Perfect Structure on the Edge of Chaos

Trapdoor Permutations from Indistinguishability Obfuscation

Nir Bitansky* Omer Paneth† Daniel Wichs‡

June 20, 2017

Abstract

We construct trapdoor permutations based on (sub-exponential) indistinguishability obfuscation and one-way functions, thereby providing the first candidate that is not based on the hardness of factoring.

Our construction shows that even highly structured primitives, such as trapdoor permutations, can be potentially based on hardness assumptions with *noisy structures* such as those used in candidate constructions of indistinguishability obfuscation. It also suggest a possible way to construct trapdoor permutations that resist quantum attacks, and that their hardness may be based on problems outside the complexity class SZK — indeed, while factoring-based candidates do not possess such security, future constructions of indistinguishability obfuscation might.

As a corollary, we eliminate the need to assume trapdoor permutations and injective one-way function in many recent constructions based on indistinguishability obfuscation.

*MIT. Email: nirbitan@csail.mit.edu.

†Boston University. Email: omer@bu.edu. Supported by the Simons award for graduate students in theoretical computer science and an NSF Algorithmic foundations grant 1218461.

‡Northeastern University. Email: wichs@ccs.neu.edu. Supported by NSF grants CNS-1347350 and CNS-1314722.

1 Introduction

In the mid '70s and early '80s, powerful number-theoretic constructions related to factoring and discrete-logs kick-started modern cryptography. As these constructions gradually evolved into a comprehensive theory, generic primitives, such as one-way functions, collision-resistent hash functions, and trapdoor permutations, were defined with the aim of abstracting the properties needed in different applications. The aforementioned number-theoretic problems provided instantiations for each of these primitives, but at the same time appeared to offer a much richer algebraic structure. Whereas this structure is highly fruitful, it also limits the hardness of the corresponding problems to low complexity classes such as statistical zero-knowledge (SZK) [GK88] and makes them susceptible to quantum attacks [Sho97]. Therefore, a fundamental goal is to base cryptographic primitives on other less structured assumptions.

In some cases, such as one-way functions, it seems that we can avoid structured assumptions altogether. Indeed, one-way functions can be constructed generically from essentially any cryptographic primitive and have candidates from purely combinatorial assumptions [BFKL93, Gol11, JP00, AC08]. However, as we consider primitives that intrinsically require some structure, candidates become more scarce. For example, *injective* one-way functions are only known based on assumptions with some *algebraic homomorphism*, albeit these may feature noisy structures, such as the ones arising from lattices [PW08]. In particular, such assumptions can be placed in SZK, but are not known to be susceptible to quantum attacks. If we also require the one-way function to be a *permutation*, candidates become even more scarce, and only known based on the the hardness of discrete-logs and factoring (or RSA) [RSA83, Rab79]. Further, *trapdoor permutations* (TDPs) are known exclusively based on factoring (or RSA).

Obfuscation. A promising source for new constructions, replacing ones that so far depended exclusively on specific algebraic assumptions, is *program obfuscation* — a method for shielding programs such that their implementation becomes hidden. Indeed, an ideal notion of obfuscation would allow us to securely express any required structure in the obfuscated programs. Understanding to what extent this intuition can be fulfilled requires looking into concrete notions of secure obfuscation. The question is what is the "right" notion to consider and under what kind of assumptions it can be achieved.

Of particular interest is the notion of *indistinguishability obfuscation* (iO), which have recently found candidate constructions [GGH$^+$13b]. The notion of iO requires that the obfuscations of any two programs of the same size and functionality are indistinguishable. While this notion may not capture *ideal obfuscation*, it turns out to be sufficient for many known cryptographic primitives, suggesting an alternative for previous number theoretic constructions [SW14, BP14, CLTV14].

From an assumption perspective, the existing constructions of iO [GGH$^+$13b, BR14, BGK$^+$13, AB15, Zim15, AJ15, BV15, GLSW14] are instantiated based on multi-linear graded encodings [GGH13a, CLT13, CLT15] thus falling into Gentry's [Gen14] world of *computing on the edge of chaos* — they all rely on *noisy structures* in an essential way. Beyond the existing candidates, understanding on which assumptions iO can be based (and how structured they should be) is an open question; in particular, as far as we know, future constructions of iO may be based on problems outside AM$\cap$coAM and/or outside BQP.

