Certifying Trapdoor Permutations, Revisited *

Ran Canetti† Amit Lichtenberg‡

July 21, 2017

Abstract

The modeling of trapdoor permutations has evolved over the years. Indeed, finding an appropriate abstraction that bridges between the existing candidate constructions and the needs of applications has proved to be challenging. In particular, the notions of certifying permutations (Bellare and Yung, 96), enhanced and doubly enhanced trapdoor permutations (Goldreich, 04, 08, 11, Goldreich and Rothblum, 13) were added to bridge the gap between the modeling of trapdoor permutations and needs of applications.

We identify an additional gap between the current modeling of trapdoor permutations and their classic use in non-interactive zero-knowledge (NIZK) proof systems: Previous works implicitly assumed that it is easy to recognize elements in the domain, as well as uniformly sample from it, even for illegitimate function indices. To demonstrate this gap, we instantiate the Feige-Lapidot-Shamir NIZK protocol together with Bellare-Yung certification using the (Bitansky-Paneth-Wichs, 15) doubly-enhanced trapdoor permutation family, and show that this results in an unsound proof system.

We propose a general notion of certifiably injective doubly enhanced trapdoor functions, and show that it suffices for implementing the FLS paradigm. We then show two very different ways to realize this notion: One is via the traditional method of RSA/Rabin with the Bellare-Yung certification mechanism, and the other using indistinguishability obfuscation and injective pseudorandom generators. In particular the latter is the first candidate trapdoor permutation from assumptions other than factoring, that suffices for the FLS paradigm.

Keywords: Non-Interactive Zero-Knowledge, Trapdoor Permutations, Indistinguishability Obfuscation

*Research supported by the Check Point Institute for Information Security and ISF grant 1523/14.

†Boston University and Tel Aviv University. Member of CPIIS. Supported by the NSF MACS project. Email: canetti@bu.edu

‡Tel Aviv University. Email: amitlich@post.tau.ac.il

1 Introduction

In the late-1970s, Rivest, Shamir and Adelman [RSA78] and Rabin [Rab79] suggested functions which are easy to evaluate, easy to invert when given a suitable secret trapdoor key, but are presumably hard to invert when only given the function description without the trapdoor. Both of these constructions use the same source of computational hardness: the hardness of factoring. These constructions were later abstracted to a formal notion of trapdoor function [Yao82], which became one of the pillars of modern cryptography. In particular, trapdoor permutations were used as building blocks for public key encryption [Yao82, GM84, BG84], oblivious transfer [EGL85] and zero-knowledge protocols [FLS90].

Non-interactive zero knowledge (NIZK) protocols [BFM88] is perhaps the quintessential application for trapdoor permutations, in the sense that trapdoor permutations are the only general assumption that was known to imply NIZK with certain desirably properties (public-coin reference string, unconditional soundness), modulo recent developments sketched in the next paragraph. The idea, proposed by Feige, Lapidot and Shamir [FLS90] and later formalized by Goldreich [Gol04], is to first construct an unconditional NIZK protocol in the more abstract *hidden-bit model,* where both sides are given a random string which is fully exposed only to the prover. The prover can choose to expose to the verifier the values at specific indices of the prover's choice. They then suggest a general transformation, based on trapdoor permutations, which transforms any non-interactive zero-knowledge proof system in the hidden-bit model to one in the common random string model. The idea here is to treat the common random string as a sequence of blocks, where a block represents an image of a trapdoor function provided by the prover. The prover is able to select a subset of these images and invert them using the secret trapdoor. The verifier can validate that the pre-images it was given are correct by forward-evaluating the trapdoor function, but is unable to invert any other image due to the hardness of inverting the function without the secret trapdoor. Soundness is based on the fact that, for any given permutation, each block in the reference string defines a unique pre-image. We refer to this protocol as the FLS protocol. Other NIZK protocols were later proposed, using the same hidden-bit model, e.g. [KP98].

Recently, [BP14] showed how to construct invariant signatures [BG90] from indistinguishability obfuscation and one-way functions. This, together with the technique of [GO92], gives a different path for realizing the hidden-bit model from assumptions other than factoring. (Previously, the only known construction of invariant signatures was from NIZK.) Still, the trapdoor-permutations-based paradigm of [FLS90] remains the textbook method for realizing non-interactive zero-knowledge proofs. See more discussion on alternative approaches for constructing non-interactive zero-knowledge proof systems at the end of the introduction.

Evolution of the *trapdoor permutations* abstraction. The work of [FLS90] is based on the assumption that the underlying trapdoor permutation is *ideal*, namely its domain is $\{0,1\}^n$ for some n, hardness holds with respect to uniformly chosen n-bit strings, and any key (index) in an efficiently recognizable set describes a permutation. As it turns out, this model is too idealized - so much so, that no realization of it is known. In particular, the two factoring-based candidates, which for many years stood as essentially the only candidates of trapdoor permutations, do not realize this idealistic modeling. Consequently a number of relaxed notions of trapdoor permutations were proposed.

Bellare and Yung [BY96] consider relaxing the notion of trapdoor permutations to families where the domain is still $\{0,1\}^n$, but it is not known how to recognize whether a given index defines a permutation. They observe that in this case the soundness of the original FLS transform is no longer guaranteed, and suggest a NIZK protocol for certifying that a given index describes a permutation. Their protocol is based on the prover providing the verifier with pre-images of a set of random images, which are taken from the common reference string. We note that this protocol introduces a positive (albeit negligible) soundness error, even when the underlying protocol in the hidden-bit model is perfectly sound. We refer to this protocol as the Bellare-Yung protocol.

Goldreich [Gol04, Gol08] points out that when the domain of the permutation is comprised of elements of specific structure (and not just the full domain $\{0,1\}^n$), the FLS protocol might no longer work since, among other issues, it may not be known how to translate blocks of the reference string to elements in the domain whose pre-image is unknown. He then defines enhanced and doubly-enhanced trapdoor permutations, which require existence of an algorithm that samples elements from the domain, such that finding the pre-image of a sampled element is hard, even given the random coins used by the sampler. Goldreich and Rothblum [Gol11, GR13] then show how doubly-enhanced trapdoor permutations can be used for the FLS protocol, by letting the prover and verifier treat the common reference string as a sequence of random coins which they can input into the domain sampler to obtain random images in the permutation's domain. Specifically, they prove that the FLS protocol remains sound and zero-knowledge when using doubly-enhanced trapdoor permutations — as long as the verifier is able to efficiently detect if the function described by the key is a permutation. To verify that this is the case, the suggested using the doubly-enhanced domain sampler for sampling the images needed by the prover and verifier in the Bellare-Yung protocol. Finally, they observe that the RSA and Rabin trapdoor permutations are doubly enhanced.

Bitansky et. al. [BPW15] give a doubly-enhanced trapdoor permutation family with a very different flavor (and is the first one to not assume hardness of factoring). Their construction, based on sub-exponentially-secure indistinguishability obfuscation and one-way function, gives the public function index as an obfuscation of a key-dependent program which evaluates a permutation. The secret trapdoor is the key used by the program. The permutation's domain is a sparse subset of $\{0,1\}^n$ which depends on the permutation key,

with different keys resulting in different domains.[1] While the secret key is encoded in the permutation program, the verifier is given only an obfuscated version of it, which "hides" the key. This results in a partial-domain trapdoor permutation, where identifying whether a given string is in the domain for a specific choice of permutation key is hard given only the public index of the permutation. This stands in contrast to the case of RSA and Rabin, where it is easy to recognize whether a given element is in the domain of the function defined by a given index.

1.1 Our Contribution

We first provide a more complete treatment of the properties needed from the underlying trapdoor permutation for sound realization of the FLS paradigm. Next, we re-assert the adequacy of existing constructions, and propose a new and quite different one.

We start by demonstrating the following gap: We show that, when instantiated with the [BPW15] doubly enhanced trapdoor permutation family, the FLS protocol is unsound, even when combined with the [BY96] certification protocol. We attribute the loss of soundness to the fact that the notion of doubly enhanced trapdoor permutations does not make sufficient requirements on indices that were not legitimately generated.

We then investigate several ways to close this gap: First, we formulate an additional set of requirements that is met by the existing factoring-based candidates, and suffices for regaining soundness of the FLS paradigm when combined with the Bellare-Yung certification. However, these additional requirements (which we call *public domain*) are rather specific. In particular, assuming indistinguishability obfuscations and injective pseudorandom generators, we construct an injective trapdoor function family that suffices for sound realization of the FLS paradigm but is not public domain.

We then formulate a more general property, called *certifiable injectivity*, that suffices for FLS and encompasses all current candidates.

Unsoundness of FLS+BY with the [BPW15] trapdoor permutations: We instantiate the FLS+BY protocols using the the [BPW15] doubly enhanced trapdoor function family, whose domain is not efficiently recognizable. We demonstrate how a malicious prover could choose an index α which describes a many-to-one function, wrongly certify it as a permutation by having the sampler sample elements only out of a restricted domain D_α which is completely invertible, but then invert any image in D_α into two pre-images - one in D_α and another outside of it. The verifier cannot detect the lie since D_α is not efficiently recognizable.

[1] In fact, the sparseness of the domains in the [BPW15] construction is essential, in order to circumvent the impossibility result of [AS15].

Regaining the soundness of BY+FLS using Public-Domain Trapdoor Permutations: We prove that the BY+FLS combination regains its soundness when the following additional properties are met by the function family, with respect to *any index* (in particular illegitimate ones): First, the domain of the permutation, as well as the forward evaluation and sampling algorithms, should be well defined for any index. Second, there must exist an efficient algorithm that decides, given some string, whether it represents an element in that domain. Last, the domain sampler of the function family should guarantee an almost uniform sample out of that domain. If all three requirements are met by the trapdoor permutation family, we say that it is *public-domain*. We note that the RSA and Rabin trapdoor permutations are indeed public-domain, while the [BPW15] permutation is not.

Certifiable Injective Trapdoor Functions: We formulate a new notion of *Certifiable Injectivity*, which captures a general abstraction of certifiability for doubly-enhanced injective trapdoor functions. This notion requires the function family to be accompanied by algorithms for generation and verification of certificates for indices, along with an algorithm for certification of individual points from the domain. It is guaranteed that if the index certificate is verified then, except for negligible probability, randomly sampled range points have only a single pre-image that passes the pointwise certification. We show that certifiable injectivity suffices for the FLS paradigm.

We additionally suggest a strengthened notion of *Perfectly Certifiable Injectivity*, which guarantees that no point generated by the range sampler has two pre-images that pass the pointwise certification. We show that by implementing FLS using this notion, the resulting error in soundness is optimal, in that it is equal to the error incurred by implementing the FLS protocol with ideal trapdoor permutations.

Doubly Enhanced Perfectly Certifiable Trapdoor Functions from iO+: We construct a doubly-enhanced family of trapdoor functions which is perfectly certifiable injective. Our construction, inspired by the work of [SW13], is based on indistinguishability obfuscation and pseudorandom generators, and is perfectly certifiable injective under the additional assumption that the underlying pseudorandom generator is (a) injective and (b) its domain is either full, or efficiently sampleable and recognizable.

