.4.3.2]). Note that in order to discuss the enhanced hardness condition (of Def. 2.1) 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 $\ell \stackrel{\text{def}}{=} 2\lfloor \log_2 N \rfloor$ 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 2.1 (as well as Definition 2.3) 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 $i \leftarrow j \cdot N + e$.

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

In contrast to the case of the RSA, the Rabin Collection (as defined in [13, Sec. 2.4.3.3]), does not satisfy Definition 2.1 (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 a *doubly-enhanced* collection of trapdoor permutations, provided that factoring is hard (in the same sense as assumed in [13, Sec. 2.4.3]).

The modification is based on modifying the domain of these permutations (following [1]). 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). Lastly, note that 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 $\{f_\alpha : D_\alpha \to D_\alpha\}_{\alpha \in \bar{I}}$, such that $D_\alpha \subseteq \{0,1\}^{|\alpha|}$ is dense (i.e., $|D_\alpha| > 2^{|\alpha|}/\text{poly}(|\alpha|)$)[11] 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 most 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. 2.1), and noting that this sampler has an efficient reversed sampler (which implies that Def. 2.3 is satisfied).

   The domain-sampler that we have in mind repeatedly selects random (i.e., uniformly distributed) $|\alpha|$-bit long strings and outputs 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 the string x.

2. An alternative way of obtaining a doubly-enhanced collection is to first define a (rather artificial) collection of *weak* trapdoor permutations, $\{f'_\alpha : \{0,1\}^{|\alpha|} \to \{0,1\}^{|\alpha|}\}_{\alpha \in \bar{I}}$, 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 [13, 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 [14, 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 [14, Apdx. C.1] is not doubly-enhanced.

C An Enhanced TDP whose Hardcore Bits are not Pseudorandom

In this section we show that a variant of the factoring based enhanced TDP presented in Appendix B.2, has an enhanced hardcore predicate for which two or more samples are not pseudorandom.

Notation. For a Blum integer N, let J_N be the set of all elements in Z_N^* that have Jacobi symbol $+1$ modulo N and let $M_N \stackrel{\text{def}}{=} J_N \cap \{1, \ldots, \lfloor \frac{N}{2} \rfloor\}$. For $x \in Z_N^*$, let $QR_N(x)$ be 1 if x is quadratic residue (modulo N) and 0 otherwise.

Construction C.1. *(A factoring-based enhanced TDP)*

$I(1^n)$: *Let $N = PQ$ where P and Q are two uniformly selected primes such that $2^{n-1} \le P, Q \le 2^n$ and $P \equiv Q \equiv 3 \bmod 4$. Select a random element $y \in J_N$ and output (N, y) as the index and (P, Q) as the trapdoor.*

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

sampler $S(N, y)$: *Select, uniformly at random* $r \in Z_N^*$, *and let* $z = y \cdot r^2 \bmod N$. *If* $z \leq \lfloor \frac{N}{2} \rfloor$, *output* z *and otherwise output* $N - z$.

$F((N, y), x)$: *Set* $z = x^2 \bmod N$. *If* $z \leq \lfloor \frac{N}{2} \rfloor$ *output* z *and otherwise output* $N - z$.

$B((N, y), x)$: *Given the factorization of* N, *it is possible to compute square roots modulo* N *and to invert this permutation (for details see [14, Sec. 2.4.4.2]).*

Note that Construction C.1 is almost the same as the enhanced TDP of Appendix B.2, where the only difference is in how elements are sampled in the domain M_N (and the augmentation of the index that is used for that purpose). In particular, the evaluation and inversion algorithms remain the same, and therefore, as discussed in Appendix B.2, the function F_N is a permutation over M_N. Additionally, the sampling algorithm $S(N, y)$ produces a uniformly distributed element in M_N, since $S(N, y)$ induces a 4-to-1 mapping from its random strings to M_N.

We proceed to show an enhanced hardcore predicate for Construction C.1 (which implies, in particular, that the TDP is enhanced). Specifically, we show that the predicate $h_{N,y}(x) \stackrel{\text{def}}{=} QR_N(F_{N,y}(x))$ (i.e., the predicate that equals 1 if the image of x under $F_{N,y}$ is a quadratic residue and 0 otherwise) is an enhanced hardcore predicate.

Claim C.2. *Assuming the quadratic residuosity assumption*[12], *the predicate* $h_{N,y}$ *is an enhanced hardcore predicate of Construction C.1.*

Proof. Given x, the predicate $h_{N,y}$ is indeed easy to compute (i.e., if $F_{N,y}(x) = x^2 \bmod N$, then $h_{N,y}(x) = 1$, otherwise it must be that $F_{N,y}(x) = N - x^2 \bmod N$ which implies that $h_{N,y}(x) = 0$). What remains to be shown is that given (N, y) and r, it is infeasible to predict $QR_N(S(N, y; r))$. The key point is that multiplication by r^2 preserves quadratic residuosity whereas multiplication by $-r^2$ complements it (i.e., $y \cdot r^2$ is a quadratic residue if and only if y is a quadratic residue and $-y \cdot r^2$ is a residue if and only if y is a non-residue). Thus, given N, y and r it is easy to check whether y and $S(N, y; r)$ have the same QR_N value (i.e., compute $QR_N(y) \oplus QR_N(S(N, y; r))$), by checking whether S multiplies y by r^2 or by $-r^2$. Thus, an adversary that computes $QR(S(N, y; r))$ can be used to compute $QR_N(y)$. Details follow.

Consider an adversary A that on input (N, y) and r, breaks the hardcore predicate by outputting $QR_N(S(N, y; r))$ with probability $\frac{1}{2} + \epsilon$. We use A to construct an adversary A' to the quadratic residuosity problem as follows. The adversary A' is given N and y and needs to compute $QR_N(y)$. To do so A' selects uniformly at random $r \in Z_N^*$, computes $b = QR_N(y) \oplus QR_N(S(N, y; r))$ and outputs $A((N, y), r) \oplus b$. With probability $\frac{1}{2} + \epsilon$ the output of A' equals $QR_N(S(N, y; r)) \oplus \Big(QR_N(y) \oplus QR_N(S(N, y; r))\Big)$ which in turn equals $QR_N(y)$. ■

Thus, based on the quadratic residuosity assumption, the predicate $h_{N,y}$ is an enhanced hardcore predicate. However, we argue that the enhanced hardcore bits are not pseudorandom. Specifically, we show that, given the index (N, y) and two random strings r_1 and r_2, it is easy to check whether the hardcore bits of the preimages of the elements sampled by r_1 and r_2 are equal or not. To do so, first compute $QR_N(y) \oplus QR_N(S(N, y; r_1))$ and $QR_N(y) \oplus QR_N(S(N, y; r_2))$, by checking whether S multiplies y by r_i^2 or by $-r_i^2$ (as above). Then, compute the exclusive-or of these two values, which yields $QR_N(S(N, y; r_1)) \oplus QR_N(S(N, y; r_2))$ (i.e., the exclusive-or of the two hardcore bits).

[12] The assumption states that given a random Blum integer N and a random element in J_N it is infeasible to decide whether the element is a quadratic residue or not (with non-negligible advantage).

Hence, the predicate $h_{N,y}$ is an enhanced hardcore predicate but is not pseudorandom (in the enhanced setting) for even two samples.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de