1.1 This Work

Our main result is a construction of trapdoor permutations based on sub-exponential indistinguishability obfuscation and one-way functions. As far as we know, this is the first construction of trapdoor permutations since the introduction of the RSA and Rabin trapdoor permutations [RSA83, Rab79] (and their variants). We also construct injective one-way functions based on standard iO and one-way functions. As a tool used in our constructions and a result of potentially independent interest, we show how to convert any one-way function into a *sometimes-injective one-way function* that is simultaneously injective

and hard-to-invert on some sub-domain of noticeable density.

Properties. Our permutations have the following additional features. First, they are doubly-enhanced. Additionally, they can be generated so to have any *prescribed cycle structure* with the necessary property that small cycles are rare enough. So far this property has only been achieved for pseudo-random permutations [NR02]. Another feature is that inverting the permutation consists of simple symmetric-key operations (unlike in existing candidates). Finally, like in the RSA permutation, given the trapdoor it is possible to iterate the permutation (or its inverse) any number of times at the same cost as computing the function once.

One difference between the trapdoor permutations we construct and those typically defined in the literature [GR13] is that we only support sampling of *pseudo-uniform* elements in the domain rather than elements that are statistically close to uniform. The sampled elements are pseudo-uniform in a strong sense, namely, even given the trapdoor or more generally the coins used to sample the function. The double-enhancement requirement is relaxed in a somewhat similar manner (see details below). As far as we know, these relaxation are sufficient in known applications. Additionally we note that our permutations are not *certifiable*, meaning that we do not know of an efficient way to certify that a key is well-formed and describes a valid permutation.

iO as a hub. Based on our results, several constructions previously based on iO and additional structured assumptions, can now be based only on iO and one-way functions (or rather the assumption that NP $\neq$ coRP [KMN$^+$14]). Examples include: non-interactive commitments [Blu81],[1] actively secure two-message oblivious transfer [SW14], non-interactive witness-indistinguishable proofs [BP14], obfuscation for Turing machines [KLW14],[2] hardness of the complexity class PPAD [BPR15], and more.

1.2 Technical Overview

Trapdoor permutations from obfuscation? Sounds easy. Thinking of obfuscation in ideal terms gives rise to a natural attempt at constructing TDPs: *simply obfuscate a pseudo-random permutation*. Clearly, with only black-box access to such a permutation, inversion is as impossible as inverting a random function. However, ideal virtual black-box obfuscation [BGI$^+$01] of pseudo-random permutations is unknown, and is in fact subject to strong limitations [BCC$^+$14]. Nevertheless, we show that some of this intuition can be recovered also when relying on the (not so ideal) notion of iO.

Outline of the construction. Our starting point is a recent construction suggested by Bitansky et al. [BPR15] to demonstrate the hardness of the complexity class PPAD. We observe that their construction can, in fact, be viewed as a trapdoor permutation family that lacks a crucial property: *it does not allow to sample elements from the permutation's domain*.

Our construction then follows the three steps below:

1. We construct a sampler for domain elements and prove the one-wayness of the permutation family even given this sampler. This involves extending the techniques developed in [BPR15].

2. We further augment the permutation family and sampler so that they will admit the requirements of enhanced and doubly-enhanced TDPs.

3. The construction of [BPR15] relies on injective one-way functions in addition to iO. We construct such injective one-way functions based on iO and one-way functions. We find this construction to be of independent interest.

[1] See Appendix B for more details on the construction of non-interactive commitments.

[2] Formally, [KLW14] rely on injective PRGs, but these can be replaced with injective one-way functions using an observation of [BCP14], as explained in Section 4.2.

We now elaborate on the construction and analysis in [BPR15], describe where it falls short of achieving an actual TDP, and then turn to describe our solutions.