The public trapdoor index for our construction is an indistinguishability obfuscation of a circuit T_k which, on input $x \in \{0,1\}^n$, outputs $(x \oplus f_k(g(x)), g(x))$, where g is a length-doubling pseudorandom generator and f_k is a puncturable pseudorandom function. The private key is the PRF key k. We implement a doubly-enhanced range sampler for our construction, based on a simple re-randomization technique. The sampling algorithm evaluates an obfuscated circuit $\tilde{S} = iO(S_{k,w})$, which, given random coins r, takes $x = h_w(r)$ and outputs $T_k(x)$, where h_w is a length-preserving PRF. Using another round of re-randomization we augment our construction into a doubly-enhanced TDF. Our re-randomization technique can be applied to any trapdoor function with an efficiently sampleable domain to obtain a doubly-enhanced domain sampler, at the cost of using iO.

Finally, we show how using the assumption that the pseudorandom generator g is injective and that its domain is efficiently recognizable, we are able to provide a perfect pointwise certification algorithm for our trapdoor functions, proving it is perfectly certifiable injective. We then show how to construct such generators from standard assumptions (such as, e.g., hardness of discrete log). This makes our construction sufficient for NIZK.

1.2 More on Trapdoor Permutations and NIZK.

Other Applications of Trapdoor Permutations. The gap between ideal and general trapdoor permutations imposes a problem in other applications as well. [Rot10, GR13] discuss the security of the [EGL85] trapdoor-permutations-based 1-out-of-k oblivious transfer protocol, which breaks in the presence of partial-domain trapdoor functions, and show how doubly enhanced trapdoor functions can be used to overcome this. The concern of certifying keys is irrelevant in the oblivious transfer applications, as the parties are assumed to be trusted. Still, certifiability concerns apply whenever dishonesty of one or more of the parties is considered an issue, such as the case of interactive proofs and multi-party computation.

Alternative Approaches for Constructing NIZK. Over the years, additional approaches were suggested to obtaining non-interactive zero-knowledge proofs which are not based on the hidden-bit model. [GOS06] constructed non-interactive zero-knowledge proofs for circuit satisfiability with a short reference string, and non-interactive zero-knowledge arguments for any NP language. [GS08] constructed non-interactive zero-knowledge proofs for assumptions on bilinear groups. [GOS12] and [SW13] constructed non-interactive zero-knowledge arguments with a short reference string for any NP language. All of these protocols either use a structured CRS whose generation requires additional randomness that's trusted to never be revealed, or achieve zero-knowledge *arguments*, where the soundness holds only with respect to computationally bounded adversaries. This leaves the hidden-bit paradigm the only known way to achieve zero-knowledge proofs for any NP language in the uniform reference string model.

1.3 Paper Organization

In section 2 we define the basic notions used in the paper, and describe in general the FLS protocol for NIZK for NP from trapdoor permutations, including the enhancements suggested by Bellare-Yung [BY96] and Goldreich-Rothblum [Gol04, Gol08, Gol11, GR13]. In section 3 we demonstrate how the soundness of the FLS protocol may be compromised when using general TDPs, and discuss the additional assumptions required to avoid this problem. In section 4 we suggest the alternative notion of *certifiably injective* trapdoor functions, and use it to overcome the limitations of the FLS+BY combination and regain the soundness of the FLS protocol. In section 5 we construct a doubly-enhanced, certifiable

injective trapdoor function family based on indistinguishability obfuscation and injective pseudorandom generators.

2 Preliminaries

The cryptographic definitions in this paper follow the convention of modeling security against non-uniform adversaries. A protocol P is said to be secure against (non-uniformly) polynomial adversaries, if it is secure against any adversary $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$, such that each circuit A_λ is of size polynomial in λ.

2.1 Notations

For a PPT algorithm A which operates on input x, we sometimes denote $A(x; r)$ as the (deterministic) evaluation A using random coins r.

We use the notation $\Pr[E_1; E_2; ...; E_n; R]$ to denote the probability of the resulting event R given that the series of events $E_1, ..., E_n$ occurred, as an alternative to $\Pr[R : E_1, E_2, ..., E_n]$.

2.2 Puncturable Pseudorandom Functions

We consider a simple case of puncturable pseudorandom functions (PPRFs) where any PRF may be punctured at a single point. The definition is formulated as in [SW13], and is satisfied by the GGM PRF [GGM86, BW13, KPTZ13, BGI14].

Definition 2.1. (Puncturable PRFs). Let n, k be polynomially bounded length functions. An efficiently computable family of functions:

$$PRF = \{PRF_S : \{0,1\}^{n(\lambda)} \to \{0,1\}^\lambda : S \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N}\}$$

associated with an efficient (probabilistic) key sampler K_{PRF}, is a puncturable PRF if there exists a poly-time puncturing algorithm $Punc$ that takes as input a key S and a point x^* and outputs a punctured key $S^* = S\{x^*\}$, so that the following conditions are satisfied:

1. **Functionality is preserved under puncturing:** For every $x^* \in \{0,1\}^{n(\lambda)}$,

$$\Pr_{S\leftarrow K_{PRF}(1^\lambda)}[S^* = Punc(S, x^*); \forall x \neq x^* : PRF_S(x) = PRF_{S^*}(x)] = 1$$

2. **Indistinguishability at punctured points:** for any polysize distinguisher D there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, and any $x^* \in \{0,1\}^{n(\lambda)}$,

$$\Pr[D(x^*, S^*, PRF_S(x^*)) = 1] - \Pr[D(x^*, S^*, u) = 1] \leq \mu(\lambda)$$

where $S \leftarrow K_{PRF}(1^\lambda)$, $S^* = Punc(S, x^*)$, and $u \leftarrow \{0,1\}^\lambda$.

2.3 Indistinguishability Obfuscation

We define indistinguishability obfuscation (iO) with respect to a given class of circuits. The definition is formulated as in [BGI+01].

Definition 2.2. (Indistinguishability Obfuscation [BGI+01]). A PPT algorithm iO is said to be an indistinguishability obfuscator for a class of circuits $\mathcal{C}$, if it satisfies:

1. **Functionality:** for any $C \in \mathcal{C}$,

$$\Pr_{iO}[\forall x : iO(C)(x) = C(x)] = 1$$

2. **Indistinguishability:** for any poly-size distinguisher D there exists a negligible function μ, such that for any two circuits $C_0, C_1 \in \mathcal{C}$ that compute the same function and are of the same size λ:

$$\Pr[D(iO(C_0)) = 1] - \Pr[D(iO(C_1)) = 1] \leq \mu(\lambda)$$

   Where the probability is taken over the coins of D and iO.

2.4 1-1 TDFs and TDPs

Definition 2.3. (Trapdoor Functions). A family of one-way trapdoor functions, or TDFs, is a collection of finite injective functions, denoted $f_\alpha : \{D_\alpha \to R_\alpha\}$, accompanied by PPT algorithm I (*index*), S_D (*domain sampler*), S_R (*range sampler*) and two (deterministic) polynomial algorithms F (*forward evaluator*) and B (*backward evaluator* or *inverter*) such that the following condition holds:

1. On input 1^n, algorithm $I(1^n)$ selects at random an index α of a function f_α, along with a corresponding trapdoor τ. Denote $\alpha = I_0(1^n)$ and $\tau = I_1(1^n)$.
2. On input $\alpha = I_0(1^n)$, algorithm $S_D(\alpha)$ returns a random element from domain D_α.
3. On input $\alpha = I_0(1^n)$, algorithm $S_R(\alpha)$ returns a random image from the range R_α.
4. On input $\alpha = I_0(1^n)$ and any $x \in D_\alpha$, $F(\alpha, x) = f_\alpha(x)$.
5. On input $\tau = I_1(1^n)$ and any $y \in R_\alpha$, $B(\tau, y) = f_\alpha^{-1}(y)$.

The standard hardness condition refers to the difficulty of inverting f_α on a random image, sampled by S_R or by evaluating $F(\alpha)$ on a random pre-image sampled by S_D, when given only the image and the index α but not the trapdoor τ. That is, it is required that, for every polynomial-time algorithm A, it holds that:

$$\Pr_{\substack{\alpha \leftarrow I_0(1^n) \\ x \leftarrow S_D(\alpha)}}[A(\alpha, F(\alpha, x)) = x] \leq \mu(n) \tag{1}$$

Or, when sampling an image directly using the range sampler:

$$\Pr_{\substack{\alpha \leftarrow I_0(1^n) \\ r \leftarrow \{0,1\}^n}}[A(\alpha, S_R(\alpha; r)) = f_\alpha^{-1}(S_R(\alpha; r))] = \mu(n) \tag{2}$$

for some negligible function μ.

If f_α is injective for all α, we say that our collection describes an injective trapdoor function family, or 1-1 TDFs. If additionally D_α and R_α coincide for any α, the resulting primitive is a trapdoor permutation.

If the function's domain is just general string, i.e. $D_\alpha = \{0,1\}^n$, we say has a full domain. Otherwise we say the domain is partial. Full and partial range and keyset are defined similarly. We say that a TDF (or TDP) is ideal if it has a full range and a full keyset.

2.4.1 Enhancements

A trivial range-sampler implementation may just sample a domain element x by applying $S_D(\alpha)$, and then evaluate the TDF on it by applying $F(\alpha, x)$. This sampler, while fulfilling the standard one-way hardness condition, is not good enough for some applications. Specifically, for the case of NIZK, we require the ability to obliviously sample a range element in a way that does not expose its pre-image (without using the trapdoor). This trivial range sampler obviously does not qualify for this case.

Goldreich [Gol04] suggested the notion of enhanced TDPs, which can be used for cases where sampling is required to be available in a way that does not expose the pre-image. They then demonstrate how enhanced trapdoor permutations can be used to obtain NIZK proofs (as we describe later in sections 2.5). We revisit this notion, while extending it to the case of 1-1 TDF (where the domain and range are not necessarily equal).

Definition 2.4. (Enhanced 1-1 TDF, Goldreich [Gol04]). Let $\{f_\alpha : D_\alpha \to R_\alpha\}$ be a collection of 1-1 TDFs, and let S_D be the domain sampler associated with it. We say that the collection is enhanced if there exists a range sampler S_R that returns random samples out ofR_α, and such that, for every polynomial-time algorithm A, it holds that:

$$\Pr_{\substack{\alpha \leftarrow I_0(1^n) \\ r \leftarrow \{0,1\}^n}}[A(\alpha, r) = f_\alpha^{-1}(S_R(\alpha; r))] = \mu(n) \tag{3}$$

where μ is some negligible function.

The range sampler of an enhanced 1-1 TDF has the property that its random coins do not reveal a corresponding pre-image, i.e. an adversary which is given an image along with the random coins which created it, still cannot inverse it with all but negligible probability.

Definition 2.5. (Doubly Enhanced 1-1 TDF, Goldreich [Gol08]). Let $\{f_\alpha : D_\alpha \to R_\alpha\}$ be an enhanced collection of 1-1 TDFs, with domain sampler S_D and range sampler S_R. We say that this collection is doubly-enhanced if it provides another polynomial-time algorithm S_{DR} with the following properties:

- **Correlated pre-image sampling:** for any $\alpha, \tau \leftarrow I(1^n)$, $S_{DR}(\alpha; 1^n)$ outputs pairs of (x, r) such that $F(\alpha, x) = S_R(\alpha; r)$
- **Pseudorandomness:** for any polysize distinguisher D there exists a negligible μ such that:

$$\Pr[\alpha, \tau \leftarrow I(1^n), (x, r) \leftarrow S_{DR}(\alpha); D(x, r, \alpha) = 1] - \Pr[\alpha, \tau \leftarrow I(1^n), r \leftarrow \{0,1\}^*, y = S_R(\alpha, r), x = I(\alpha, y); D(x, r, \alpha) = 1] < \mu(n)$$

S_{DR} provides a way to sample pairs of an element x in the function's domain, along with random coins r which explain the sampling of the image $y = f_\alpha(x)$ in the function's range. Note that since the collection is enhanced, r must not reveal any information of x.

2.5 Non-Interactive Zero-Knowledge

2.5.1 Definition

Definition 2.6. (Non-Interactive Zero Knowledge, Blum-Feldman-Micali [BFM88]) A pair of algorithms (P, V) provides a None-Interactive Zero Knowledge (NIZK) proof system for language $L \in NP$ with relation R_L in the *Common Reference String* (CRS) Model if it provides:

- **Completeness:** for every $(x, w) \in R_L$ we have that:

$$\Pr_{P, crs}[\pi \leftarrow P(x, w, crs); V(x, crs, \pi) = 0] < \mu(|x|)$$

  where the probability is taken over the coins of P and the choice of the CRS, and $\mu(n)$ is some negligible function.

- **Soundness:** for every $x \notin L$:

$$\Pr_{crs}[\exists \pi : V(x, crs, \pi) = 1] < \mu(|x|)$$

  where the probability is taken over the choice of the CRS, and $\mu(n)$ is some negligible function.

- **Zero-Knowledge:** there exists a simulator S such that:

$$\{(crs, \pi) : crs \leftarrow U, \pi \leftarrow P(x, w, crs)\}_{(x,w) \in R_L} \approx_c \{S(x)\}_{(x,w) \in R_L}$$

The common reference string is considered the practical one for NIZK proof systems, and is the one widely accepted as the appropriate abstraction. When discussing NIZK proof systems, we sometime omit the specific model being assumed, in which case we mean the CRS model.

2.5.2 NIZK in the Hidden-Bit Model

A fictitious abstraction, which is nevertheless very helpful for the design of NIZK 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 prover generates, along with a proof π, a subset I of indices in the CRS, and pass them both to the verifier. The verifier may only inspect the bits of the CRS that reside in the locations that have been specified by the prover in I, while all other bits of the CRS are hidden to the verifier.

Definition 2.7. (NIZK in the Hidden-Bit Model [FLS90, Gol98]). For a bit-string s and an index set I denote s_I the set of values of s in the indexes given by I: $s_I \coloneqq \{(i, s[i]) : i \in I\}$. A pair of algorithms (P, V) provides a NIZK proof system for language $L \in NP$ with relation R_L in the *Hidden-Bit* (HB) Model if it provides:

- **Completeness:** for every $(x, w) \in R_L$ we have that:

$$\Pr_{P,crs}\left[(\pi, I) \leftarrow P(x, w, crs); V(x, I, crs_I, \pi) = 0\right] < \mu(|x|)$$

  where the probability is taken over the coins of P and the choice of the CRS, and $\mu(n)$ is some negligible function.

- **Soundness:** for every $x \notin L$:

$$\Pr_{crs}\left[\exists \pi, I : V(x, I, crs_I, \pi) = 1\right] < \mu(|x|)$$

  where the probability is taken over the choice of the CRS, and $\mu(n)$ is some negligible function.

- Zero-Knowledge: there exists a simulator S such that:

$$\{(crs_I, \pi) : crs \leftarrow U, \pi, I \leftarrow P(x, w, crs)\}_{(x,w)\in R_L} \approx_c \{S(x)\}_{(x,w)\in R_L}$$

While the hidden-bit model is an unrealistic one, its importance lies in two facts. Firstly, it provides a clean abstraction for NIZK systems, which facilities the design of "clean" proof systems. Efficient NIZK proof systems for NP-hard languages exist unconditionally in the hidden-bit model [FLS90, Gol98]. Secondly, proof systems in the hidden-bit model can be easily transformed into proof systems in the more realistic CRS model, using general hardness assumptions. Feige, Lapidot and Shamir [FLS90] suggests such a transformation.

We remark that in the hidden-bit model, we can obtain both perfect soundness (with a negligible completeness error) and perfect completeness (with a negligible soundness error).

In the rest of this section, we describe their construction and the subtle details of the underlying hardness assumptions.

2.5.3 From Hidden-Bit to CRS: The FLS and BY Protocols

Assuming the existence of one-way permutations, Feige, Lapidot and Shamir [FLS90] constructed a NIZK proof-system in the CRS model for any NP language. They also offer an efficient implementation of the prescribed prover, using trapdoor permutations. We refer to this construction, described next, as the FLS protocol.

The Construction: Let:

- (P_{HB}, V_{HB}) be a hidden-bit proof system for language L (which exists unconditionally)
- $f : \{0,1\}^n \to \{0,1\}^n$ is an injective one-way function, and b a hard-core predicate for it.

Let (P, V) be the following proof system for input x:

- CRS: a sequence of m random items $y_1, ..., y_m$ where each $y_i \in \{0,1\}^n$.
- Prover (P):
  1. Compute $x_i := f^{-1}(y_i)$ and $r_i = b(x_i)$ for $i \in [m]$.
  2. Emulate P_{HB} to obtain $(I, \pi) = P_{HB}(x, r_1 \cdots r_m)$
  3. Output (π, Σ), where $\Sigma := \{(i, x_i) : i \in I\}$.
- Verifier (V): given the proof $(\pi, \Sigma = \{(i, x_i) : i \in I\})$:
  1. check that $x_i = f(y_i)$ for each $i \in I$. Otherwise reject.
  2. compute $r_i = b(x_i)$ for $i \in I$, let $r_I = \{(i, r_I) : i \in I\}$
  3. emulate V_{HB} on (x, r_I, π) , and accept if and only if it accepts.

[FLS90] showed that the resulting construction is a NIZK proof system for L in the CRS model:

Theorem 2.1. *([FLS90]) Assuming the existence of **one-way permutations**, there exists a NIZK proof system in the CRS model with an **inefficient prover** for any NP language.*

Implementing an Efficient Prover using Ideal Trapdoor Permutations: In order for the prover P in the FLS system to be efficient, it must be able to efficiently invert f. On the other hand, the verifier V must not be able to efficiently invert f in order to preserve the zero-knowledge property of the system. The obvious solution is to use a family of trapdoor permutations, and let the prover choose the permutation. The prover invokes the generation algorithm of the TDP to receive an index α and a trapdoor τ. It then uses τ to invert the y_i's. The verifier receives α from the prover and uses it to evaluate f and b. As we can no longer assume that the permutation key chosen by the prover is truly random, we consider the probability of success of the prover for any specific choice of permutation, and then union bound over all possible permutations. This means that in order to guarantee soundness, the initial soundness error must be smaller than inverse the number of permutations. We guarantee that by enhancing the soundness error via repetition. We omit the rest of the details.

Theorem 2.2. *([FLS90]) Assuming the existence of an* ***ideal trapdoor permutation*** *family, there exists a NIZK proof system in the CRS model (with an efficient prover) for any NP language.*

As shown by [FLS90], the FLS protocol provides a NIZK proof system assuming that the underlying TDP is ideal. However, existing instantiations of TDPs are *not* ideal, and in fact are far from it. Most reasonable constructions of TDPs have both partial keysets and partial domains. This leads to two gaps which arise when using general TDPs, in place of ideal ones.

Ideal Domains + General Keys: The Bellare-Yung Protocol: The first hurdle, discovered by Bellare and Yung [BY96], involves the use of general trapdoor keys (rather than ideal ones). The problem is that the soundness of the FLS protocol relies on the feasibility of recognizing permutations in the collection. If the permutation is ideal then every key describes a permutation, and therefore detecting a permutation is trivial. However, existing instantiations of TDPs require sampling keys from a certain form using a specific protocol. This brings us to the problem of *certifying permutations*, which aims to answer the question of how to certify that a given key indeed describes a valid permutation. Bellare and Yung [BY96] suggested a certification procedure for permutations, assuming nothing of the keyset, but requiring that the range remains full. We refer to this procedure as the Bellare-Yung protocol. The following is an overview of the construction and proof given in section 4 of [BY96].

Definition 2.8. (Almost-Permutations). Let $C(f)$, the *Collision Set* of f, be the set of all n-bit strings which have more than one pre-image:

$$C(f) := \{y \in \{0,1\}^n : \exists x_1 \neq x_2 \in \{0,1\}^n . f(x_1) = f(x_2) = y\} \tag{4}$$

We say that f is an ε-permutation (for $0 \leq \varepsilon \leq 1$) if its collision set is at most an ε-fraction of the entire domain, i.e. $|C(f)| \leq \varepsilon 2^n$. If f is a 0-permutation then it is by definition a permutation. We say that f is an *almost* permutation if it is an $\varepsilon(n)$-permutation for some negligible $\varepsilon(n)$.

For general functions (with different domain and range), we define *almost injectivity* in a similar way: if $Range(f) \subseteq \{0,1\}^m$, then the collision set is defined as the set of all m-bit strings which have more than one pre-image. Next, we say that f is ε-injective if $|C(f)| \leq \varepsilon \cdot |Domain(f)|$, and that it is almost injective if it is $\varepsilon(n)$-injective for some negligible $\varepsilon(n)$.

The main observation is that f is an ε-permutation if and only if at most ε-fraction of $\{0,1\}^n$ has no pre-image. Given a trapdoor permutation family described by (I, S_D, F, B) (where S_D just samples a string from $\{0,1\}^n$), Bellare-Yung described the following protocol for certifying that some (α, τ) describe an almost-permutation. The prover and verifier treat the CRS as a sequence of some l range items $y_1, ..., y_l$ (where $y_i \in \{0,1\}^n$). The prover provides the verifier with a list of pre-images $x_1, ..., x_l$ such that $x_i = B(\tau, y_i)$ (where B is the backwards-evaluation or inversion algorithm of the TDP family). The verifier accepts if $y_i = F(\alpha, x_i)$ for all i (where F is the forward evaluator). By asking the prover to invert sufficiently many random domain element, the verifier is convinced that the collision set is small enough, meaning that the given index describes an almost-permutation. Finally, as it turns out, being an almost-permutation is sufficient for the purpose of the FLS protocol.

Theorem 2.3. *([BY96]) Assuming the existence of a **full-domain trapdoor permutation** family (whose keys may be hard to recognize), there exists a NIZK proof system in the CRS model for any NP language (with an efficient prover).*

General Domains: Doubly Enhanced TDPs: The second gap concerns the case of partial domains, where the function's domain is comprised of elements of specific structure (and not just $\{0,1\}^n$). The FLS protocol treats the CRS as a sequence of range elements. In the case of the general abstraction of trapdoor permutations, an additional domain sampling algorithm is required. This problem is solved by requiring the use of doubly enhanced trapdoor permutations. Given the permutation index α, both the prover and the verifier use the enhanced sampling algorithm $S_R(\alpha)$ to sample elements from the permutation's range. They treat the CRS as a sequence $r_1, ..., r_l$, where each $r_l \in \{0,1\}^n$ is handled as random coins for the range sampler. They create a list of range items $y_i = S_R(\alpha; r_i)$ and use them for the rest of FLS protocol. Using an enhanced range sampler solves the completeness issue of NIZK in the CRS model for permutations with general domains. However, the resulting protocol may no longer be zero-knowledge, as the verifier now obtains a list of random pairs (x_i, r_i) such that $f_\alpha(x_i) = S_\alpha(r_i)$, but it is not clear that it could have generated such pairs itself. The second enhancement solved just that, and allows the verifier to obtain such pairs on its own.