1.2.1 A Closer Look into [BPR15]

Bitansky et al. construct a hard instance of the END-OF-THE-LINE problem based on iO. In the END-OF-THE-LINE problem we consider a sequence of nodes

$$x_1 \to x_2 \to \cdots \to x_T \ ,$$

and a program F that maps x_i to x_{i+1} for $1 \leq i < T$. The problem is given the source node x_1 and the program F find the sink node x_T. In the construction of [BPR15], each node x_i is a pair $(i, \mathsf{PRF}_S(i))$ where $\mathsf{PRF}_S : \mathbb{Z}_T \to \{0,1\}^\lambda$ is sampled from a family of pseudo-random functions, and $T \in \mathbb{N}$ is super-polynomial in the security parameter λ. The instance also contains an obfuscated program $\widetilde{\mathsf{F}}$ that maps x_i to x_{i+1} and outputs $\perp$ on any other input.

Bitansky et al. show that given strong enough iO and injective one-way functions (used only in the analysis) it is hard to find x_T given x_1 and the obfuscated program $\widetilde{\mathsf{F}}$. Intuitively, the path from x_1 to x_T can be thought of as an authenticated chain where a signature σ corresponding to some pair (i, σ) cannot be obtained without first obtaining all previous signatures on the path. It is not difficult to show that any efficient algorithm that only invokes $\widetilde{\mathsf{F}}$ as a black box cannot find the signature $\mathsf{PRF}_S(T)$. Their proof shows that the same hardness holds even given full access to the obfuscated program $\widetilde{\mathsf{F}}$.

Constructing trapdoor permutations. Indeed, the construction of [BPR15] described above gives rise to a natural candidate for a trapdoor permutation. A given permutation is over the set of nodes $\{x_i\}_{i \in \mathbb{Z}_T}$ and is defined by the cycle

$$x_1 \to x_2 \to \cdots \to x_T \to x_1 \ .$$

The public key describing the permutation consists of the obfuscated program $\widetilde{\mathsf{F}}$ that maps x_i to x_{i+1} (where $i+1$ is computed modulo T) and outputs $\perp$ on any other input. The trapdoor is simply the seed S of the pseudo-random function that allows us to efficiently invert the permutation. However, without the trapdoor, inverting the permutation on x_i is as hard as finding the end of the chain starting at x_i and ending at x_{i-1}.

To obtain a complete construction of a TDP, we need to specify how to sample random domain elements. The challenge here is that the domain of our permutation is very *sparse* and it is not clear how to sample from it without the trapdoor S. A naive suggestion is to include, as part of the public key, an obfuscated sampler program that given i outputs the node x_i. However, publishing such a program (obfuscated or not) clearly makes the permutation easy to invert. To explain, how this is solved, we now look more closely into the security proof of [BPR15].

The proof of [BPR15]. We sketch the argument from [BPR15] showing that the basic TDPs construction above (without any domain sampler) is one-way. That is, given $\widetilde{\mathsf{F}}$ and x_i for a random $i \in \mathbb{Z}_T$, it is hard to obtain x_{i-1} (in fact we prove this for every $i \in \mathbb{Z}_T$). A more detailed overview of the proof of [BPR15] can be found in Appendix A. To prove that finding the node x_{i-1} is hard it is sufficient to prove that the obfuscated circuit $\widetilde{\mathsf{F}}$ is computationally indistinguishable from a circuit that on input x_{i-1} returns $\perp$, rather than x_i as $\widetilde{\mathsf{F}}$ would. Indeed, any algorithm that can find x_{i-1} can also distinguish the two circuits. We next explain how indistinguishability of these two circuits is shown.

For every $\alpha, \beta \in \mathbb{Z}_T$ we consider the circuit $\widetilde{\mathsf{F}}_{\alpha,\beta}$ that is identical to $\widetilde{\mathsf{F}}$, except that for every j in the range from α to β (wrapping around T in case that $\alpha > \beta$) $\widetilde{\mathsf{F}}_{\alpha,\beta}$ on the input x_j outputs $\perp$. The argument proceeds in two steps.