Theorem 2.4. *([GR13]) Assuming the existence of a general* ***doubly-enhanced trapdoor permutation*** *family with* ***efficiently recognizable keys****, there exists a NIZK proof system in the CRS model for any NP language (with an efficient prover).*

Moreover, in order to certify general keys, [Gol11, GR13] suggested combining between doubly enhanced permutations and the Bellare-Yung protocol, by using the doubly-enhanced domain sampler to sample images by the Bellare-Yung prover and verifier. We reexamine this suggestion in section 3.

Basing FLS on Injective Trapdoor Functions: Before moving on, we mention that while the FLS protocol is originally described using (trapdoor) permutations, it may just as well be described and implemented using general injective trapdoor functions. In this case, since the CRS is used to generate range elements, there is no useful notion of "ideal" injective trapdoor functions; if f maps n-bit strings into m-bit strings, where $m > n$, then there must exists some m-bit strings which do not have a pre-image under f. However, using a doubly-enhanced general injective trapdoor function, the FLS protocol and the generalization into general TDPs will work without any changes, under assuming the keys are efficiently recognizable. In section 5 we will show an example for such a 1-1 TDF and it's application to NIZK proof systems.

3 On the Unsoundness of FLS with General Doubly Enhanced TDPs

We begin with a careful reexamination of the FLS protocol, in light of the work of [Gol11, GR13]. We discuss a crucial problem yet to be detected when applying the Bellare-Yung protocol on general TDPs, which have both partial domains and partial keysets. Specifically, we identify that the soundness of the FLS protocol may be compromised when using such trapdoor functions.

3.1 The Counter Example

We once again sketch the full details of the Bellare-Yung protocol, this time allowing both partial range and partial keyset for our TDPs, as suggested by [GR13]. To simplify matters, we limit this part of the discussion to the case of trapdoor permutations (rather than any injective trapdoor functions), which is consistent with the efforts done by previous work.

Recall that we are provided with a doubly-enhanced TDP family, described using the algorithms $I(1^n) \to (\alpha, \tau), F(\alpha, x) \to y, B(\tau, y) \to x, S(\alpha; r) \to x$. We treat the CRS as a sequence of random coins for the sampler S, and apply S both on the prover and on the verifier side to obtain range elements.

- Input: $(\alpha, \tau) \leftarrow I(1^n)$
- CRS: a sequence of l random strings $r_1, ..., r_l$, each acts as random coins for S
- Prover: is given (α, τ) and does the following:
  1. Calculate $y_i := S(\alpha; r_i)$ for each $1 \leq i \leq l$.
  2. Calculate $x_i := B(\tau, y_i)$ for each $1 \leq i \leq l$.
  3. Output $\{(i, x_i) : 1 \leq i \leq l\}$
- Verifier: is given α and $\{(i, x_i) : 1 \leq i \leq l\}$, and does the following
  1. Calculate $y_i := S(\alpha; r_i)$ for each $1 \leq i \leq l$.
  2. Validate that $y_i = F(\alpha, x_i)$ for each $1 \leq i \leq l$. If any of the validations fail, reject the proof. Otherwise, accept it.

Looking into the details of the protocol, we detect a potential problem. We demonstrate it by instantiating the FLS+BY protocols using a specific family of doubly-enhanced trapdoor permutations, which was proposed by [BPW15]:

Let PRF_k be a pseudorandom function family, and iO an indistinguishability obfuscator. Let C_k be the circuit that, on input (i, t), if $t = PRF_k(i)$ outputs $(i + 1, PRF_k(i + 1))$ (where $i + 1$ is computed modulo some T) and otherwise outputs $\perp$. Denote by $\tilde{C} := iO(C_k)$ the obfuscation of C_k. The BPW construction gives $\tilde{C}$ as the public permutation index, and keeps k as the trapdoor. To evaluate the permutation on a domain element $(i, PRF_k(i))$, just apply $\tilde{C}$. To invert $(i + 1, PRF_k(i + 1))$ given k, return $(i, PRF_k(i))$. The range sampler is given as an obfuscation of a circuit which samples out of a (sparse) subset of the function's range. One-wayness holds due to a hybrid puncturing argument: the obfuscation of the cycle $(i, PRF_k(i)) \rightarrow (i+1, PRF_k(i+1))$ (where $i+1$ is computed module T) is indistinguishable from that of the same cycle when punctured on a single spot i^*, by replacing the edge $(i^*, PRF_k(i^*)) \rightarrow (i^* + 1, PRF_k(i^* + 1))$ with a self loop from $(i^*, PRF_k(i^*))$ to itself. By repeating the self-loops technique we obtain a punctured obfuscated cycle where arriving from $(i, PRF_k(i))$ to its predecessor $(i - 1, PRF_k(i - 1))$ cannot be done efficiently without knowing k itself.[2]

Suppose that the [BPW15] construction is used to instantiate the FLS+BY protocols, and consider the following malicious prover: Let C'_k be a circuit which, given input (i, t), does the following: if $t = PRF_k(i)$ or $t = PRF_k(i - 1)$, output $(i + 1, PRF_k(i + 1))$.

[2] In order to add an enhanced domain sampler, the BPW construction returns elements of the form $(PRG(r), PRF_k(PRG(r)))$, where PRG is a pseudorandom generator which lengthens the input by a significant factor. The domain sampler is just an obfuscation of a circuit which outputs the above pair on some random r. By augmenting the sampler even more, they were able to doubly-enhance their TDP, at the cost of creating a very sparse part of the domain which is sampleable. We leave the rest of the details to the reader.

Otherwise, output $\perp$. Denote $\tilde{C}' \coloneqq iO(C'_k)$. We give out $\tilde{C}'$ as the public key and keep k as the trapdoor. We keep the domain sampler as it is, that is, it returns only items of the form $(i, PRF_k(i))$.

Denote $D_k = \{(i, PRF_k(i) : i \in [1...T])\}$ and $\tilde{D}_k = \{(i, PRF_k(i)) : i \in [1...T]\} \cup (i, PRF_k(i-1)) : i \in [1...T]\}$. It is easy to see that C'_k is a permutation when restricted to the the domain D_k, but it many-to-one when evaluated on the domain $\tilde{D}_k$: each item $(i+1, PRF_k(i+1)) \in D_k$ has 2 pre-images: $(i, PRF_k(i))$ and $(i, PRF_k(i-1))$. Note that the one-wayness of the trapdoor function is maintained even when extended to the domain $\tilde{D}_k$: For each image $(i+1, PRF_k(i+1))$ we now have two pre-images, one is $(i, PRF_k(i))$ which is hard to invert to due to the same puncturing argument as in the original BPW paper, and the second is $(i, PRF_k(i-1))$ which has no pre-image of its own, and therefore no path on the cycle can lead to it (keeping the same one-wayness argument intact).

Finally, our cheating prover can wrongly "certify" the function as a permutation. The domain sampler will always give an image in D_k as it was not altered. During the Bellare-Yung certification protocol, the prover can invert $y = (i+1, PRF_k(i+1)) \in D_k$ to, say, $(i, PRF_k(i))$, which will pass the validation. However, during the FLS protocol, the verifier can choose to invert any $y \in D_k$ to one of its two distinct pre-images, one from D_k and another from $\tilde{D}_k \setminus D_k$.

3.2 Discussion

We attribute the loss in soundness when applying the FLS+BY combination on the [BPW15] construction to a few major issues.

First, we observe that both the sampling and forward evaluation algorithms are required to operate even on illegitimate keys. However, the basic definition of trapdoor permutations (c.f. [Gol98]) does not address this case at all. Ignoring this case may make sense in settings where the party generating the index is trusted, but this is not so in the case of NIZK proof systems. We therefore generalize the basic definition of trapdoor permutations so that the forward evaluation and domain sampling definitions generalize to any α, rather than just those which were generated by running the index-generation algorithm. That is, for every α, D_α is some domain over which $F(\alpha, \cdot)$ is well defined, and $S(\alpha; r)$ returns elements from that domain.

We next claim that in order for the soundness of the complete FLS+BY protocol to be preserved, two additional requirements are needed: First, membership in D_α should be efficiently recognizable given α. That is, there should exist an efficient algorithm which, given α and some string x, decides if x represents an element in D_α or not. Second, the domain sampler S should be guaranteed to sample (almost) uniformly out of D_α. We stress that both these requirements should hold with respect to *any* index α, in particular indices that were not generated truthfully. Furthermore, they are made *on top of* the existing requirements from doubly-enhanced trapdoor permutations.

We call doubly enhanced trapdoor permutations that have these properties *public domain*. We formalize this notion in Definition 4.2 and prove that it indeed suffices for regaining the soundness of the FLS+BY combination in theorem 4.2 (see section 4.3).

In the rest of this section, we show that these two requirements are indeed necessary, by demonstrating that if either of the two do not hold then the resulting proof system is not sound.

First, consider the case where S does not sample almost uniformly from D_α. The soundness of Bellare-Yung depends on the observation that if the function is not an almost-permutation, then by sampling enough random images from the function's domain, there must be a sample with cannot be inverted (with all but negligible probability). However, if the sampler does not guarantee uniformity this claim no longer holds, as the prover may give out a sampler which samples only out of that portion of the range which is invertible.

Secondly, assume S indeed samples uniformly from the domain, and consider the case where D_α is not efficiently recognizable. As it turns out, both the Bellare-Yung protocol and the original FLS protocol require the verifier to determine whether pre-images provided by the prover are indeed in D_α. Otherwise, a malicious prover could certify the permutation under a specific domain, but later provide pre-images taken out of an entirely different domain, thus enabling it to invert some images to two or more pre-images of its choice.

Indeed, the attack described in section 3.1 takes advantage of the loophole resulting from the fact that the domain of the [BPW15] is neither efficiently recognizable nor efficiently sampleable. The exact reason for the failure depends on how the domain of [BPW15] is defined with respect to illegitimate indices. Say for $\alpha = \tilde{C}$, we give out D_α which includes only pairs (i, x) such that $x = PRF_k(i)$ (for the specific k used to construct $\tilde{C}$). In that case, S indeed samples uniformly from D_α. However since D_α is not efficiently recognizable, the prover cannot check that the pre-image it was given is from D_α. In particular it cannot tell if it is from $D_k = D_\alpha$ or from $\tilde{D}_k$. On the other hand, if $D_\alpha = \{0,1\}^*$, then D_α may be trivially recognizable for any index, but S does not guarantee a uniform sample from D_α. Indeed, S may sample only from that subset of D_α which is invertible, thus harming the soundness.

4 Certifying Injectivity of Trapdoor Functions

We go back to the original problem of certifying permutations in a way that is sufficient for the FLS protocol, while addressing the more general problem of certifying injectivity of trapdoor functions (which may or may not be permutations). We note that although this problem is motivated by the need to fill in the gaps in the FLS protocol, a solution for it might be interesting on its own.

In section 4.1 we define the notion of *Certifiable Injectivity* as a general abstraction of certifiability for doubly-enhanced injective trapdoor functions. In section 4.2 we prove that this notion indeed suffices for regaining the soundness of the FLS protocol. In section 4.3

we show how certifiable injectivity can be realized by any trapdoor permutations whose domain provides certain additional properties, by using the Bellare-Yung certification protocol. In section 4.4 we suggest the notion of *Perfectly Certifiable Injectivity* as a specific variant of certifiable injectivity, where there is no longer need for a certification protocol and the resulting soundness is optimal.

4.1 Certifiable Injectivity - Definition

We define a general notion of certifiability for injective trapdoor functions, which requires the existence of a general prover and verifier protocol for the function family. The purpose of this protocol is to guarantee that if the verifier accepts the proof given by the prover on a certain index α, then with all but negligible probability (over the coins of the range sampler), the range sampler cannot sample images which can be inverted to any two pre-images which are both acceptable by the verifier. We formulate the notion of "acceptable" pre-images by requiring an additional efficient pointwise certification algorithm $ICert$ where, given an index α and an image y, with all but negligible probability over the choice of y, there exists only one pre-image x such that $ICert$ accepts on (α, x).

Definition 4.1. (Certifiable Injective Trapdoor Functions (CITDFs)). Let $F = \{f_\alpha : D_\alpha \to R_\alpha\}$ be a collection of doubly enhanced injective trapdoor functions, given using algorithms I, F, B, S_D, S_R. We say that F is *certifiably injective* (in the common reference string model) if there exists an efficient algorithm $ICert$ and a pair of efficient probabilistic algorithms (P, V), which provides the following properties:

- **Completeness:** for any $\alpha, \tau \leftarrow I(1^n)$ we have:
  1. $\Pr_{P,V,crs}[\pi \leftarrow P(\alpha, \tau, crs); V(\alpha, crs, \pi) = 1] = 1$, where the probability is taken over the coins of P and V and the choice of the CRS, and
  2. For any $x \in D_\alpha$, $ICert(\alpha, x) = 1$.
- **Soundness:** there exists a negligible function μ such that for any α the following holds:

$$\Pr_{crs,V,r}[\exists \pi, x_1, x_2 : V(\alpha, crs, \pi) = 1, F(\alpha, x_1) = F(\alpha, x_2) = S_R(\alpha, r), \\ ICert(\alpha, x_1) = ICert(\alpha, x_2) = 1] \leq \mu(n)$$

  where the probability is taken over the coins of V the choice of the CRS, and the random coins given to the range sampler. Note that this must hold for any α, including those that I cannot output, and that π can be chosen adaptively given the common reference string.

- **Hardness (even) given the Proof:** for any polynomial-time algorithm A there exists a negligible function μ, such that the following holds:

$$\Pr_{P,crs}[\alpha,\tau \leftarrow I(1^n); \pi \leftarrow P(\alpha,\tau,crs); x \leftarrow S_D(\alpha); A(\alpha, F(\alpha,x), crs, \pi) = x] \leq \mu(n)$$

  where the probability is taken over the coins of P and the choice of the CRS.

Certifiable injectivity gives a general way to certify that a given key describes an injective function, even when using general, partial-domain/range functions. The proof generated by P and verified by F is used to certify that the given key α is indeed injective, in the sense that if V accepts it then no two acceptable pre-images can map to the same image (with all but negligible probability).

4.2 Certifiable Injectivity Suffices for the Soundness of FLS

Our key theorem, stated formally next, shows how combining certifiable injectivity with the FLS protocol and doubly-enhanced permutations, we overcome the existing problems and obtain NIZK for NP from general permutations. The intuition is simple: we take a doubly-enhanced, certifiably injective collection of trapdoor permutations, and treat the CRS as two separate strings. The first string is used to certify the injectivity of the trapdoor function, using the CI-prover and verifier, while the second is used for the FLS protocol. Moreover, we add a certification step to the FLS protocol itself, by having the verifier run $ICert$ on any pre-image provided to it by the prover. The soundness guarantee of CI notion ensures that a malicious prover must choose a trapdoor index which describes a permutation (or at least an almost-permutation) over the domain of elements accepted by $ICert$, or otherwise the CI verifier would reject the first part of the proof. The hardness guarantee ensures that the FLS proof remains zero-knowledge, even in the presence of the CI proof.

Theorem 4.1. *(DECITDFs $\rightarrow$ NIZK) Let $F = \{f_\alpha : D_\alpha \rightarrow R_\alpha\}$ be a collection of doubly-enhanced, certifiably injective trapdoor functions, and let L be an NP language. Then there exists a NIZK proof system for L in the CRS model.*

Proof. We treat the common reference string as two separate substrings c_1, c_2. c_1 will be used by the CI-prover and CI-verifier (P_{CI}, V_{CI}) for F. c_2 will be used by the prover-verifier pair from the FLS protocol, denoted (P_{FLS}, V_{FLS}). For $\alpha, \tau \in I(1^n)$, Denote by $P_{FLS}[\alpha,\tau]$ and $V_{FLS}[\alpha]$ the evaluation of P_{FLS}, V_{FLS} with α, τ as the trapdoor and index. Let (P, V) be the following protocol:

- The prover P is given an instance-witness pair $(x, w) \in R_L$. It selects $(a, \tau) \leftarrow I(1^n)$. It then generates a proof $\pi_1 \leftarrow P_{CI}(\alpha, \tau, c_1)$ for the injectivity of f_α. Next, it generates a proof π_2 for x, by taking $\pi_2 \leftarrow P_{FLS}[\alpha,\tau](x, w, c_2)$. P outputs $\pi = (\pi_1, \pi_2, \alpha)$ as the proof.

- The verifier V receives $\pi = (\pi_1, \pi_2, \alpha)$. It first validates π_1 by running $V_{CI}(\alpha, c_1, \pi_1)$. Then, it checks π_2 by running $V_{FLS}[\alpha](x, \pi_2, c_2)$, with the addition the for every pre-image x_i submitted by the prover, the verifier runs $ICert(\alpha, x_i)$. V accepts only if both the validators accepted, and $ICert(\alpha, x_i)$ returned one on all the pre-images.

We will show that (P, V) provide a NIZK proof system for L in the CRS model.

Completeness follows immediately from the completeness of the CI notion and of the FLS protocol.

Zero Knowledge follows from the zero-knowledge property of the FLS protocol, along with the hardness given the proof of the CI notion. From the zero-knowledge of the FLS protocol, we have that there exists a simulator S_{FLS} such that

$$\{(c_2, \pi_2) : c_2 \leftarrow U, \pi_2 \leftarrow P_{FLS}(x, w, c_2)\}_{(x,w)\in R_L} \approx \{S_{FLS}(x)\}_{(x,w)\in R_L} \tag{5}$$

Adding a simulator for the CI part of the proof is straightforward, as the proof π_1 does not depend on x at all. So, let S be the following algorithm: given an input x, first choose $\alpha, \tau \leftarrow I(1^n)$ and generate a proof $\pi_1 = P_{CI}(\alpha, \tau, c_1)$ (with some randomly selected c_1). Next, get π_2 by running $S_{FLS}[\alpha, \tau](x)$. S outputs $((c_1, c_2), (\pi_1, \pi_2, \alpha))$. Since c_1, π_1 are independent of x, due to equation 5, this is indistinguishable from a proof generated by P.

The **Soundness** of the overall protocol is straightforward. By following the arguments presented by [BY96], in order for the FLS protocol to maintain soundness, it suffices that for, with all but negligible probability over the random coins r used by the range sampler, the image $y = S_R(\alpha; r)$ has only one pre-image x which V accepts, i.e. such that $ICert(\alpha, x) = 1$. Assuming V_{CI} accepted π_1, we use a similar union-bound argument as that of [BY96], to bound the additional error incurred in the case of a common reference string $\sigma = \sigma_1, ..., \sigma_l$ such that $y_i = S_R(\alpha, \sigma_i)$ has two pre-images x_i^1, x_i^2 which both pass $ICert(\alpha, \cdot)$, by a negligible factor. This is since if V_{CI} accepted the proof, then the probability over the coins of S_R that there exists such a pair of certified pre-image is negligible. □

4.3 Certifiable Injectivity for Public-Domain TDPs using Bellare-Yung

Building on the discussion in section 3.2, we formalize the notion of *public-domain* trapdoor permutations. We then show that, when applied to public-domain permutation, the BY certification mechanism suffices for guaranteeing Certifiably Injectivity (and, thus, also soundness of the FLS paradigm.)

Definition 4.2. (Public-Domain Trapdoor Permutations.) Let $f_\alpha : \{D_\alpha \rightarrow D_\alpha\}$ be a trapdoor permutation family, given by (I, S, F, B). We say that it is public-domain if the following two additional properties hold:

- **The domain is efficiently recognizable:** that is, there exists an efficient algorithm Rec which, for any index α and any string $x \in \{0,1\}^*$, accepts on (α, x) if $x \in D_\alpha$. In other words, D_α is defined as the set of all strings x such that $Rec(\alpha, x)$ accepts.

- **The domain is efficiently sampleable:** that is, for any index α, $S(\alpha)$ samples almost uniformly from D_α.

We stress that both properties should hold with respect to any α, including ones that were not generated by running I.

We show that indeed, for the case of public-domain doubly-enhanced trapdoor permutations, Bellare-Yung can be used to obtain certifiable injectiveness.

Theorem 4.2. *Any doubly-enhanced public-domain trapdoor permutation family is certifiably injective.*

Proof. Let F be a doubly enhanced public-domain trapdoor permutation. Let (P, V) the prover and verifier from the enhanced Bellare-Yung protocol for F, that is, the version of Bellare-Yung that uses the enhanced range sampler to generate images from the random coins given in the common reference string, as described in section 3.1. Let Rec be an efficient domain recognizer for D_α, for any index α (which exists since the permutation family is public-domain). We claim that F is certifiably injective, with $ICert(\alpha, x) = Rec(\alpha, x)$ and (P, V) giving the CI prover and verifier.

As shown by [BY96] for the case of full-domain trapdoor permutations, (P, V) provide soundness, certifiable injectivity and zero-knowledge (which implies the hardness requirement of CI). Moreover, if V accepts the proof then the size of the collision set is negligible, which implies that the probability that a random image has two pre-images is indeed negligible. In the case of general doubly-enhanced trapdoor permutations, the only property at risk is soundness. We prove that it is indeed maintained for public-domain trapdoor permutations.

Let α be an index such that $F(\alpha, \cdot)$ is not almost-injective over D_α. Let C_α be the collision set of α out of D_α: $C(\alpha) = \{y \in D_\alpha : \exists x_1 \neq x_2 \in D_\alpha \text{ s.t. } F(\alpha, x_1) = F(\alpha, x_2)$. Then $|C_\alpha| \geq \varepsilon(n) \cdot |R_\alpha|$ for some non-negligible $\varepsilon(n)$, hence there exist at least $\varepsilon(n) \cdot |D_\alpha|$ range items with no pre-image in D_α. Moreover, $ICert(\alpha, x) = Rec(\alpha, x)$ is efficient and recognizes D_α, V will not accept any pre-image outside of D_α, hence there exists at least $\varepsilon(n) \cdot |D_\alpha|$ range items with no pre-image that V accepts (in D_α or outside of it). Denote that uninvertible portion of D_α as $U(\alpha)$.

The prover and the verifier apply $S(\alpha; r_i)$ on a series of random coins $r_1, ..., r_l$ taken from the common reference string. Using a similar argument to that presented in [BY96], for a large enough l (polynomial in n), with all but negligible probability (over the CRS), there must exist at least one $y_i = S(\alpha; r_i) \in U(\alpha)$. This holds since the $S(\alpha; \cdot)$ is guaranteed to generate uniform samples out of D_α, meaning given enough samples, one has to fall into the non-negligible part $U(\alpha)$. Hence, for a large enough l, V rejects π with all but negligible probability.

□

We note that some existing candidate constructions, such as ones on the line of [BPW15], are not public-domain, as they inherently need the sampling algorithm to hold secrets. Indeed, as demonstrated in section 3, Bellare-Yung does not suffice to guarantee soundness when instantiating FLS with such a candidate. On the other hand, the RSA TDPs are public-domain: the domain Z_N^* is indeed efficiently recognizable for any public index N, and an efficient certifiably uniform domain sampler can be described for any public key N of RSA, by mapping strings in $\{0,1\}^n$ to Z_N^* in a way that obtains (almost) uniform samples in Z_N^*[3]. For those constructions the FLS+BY combination is indeed sound.

4.4 Perfectly Certifiable Injectivity

While certifiable injectivity seems to capture the minimal requirement for a trapdoor permutation that suffices for FLS, the requirement of a prover and verifier algorithms are somewhat cumbersome when viewed purely in the context of trapdoor permutations. We thus suggest a strengthened notion of Perfectly Certifiable Injectivity, which is a variant of certifiable injectivity in which the pointwise certification algorithm $ICert$ provides a stronger guarantee, eliminating the need for an additional prover-verifier protocol.

Definition 4.3. (Perfectly Certifiable Injective 1-1 TDFs). A doubly-enhanced injective TDF family is *perfectly certifiable injective* if, in addition to the standard set of algorithms I, S_D, S_R, F, B, it defines a certification algorithm $ICert$.

$ICert$ is given a permutation index α and a pre-image x, and accepts or rejects, providing the following two guarantees:

- **Completeness:** If $\alpha \leftarrow I_0(1^n)$ and $x \leftarrow S_D(\alpha)$ then $ICert(\alpha, x) = 1$.
- **Perfect Soundness:** For any index α, there do not exist any $x_1 \neq x_2 \in \{0,1\}^*$ such that $F(\alpha, x_1) = F(\alpha, x_2)$ and $ICert(\alpha, x_1) = ICert(\alpha, x_2) = 1$.

  Note that α needs not be generated honestly by I.

The standard hardness condition is required as usual (and must apply even in the presence of $ICert$).

Perfect CI is a special case of general CI, where the soundness of $ICert$ is absolute; for any α, x_1, if $ICert(\alpha, x_1) = 1$ then it is guaranteed that there exists no second pre-image x_2 which maps to $F(\alpha, x_1)$ and accepted by $ICert(\alpha, \cdot)$. It turns out that in the specific case where the trapdoor function family in use is perfectly certifiable injective with, the index certification protocol can be completely avoided. Indeed, the soundness requirement of definition 4.1 is trivially fulfilled, as:

$$\Pr_r[\exists x_1, x_2 : F(\alpha, x_1) = F(\alpha, x_2) = S_R(\alpha, r), ICert(\alpha, x_1) = ICert(\alpha, x_2) = 1] = 0$$

[3] Full details can be found in [BY96] and [GR13], appendix B

An important property of this technique is that the soundness it provides is perfect, in that it is identical to the soundness obtained by using ideal trapdoor permutations. No additional error is incurred, since for every image there exists a single acceptable pre-image (unconditionally).

5 Doubly Enhanced Perfectly Certifiable Injective Trapdoor Functions from iO+

We construct doubly-enhanced injective trapdoor functions using iO + pseudorandom generators (which can be constructed from one way functions). Additionally, assuming the pseudorandom generator is injective, we show that the injectivity of our construction is perfectly certifiable. Using the additional certification procedure, our construction suffices for general NIZK proofs for NP-languages. This construction is motivated by the [SW13] CPA-secure public key encryption system.

For simplicity, in sections 5.1-5.4, we assume that the PRGs and PPRFs being used by our construction are *full domain*. This assumption makes sense in the context of general pseudorandom generators and puncturable pseudorandom functions, where natural full-domain candidates exist (c.f. [GGM86]). However this is not the case for *injective* PRGs, which are required for our certifiable injectivity proof. In section 5.5 we show how this assumption can be relaxed, by allowing injective PRGs with a domain which is efficiently sampleable and recognizable. We additionally demonstrate how these requirements can be realized by existing candidates.

5.1 Construction

Let:

- $g : \{0,1\}^n \to \{0,1\}^{2n}$ be a n-to-$2n$-bits pseudorandom generator
- $d : \{0,1\}^{n/2} \to \{0,1\}^n$ be a $n/2$-to-n pseudorandom generator
- $\{f_k : \{0,1\}^{2n} \to \{0,1\}^n\}_{k \in K}$ be a puncturable pseudorandom function family
- $\{h_w : \{0,1\}^n \to \{0,1\}^n\}_{w \in W}$ be a length-preserving puncturable PRF family
- iO be an indistinguishability obfuscation scheme, where $iO, EvalO$ are its obfuscation and evaluation algorithms.

Let $T_k, S_{k,w}$ and Q_w be the following circuits:

$T_k(x)$: // (Forward evaluator)
    constants:

```
        puncturable PRF key k
    t = g(x)
    s = f_k(t)
    return (x ⊕ s, t)

S_{k,w}(r): // (Range Sampler)
    constants:
        puncturable PRF key k for f
        puncturable PRF key w for h
    x = h_w(r)
    return T_k(x)

Q_w(ρ): // (Correlated Image Sampler)
    constants:
        puncturable PRF keys w for h
    r = d(ρ)
    x = h_w(r)
    return (x, r)
```

We define our 1-1 TDF in the following way:

- $I(1^n)$: Choose $k \leftarrow K$ as a PRF key for f, and $w \leftarrow W$ as a PRF key for h. Denote $\tilde{T} := iO(T_k)$, $\tilde{S} := iO(S_{k,w})$, $\tilde{Q} := iO(Q_w)$. Output $\alpha := (\tilde{T}, \tilde{S}, \tilde{Q})$ as the public TDP index, and $\tau := k$ as the trapdoor.
- $F(\alpha = (\tilde{T}, \tilde{S}, \tilde{Q}), x \in \{0,1\}^n)$: output $EvalO(\tilde{T}, x)$.
- $B(\tau = k, y = (c \in \{0,1\}^n, t \in \{0,1\}^{2n}))$: output $c \oplus f_k(t)$.
- $S_D(\alpha = (\tilde{T}, \tilde{S}, \tilde{Q}), r \in \{0,1\}^n)$: output r.
- $S_R(\alpha = (\tilde{T}, \tilde{S}, \tilde{Q}), r \in \{0,1\}^n)$: output $EvalO(\tilde{S}, r)$.

Motivation: $\tilde{T} = iO(T_k)$ is used as the forward evaluation algorithm, with the secret key k used to invert it. $\tilde{S} = iO(S_{k,w})$ is used as a range sampler providing the first enhancement, with h_w being used to re-randomize the random coins provided to in to create a secret pre-image. $\tilde{Q} = iO(Q_w)$ will be used to provide the second enhancement, using yet another round of re-randomization on the coins provided to it.

Theorem 5.1. *(Completeness / Injectivity) The above construction describes an injective function family over $\{0,1\}^n \to \{0,1\}^{3n}$. Furthermore, B calculates the inversion of F over the above domain and range.*

Proof. Suppose $(c_1, t_1) = (c_2, t_2)$. Then $t_1 = t_2$, hence for $s_1 = f_k(t_1)$ and $s_2 = f_k(t_2)$ we have $s_1 = s_2$. So $c_1 = s + x_1$, $c_2 = s + x_2$, and $c_1 = c_2$, hence $x_1 = x_2$.

For $x \in \{0,1\}^n$, $F(\alpha, x)$ outputs $y = (x \oplus s, t)$. $B(\tau, y)$ outputs $x \oplus s \oplus f_k(t) = x$. $\square$

5.2 Hardness

Theorem 5.2. *(Hardness) The function family described by the above construction is one-way.*

Proof. We define the TDF hardness using a game between a game-master GM and an adversary A:

1. GM chooses random keys k, w and a random pre-image $x^* \leftarrow \{0,1\}^n$. It takes $t^* = g(x^*), s^* = f_k(t^*), z^* = x * \oplus s^*, y^* = (z^*, t^*), \tilde{T} = iO(T_k), \tilde{S} = iO(S_{k,w}), \tilde{Q} = iO(Q_w)$.
2. A receives $\alpha = (\tilde{T}, \tilde{S}, \tilde{Q})$ and y^*, and outputs x'.

We define the advantage of A as $adv(A) = \Pr[\tilde{T}(x') = y^*]$ (where the probability is taken over the coins of I and the selection of x^*). We require that for any PPT adversary A, $adv(A) \leq \mu(n)$ for a negligible function μ. It is easy to see that this definition is equivalent to the standard hardness definition given in section 2.4.

We now continue the proof using a hybrid argument. We define a series of hybrids, each describing a game between the game master GM and the adversary A. We show that in each pair of consecutive hybrids, denoted H_i and H_{i+1}, the advantage obtained by the adversary must be computationally close, denoted $H_i \approx H_{i+1}$, or otherwise some underlying hardness assumption will break. In the last hybrid we will show that no adversary can win with non-negligible advantage (unconditionally), thus proving the hardness of the TDF obtained by the construction. Note that the key w remains unpunctured and $\tilde{Q}$ remains unchanged throughout the hybrids.

- H_0: the game is played between A and GM as described above.
- H_1: same as in H_0, only GM replaces $\tilde{T}$ and $\tilde{S}$ with obfuscation of two different programs. That is, GM chooses random keys k, w, a random pre-image $x^* \leftarrow \{0,1\}^*$ and $t^* = g(x^*), s^* = f_k(t^*), z^* = x^* \oplus s^*, y^* = (z^*, t^*)$ as before. Let $k^* = k(\{t^*\})$ be the punctured PRF key at point t^*, and let $\tilde{T} = iO(T_1\{k^*, t^*, z^*\})$, where $T_1\{k^*, t^*, z^*\}$ is the following program:

```
T_1{k*, t*, z*}(x):
    constants:
        punctured PRF key k*
        points t* ∈ {0, 1}^{2n}, z* ∈ {0, 1}^n
    t = g(x)
    if t = t* then z = z*
    else
        s = f_{k*}(t)
        z = x ⊕ s
    return (z, t)
```

Let $S_1\{k^*, t^*, z^*, w\}$ be the result of replacing the call to T_k in $S_{k,w}$ with a call to $T_1\{k^*, t^*, z^*\}$ and $\tilde{S} = iO(S_1\{k^*, t^*, z^*, w\})$. Let $\tilde{Q} = iO(Q_w)$. GM gives $\tilde{T}, \tilde{S}, \tilde{Q}$ and y^* to A. A returns x' and wins if $\tilde{T}(x') = y^*$.

$H_0 \approx H_1$: under our selection of k^*, t^*, z^*, it is clear that T_k and $T_1\{k^*, t^*, z^*\}$ have the exact same functionality, and so do $S_{k,w}$ and $S_1\{k^*, t^*, z^*, w\}$. Therefore, any significant difference between the advantage of A in H_0 and H_1 could be used to break the security of the iO scheme: let B an adversary for the iO scheme which runs GM in both hybrids to obtain T_k and $T_1\{k^*, t^*, z^*\}$, outputs them both and accepts back $\tilde{T}$ which is an obfuscation of one of the two, and similarly for $\tilde{S}$. It runs A on the programs it got and outputs 1 if A wins. If $\tilde{T}, \tilde{S}$ are obfuscations of $T_k, S_{k,w}$ then A is in H_0, and if they are of $T_1\{k^*, t^*, z^*\}$ and $S_1\{k^*, t^*, z^*, w\}$ then A is in H_1.

- H_2: Same as in H_1, only GM replaces $s^* = f_k(t^*)$ with a truly random s^*. That is, GM chooses keys k, w, a pre-image $x^* \leftarrow \{0,1\}^n$, $t^* = g(x^*)$, $s^* \leftarrow \{0,1\}^n$, $z^* = x^* \oplus s^*, y^* = (z^*, t^*), k^* = k(\{t^*\}), \tilde{T} = iO(T_1\{k^*, t^*, z^*\}), \tilde{S} = iO(S_1\{k^*, t^*, z^*, w\})$ and $\tilde{Q} = iO(Q_w)$. GM gives $\tilde{T}, \tilde{S}, \tilde{Q}, y^*$ to A. A returns x' and wins if $\tilde{T}(x') = y^*$.

  $H_1 \approx H_2$: assume otherwise, and let B be the following adversary for the security of the punctured PRF key $k^* = k(\{t^*\})$ at the punctured point t^*: B chooses x^* and takes $t^* = g(x^*)$. It gives out t^*, and gets back $k^* = k(\{t^*\})$ and a challenge s^* which is either $f_k(t^*)$ or random. B chooses w, generates $z^* = x^* \oplus s^*, y^* = (z^*, t^*), \tilde{T} = iO(T_1\{k^*, t^*, z^*\})$, $\tilde{S} = iO(S_1\{k^*, t^*, z^*, w\})$ and $\tilde{Q} = iO(Q_w)$. It runs A on $\tilde{T}, \tilde{S}, \tilde{Q}, y^*$ and outputs 1 if A wins. If $s^* = f_k(t^*)$ then A is in H_1, and if s^* is random then A is in H_2.

  Note that s^* is random in H_3, and is no longer in the adversary's view. Therefore we can completely remove it and treat z^* as truly random instead. That is, GM can choose keys k, w, a pre-image $x^* \leftarrow \{0,1\}^n, t^* = g(x^*), z^* \leftarrow \{0,1\}^n, y^* = (z^*, t^*), k^* = k(\{t^*\}), \tilde{T} = iO(T_1\{k^*, t^*, z^*\})$, $\tilde{S} = iO(S_1\{k^*, t^*, z^*, w\})$, $\tilde{Q} = iO(Q_w)$ and give $\tilde{T}, \tilde{S}, \tilde{Q}, y^*$ to A.

- H_3: same as in H_2, only GM replaces $t^* = g(x^*)$ with a random t^*. That is, GM chooses keys k, w, $t^* \leftarrow \{0,1\}^{2n}$, $z^* \leftarrow \{0,1\}^n, y^* = (z^*, t^*), k^* = k(\{t^*\}), \tilde{T} = iO(T_1\{k^*, t^*, z^*\})$, $\tilde{S} = iO(S_1\{k^*, t^*, z^*, w\})$, $\tilde{Q} = iO(Q_w)$ and give $\tilde{T}, \tilde{S}, \tilde{Q}, y^*$ to A. A returns x' and wins if $\tilde{T}(x') = y^*$.

  $H_2 \approx H_3$: Assume otherwise, and let B be the following adversary for the security of the pseudorandom generator g. B gets a value t^* which is either $g(x^*)$ on a random x^*, or a random $2n$-bits value. B then chooses $k, w, z^* \leftarrow \{0,1\}^n, y^* = (z^*, t^*), \tilde{T} = iO(T_1\{k^*, t^*, z^*\})$, $\tilde{S} = iO(S_1\{k^*, t^*, z^*, w\})$, $\tilde{Q} = iO(Q_w)$, runs A on $\tilde{T}, \tilde{S}, \tilde{Q}, y^*$, and outputs 1 if A wins. If $t^* = g(x^*)$ then A is in H_2, and if it is random then A is in H_3.

Finally, in H_3, the adversary A sees $y^* = (z^*, t^*)$ for random $z^* \leftarrow \{0,1\}^n$ and $t^* \leftarrow \{0,1\}^{2n}$, and the obfuscated programs, and needs to guess an x' such that $g(x') = t^*$ and $x' \oplus f_{k^*}(t^*) = z^*$. But, as g maps n bits to $2n$ bits, at most 2^{-n} items in $\{0,1\}^{2n}$ have a pre-image. Hence, for a random $t^* \leftarrow \{0,1\}^{2n}$, with all but negligible probability there exists no x' such that $g(x') = t^*$, and in particular A cannot guess any such x'. So, with all but negligible probability over the coins of y^*, A absolutely cannot invert it, meaning the advantage of A in H_3 is negligible.

□

5.3 Enhancements

Theorem 5.3. *The TDF family describes an **enhanced** 1-1 TDF.*

Proof. We will show that for any PPT adversary A, it holds that:

$$\Pr_{\substack{\alpha \leftarrow I_0(1^n) \\ r \leftarrow \{0,1\}^n}}[A(\alpha, r) = f_\alpha^{-1}(S_R(\alpha, r))] \leq \mu(n) \tag{6}$$

for some negligible function μ.

We describe the first enhancement as a game between a game master GM and an adversary A:

1. GM:
   - generates random $k \leftarrow K, w \leftarrow W$
   - $\alpha = (\tilde{T} = iO(T_k), \tilde{S} = iO(S_{k,w}), \tilde{Q} = iO(Q_w))$ and $\tau = k$
   - $r^* \leftarrow \{0,1\}^n$
   - Give $\tilde{T}, \tilde{S}, \tilde{Q}, r^*$ to A
2. A sees $\tilde{T}, \tilde{S}, \tilde{Q}, r^*$, outputs x, and wins if $\tilde{T}(x) = \tilde{S}(r^*)$.

We denote by $adv(A) = \Pr[\tilde{T}(x) = \tilde{S}(r^*)]$ the advantage of A in the above game. We next describe a series of hybrids. The first hybrid describes the first enhancement game between GM and A, as above. We show that the advantage on A between each two consecutive hybrids must be negligibly close, and that the advantage in the last hybrid must be negligible, which proves our claim. Note that k remains unpunctured and $\tilde{T}$ remains unchanged throughout the hybrids.

- H_0: the 1st enhancement game is played as described above.
- H_1: The game is the same as in H_0, only $S_{k,w}$ and Q_w are replaced other obfuscated programs, $S_1\{\tilde{T}, w^*, r^*, y^*\}$ and $Q_1\{w^*\}$, as described below:

1. GM:
   - generates random $k \leftarrow K, w \leftarrow W$
   - $\tilde{T} = iO(T_k)$
   - $r^* \leftarrow \{0,1\}^n$
   - $w^* = w(\{r^*\})$ is the punctured PRF key w at point r^*.
   - $x^* = h_w(r^*), y^* = T_k(x^*)$
   - $\tilde{S} = iO(S_1\{\tilde{T}, w^*, r^*, y^*\})$ (described below)
   - $\tilde{Q} = iO(Q_1\{w^*\})$ (described below)
   - Give $(\tilde{T}, \tilde{S}, \tilde{Q}, r^*)$ to A
2. A sees $\tilde{T}, \tilde{S}, \tilde{Q}, r^*$, outputs x, and wins if $\tilde{T}(x) = \tilde{S}(r^*)$.

where $S_1\{\tilde{T}, w^*, r^*, y^*\}$ and $Q_1\{w^*\}$ are the following programs:

$S_1\{\tilde{T}, w^*, r^*, y^*\}$:
```
    constants:
        obfuscated program T̃
        punctured PRF key w* for h
        r* ∈ {0, 1}^n
        y* ∈ {0, 1}^{3n}
    if r = r* then
        return y*
    x = h_{w*}(r)
    return EvalO(T̃, x)
```

$Q_1\{w^*\}(\rho)$:
```
    constants:
        punctured PRF key w* for h
    r = d(ρ)
    x = h_{w*}(ρ)
    return (x, r)
```

$H_0 \approx H_1$: $S_{k,w}$ and $S^*_{\tilde{T},w^*,r^*,y^*}$ are functionally equivalent: on all $r \neq r^*$ they both take $x = h_w(r) = f_{w^*}(r)$ and return $T_k(x)$. For r^*, $S_{k,w}$ returns $T_k(h_w(r^*))$, and $S_1\{\tilde{T}, w^*, r^*, y^*\}$ returns y^*, which is chosen by GM to be $T_k(h_w(r^*))$. As per Q_w and $Q_1\{w^*\}$: d is length-doubling, hence for a randomly selected $r^* \leftarrow \{0,1\}^n$, the probability that there exists a $\rho \in \{0,1\}^{n/2}$ such that $d(\rho) = r^*$ is negligible. Therefore, with all but negligible probability over the choice of r^*, Q_w and $Q_1\{w^*\}$ are functionally equivalent as well. So, if A's advantage between the two hybrids is none-negligible, we can construct an adversary B for the security of the iO scheme.

- H_2: the same as in H_1, only x^* (the pre-image of y^*) is taken to be a truly random string (rather than $h_w(r^*)$).

  1. GM:
     - generates random $k \leftarrow K, w \leftarrow W$
     - $\tilde{T} = iO(T_k)$
     - $r^* \leftarrow \{0,1\}^n$
     - $w^* = w(\{r^*\})$ is the punctured PRF key w at point r^*.
     - $x^* \leftarrow \{0,1\}^n, y^* = T_k(x^*)$
     - $\tilde{S} = iO(S_1\{\tilde{T}, w^*, r^*, y^*\})$
     - $\tilde{Q} = iO(Q_1\{w^*\})$
     - Give $(\tilde{T}, \tilde{S}, \tilde{Q}, r^*)$ to A
  2. A sees $\tilde{T}, \tilde{S}, \tilde{Q}, r^*$, outputs x, and wins if $\tilde{T}(x) = \tilde{S}(r^*)$.

  $H_1 \approx H_2$: suppose otherwise, and let B be the following adversary for the selective security of the punctured PRF key w^* at the punctured point r^*. B gives out r^* and gets the punctured key $w^* = w(\{r^*\})$ along with either $x^* = h_w(r^*)$ or a random x^*. It then selects k, generates $\tilde{T} = iO(T_k)$, $y^* = T_k(x^*)$, and generates $\tilde{S} = iO(S_1\{\tilde{T}, w^*, r^*, y^*\})$ and $\tilde{Q} = iO(Q_1\{w^*\})$. It runs A on $\tilde{T}, \tilde{S}, \tilde{Q}, r^*$ and returns 1 if A wins.

  Finally, we claim that if the advantage of A in H_2 is non-negligible, then the one-wayness of the trapdoor function is compromised, contradicting the hardness proof of our construction (section 5.2). Indeed, suppose A is able to provide x such that $\tilde{T}(x) = \tilde{S}^*(r^*)$, and let B be the following adversary for the hardness of the trapdoor function. B is given $\tilde{T} = iO(T_k), \tilde{S} = iO(S_{k,w'}), \tilde{Q} = iO(Q_{w'}), y^*$ and should output x' such that $\tilde{T}(x') = y^*$. B samples a key $w \leftarrow W$, takes $r^* \leftarrow \{0,1\}^n, w^* = w(\{r^*\})$, generates $\tilde{S} = iO(S\{\tilde{T}, w^*, r^*, y^*\})$, $\tilde{Q} = iO(Q_1\{w^*\})$ and runs A on $\tilde{T}, \tilde{S}, \tilde{Q}, r^*$. A outputs some value x' which, with non-negligible probability, provides $\tilde{T}(x') = \tilde{S}_1(r^*)$. By definition of S_1, we have that $\tilde{S}_1(r^*) = S_1\{\tilde{T}, w^*, r^*, y^*\}(r^*) = y^*$. So, by outputting x', B is able to invert y^* with non-negligible probability.

□

Theorem 5.4. *The TDF family describes a* ***doubly-enhanced*** *1-1 TDF.*

Proof. We claim that $\tilde{Q}$ provides a correlated-preimage sampler $S_{DR} = EvalO(\tilde{Q}; \rho)$. Clearly, $EvalO(\tilde{Q}; \rho)$ returns pairs of (x, r) such that $x = h_w(r)$, that is $\tilde{T}(x) = \tilde{T}(h_w(r)) = \tilde{S}(r)$.

The pseudorandomness of r, conditioned on $\alpha = (\tilde{T}, \tilde{S}, \tilde{Q})$ and x (which is either sampled along with r by running Q or inverted from $S_R(\alpha, r)$), follows directly from the pseudorandomness of d.

□

An interesting point about our construction is that both enhancements do not depend at all on the structure of our original TDF. In fact, all the enhancements need in order to work is any full-domain, or even efficiently sampleable domain, TDF, and the proof remains the same. Hence, our technique of re-randomizing the input via a length-preserving PRF can be considered as a generic method for doubly-enhancing any efficiently-sampleable-domain TDF, using iO and one-way functions.

5.4 Certifiable Injectivity

We show that our construction is perfectly certifiable injective, under the assumption that the PRG g is injective. Moreover, the soundness of the certification protocol is perfect. This shows that our construction is sufficient for realizing the FLS paradigm.

Recall that, on input x, our TDF evaluation returns $(x \oplus s, t)$, where $t = g(x)$ (and s is determined by the secret trapdoor). The certifier $ICert$ is given x, obtains $y = F(\alpha, x)$, and compares the last $2n$ bits of y to $g(x)$. If they are equal, $ICert$ accepts. Otherwise it rejects.

Theorem 5.5. *Assuming g is a full-domain injective PRG, our TDF family, along with $ICert$, is perfectly certifiable injective.*

Proof. For $y \in \{0,1\}^{3n}$, denote by $y[n+1 : ...3n]$ the last $2n$ bits of y.

1. Completeness: if $y = F(\alpha, x)$ for an honestly created α, then by the definition of our TDF we have $y = (c, t)$ for $t = g(x)$ and $c = x \oplus f_k(t)$. So $y[n+1 : ...3n] = t = g(x)$ and $ICert$ accepts.

2. Soundness: Suppose x_1, x_2, y such that $F(\alpha, x_1) = F(\alpha, x_2) = y$ and $ICert(\alpha, x_1) = ICert(\alpha, x_2) = 1$. By definition, since $ICert(\alpha, x_i) = 1$ for both x_1 and x_2, we have that $g(x_1) = y[n+1 : ...3n] = g(x_2)$. Since g is injective, this means $x_1 = x_2$.

The soundness, hardness and enhancements proofs for the TDF are not harmed, as $ICert$ does not depend on the private key k. □

5.5 On the Assumption of Full-Domain iPRGs

As mentioned in the opening of section 5, our construction and security proof rely on the assumption that the underlying PRGs and PPRFs are full-domain; That is, every string in $\{0,1\}^{p(n)}$ (where $p(n)$ is the size of the specific function's domain) can be mapped to a pre-image of the function. This assumption makes sense in the case of general PRGs and PPRFs, where natural full-domain candidates exists. However this is not the case for injective PRGs, which are required for our certifiable injectivity proof.

We first note that for the completeness, security and enhancements, the full-domain assumption can be relaxed by allowing functions with an efficiently sampleable domain.

The domain sampler is then used to map random coins, as well as the output of some of the primitives we use, into domain items.

Secondly, we show that the certifiable injectivity of our construction is maintained under the relaxed assumption of an injective PRG with a domain which is *efficiently recognizable* (as well as sampleable). That is, we require that there exists an efficient global domain recognizer algorithm Rec which, given some string $x \in \{0,1\}^n$, decides if that string is in the domain or not, and g is injective over the set of all strings which Rec accepts. Assuming the existence of such an efficient recognizer algorithm Rec, we modify $ICert$ such that given a supposed pre-image x, $ICert$ first runs $Rec(x)$. Only after, $ICert$ continues to compare the last $2n$ bits of $y = F(\alpha, x)$ to $g(x)$. It accepts only if both conditions passed. The CI soundness requirement follows directly.

We point out that the recognizable domain requirement is indeed necessary for certifiable injectivity. Without it, a malicious prover might be able to cheat using a similar attack to the one described in section 3: the prover can give pre-images taken outside of the PRG's supposed domain, on which $ICert$ might arbitrarily accept, and the verifier won't be able to tell the difference.

Finally, we demonstrate how injective pseudorandom generators with efficiently recognizable and sampleable domains can be constructed based on standard assumptions:

- Based on the DDH assumption [DH76], [Bon98] suggested the the following candidate for injective PRGs. Let $G_p = \{x^2 : x \in Z_p\}$, where p is a safe prime (that is $p = 2q + 1$ for some prime q). We define the following enumeration from G_q to Z_q (see e.g. [CS03, CFGP05]):

$$i(x) = \begin{cases} x & \text{if } 1 \leq x \leq q \\ p - x & \text{if } q + 2 \leq x \leq p \\ 0 & \text{otherwise} \end{cases}$$

  Let g be a generator for G_p. For $a, b \in Z_q$, let:

$$prg(a, b) = i(g^a), i(g^b), i(g^{ab})$$

  Then by the DDH assumption, prg is an injective pseudorandom generator from $Z_q^2 \to Z_q^3$. Using the same technique, an injective length-doubling PRG from $Z_q^3 \to Z_q^6$ can ge constructed by using

$$prg(a, b, c) = i(g^a), i(g^b), i(g^c), i(g^{ab}), i(g^{ac}), i(g^{bc})$$

- Assuming one-way permutations with an efficiently sampleable domain, an injective length-doubling pseudorandom generator can be obtained using the textbook

construction (c.f. [Gol98]). That is, let owp be a one-way permutation over domain $D_n \subseteq \{0,1\}^n$, and let hcb be a hard-core predicate for it. Then $prg_1(x) = (owp(x), hcb(x))$ is a pseudorandom generator which is single-bit expending. Fo4 $i > 1$, let $prg_i(x) := prg_{i-1}(owp(x)), hcb(x)$ be the result of recursively applying prg_q on the first n bits of the output. Using an hybrid argument, $prg_n(x)$ is a injective length-doubling PRG. Constructing an injective pseudorandom generator from primitives weaker then one-way permutations remains an open question[4].

For the certifiable injectivity of our TDP construction, we require that the PRG's domain, D_n, be efficiently recognizable. However when this is the case additional attention is required, since the first n bits of $f(x)$ describe an element in that domain, and hence they are clearly distinguishable from just any n-bit string. We circumvent this issue by defining our PRG as pseudorandom with respect to $D_n \circ U_n := \{(x,s) : x \leftarrow D_n, s \leftarrow \{0,1\}^n\}$. That is, we adapt the security requirement of the PRG to the following: for any PPT adversary A, $\Pr[x \leftarrow D_n : A(prg_n(x)) = 1] - \Pr[x \leftarrow D_n, s \leftarrow \{0,1\}^n : A((x,s)) = 1] < \mu(n)$, where $\mu(n)$ is negligible. Under the revised definition, our security proof remains sound, with the change that when replacing $t^* = prg_n(x^*)$ with a random t^*, the replaced value is taken out of $D_n \circ U_n$ (instead of a random $2n$-bit string).

An injective one-way permutation with an efficiently recognizable domain can be obtained, e.g., based on the discrete log assumption.

Acknowledgments

We thank Oded Goldreich and Ron Rothblum for their very useful comments.

References

[AS15] Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2015/752, 2015. http://eprint.iacr.org/2015/752.

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on computing*, 15(2):364–383, 1986.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

[4] [Rud84, KSS00, MM11] give a black-box separation between one-way permutations and weaker primitives, such as one-way functions

[BG84] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 289–299. Springer, 1984.

[BG90] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 194–211. Springer, 1990.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual International Cryptology Conference*, pages 1–18. Springer, 2001.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *International Workshop on Public Key Cryptography*, pages 501–519. Springer, 2014.

[Bon98] Dan Boneh. The decision diffie-hellman problem. *Algorithmic number theory*, pages 48–63, 1998.

[BP14] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/295, 2014. http://eprint.iacr.org/2014/295.

[BPW15] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos. Cryptology ePrint Archive, Report 2015/126, 2015. http://eprint.iacr.org/2015/126.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 280–300. Springer, 2013.

[BY96] Mihir Bellare and Moti Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996.

[CFGP05] Olivier Chevassut, Pierre-Alain Fouque, Pierrick Gaudry, and David Pointcheval. Key derivation and randomness extraction. *IACR Cryptology ePrint Archive*, 2005:61, 2005.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 308–317. IEEE, 1990.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent. In *Annual International Cryptology Conference*, pages 228–245. Springer, 1992.

[Gol98] Oded Goldreich. Foundation of cryptography, february 1995, 1998.

[Gol04] Oded Goldreich. Foundations of cryptography. basic applications, vol. 2, 2004.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM SIGACT News*, 39(3):35–39, 2008.

[Gol11] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 406–421. Springer, 2011.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 339–358. Springer, 2006.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM (JACM)*, 59(3):11, 2012.

[GR13] Oded Goldreich and Ron D Rothblum. Enhancements of trapdoor permutations. *Journal of cryptology*, 26(3):484–512, 2013.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–432. Springer, 2008.

[KP98] Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.

[KSS00] Jeff Kahn, Michael Saks, and Cliff Smyth. A dual version of reimer's inequality and a proof of rudich's conjecture. In *Computational Complexity, 2000. Proceedings. 15th Annual IEEE Conference on*, pages 98–103. IEEE, 2000.

[MM11] Takahiro Matsuda and Kanta Matsuura. On black-box separations among injective one-way functions. In *Theory of Cryptography Conference*, pages 597–614. Springer, 2011.

[Rab79] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, DTIC Document, 1979.

[Rot10] Ron Rothblum. A taxonomy of enhanced trapdoor permutations. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 17, page 145, 2010.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Rud84] Steven Rudich. *Limits on the provable consequences of one-way functions*. PhD thesis, Wesleyan University, 1984.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint.iacr.org/2013/454.

[Yao82] Andrew C Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 80–91. IEEE, 1982.