Step 1: split the chain into two parts. We show that for a *random* $u \in \mathbb{Z}_T$, the obfuscation $\widetilde{\mathsf{F}}_{u,u}$ is computationally indistinguishable from $\widetilde{\mathsf{F}}$. Intuitively this "splits" the authenticated chain into two

parts: from x_i to x_u and from x_{u+1} to x_{i-1}. The proof of this step relies on the fact that the chain is of super-polynomial length T and therefore the index u of a random node in the chain is hard to guess.

Step 2: erase the second part of the chain. We show that after the chain is split, is it hard to find any node x_j in the second part of the chain. Formally, we prove that the obfuscated circuits $\widetilde{\mathsf{F}}_{u,u}$ and $\widetilde{\mathsf{F}}_{u,i-1}$ are computationally indistinguishable. The proof is by a sequence of hybrids: for every j in the range between u and $i-2$, we rely on injective one-way functions and iO with super-polynomial hardness to show that the obfuscated circuits $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ are $T^{-\Theta(1)}$-indistinguishable. To prove that we can indistinguishably change the output of $\widetilde{\mathsf{F}}_{u,j}$ on the node x_{j+1} to $\perp$, we rely on the fact that in the circuit $\widetilde{\mathsf{F}}_{u,j}$ the successor of x_j is already erased and therefore, the circuit $\widetilde{\mathsf{F}}_{u,j}$ never explicitly outputs the node x_{j+1}.

1.2.2 Sampling from the Domain

As mentioned before, to allow sampling of elements in the domain we cannot simply provide a circuit that outputs x_i given $i \in \mathbb{Z}_T$, as this would result in an obvious attack — given $x_i = (i, \mathsf{PRF}_S(i))$, one can directly obtain the preimage x_{i-1}. The idea is to provide instead an obfuscation $\widetilde{\mathsf{X}}$ of a sampler X_S that is supported on a very sparse, but still pseudo-random, subset of the domain. Concretely, X_S takes as input a seed s for a length doubling pseudo-random generator $\mathsf{PRG} : \mathbb{Z}_{\sqrt{T}} \to \mathbb{Z}_T$, and outputs x_i for $i = \mathsf{PRG}(s)$.

First, note that by pseudo-randomness, inverting x_{i+1} where $i = \mathsf{PRG}(s)$ is pseudorandom is as hard as inverting x_{i+1} when i is chosen truly at random. Thus, we fucus on showing that inverting x_i is hard for a truly random i even in the presence of the obfuscated sampler $\widetilde{\mathsf{X}}$.

The one-wayness proof described above, however, fails when the adversary is given the sampler $\widetilde{\mathsf{X}}$. The problem is that in the second step, when arguing that the obfuscated circuits $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ are indistinguishable, we used the fact that $\widetilde{\mathsf{F}}_{u,j}$ never explicitly outputs the node x_{j+1}. However, if $j+1$ is in the image of PRG, the sampler $\widetilde{\mathsf{X}}$ explicitly outputs x_{j+1} and we can no longer prove that $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ are indistinguishable.

Our solution is to consider, instead of the entire chain starting from x_i and ending at x_{i-1}, only a suffix of this chain of length $\sqrt[4]{T}$ starting from $x_{i-\sqrt[4]{T}}$ and ending at x_{i-1}. On the one hand, this chain segment is still of super-polynomial length, and therefore, we can still split the segment following Step 1 above. On the other hand, the segment is also not too large (of density $T^{-3/4}$ in $\mathbb{Z}_T$). Since that segment starts at a random index $i - \sqrt[4]{T}$, and since the image of PRG is of size only $\sqrt{T}$, we have that with overwhelming probability $1 - T^{-1/4}$ the segment interval will not contain any nodes in the support of the sampler $\widetilde{\mathsf{X}}$. When the segment and the support of $\widetilde{\mathsf{X}}$ are disjoint, we can again erase the entire chain segment following Step 2 above.

Enhancements. In applications of TDPs, it is often required that the TDPs are *enhanced* or even *doubly enhanced* [GR13]. We briefly recall these properties and explain how they are obtained. In enhanced TDPs, we essentially ask that it is possible to *obliviously* sample domain elements, without knowing their pre-images. Translating to our setting, we require that $x_{\mathsf{PRG}(s)} \leftarrow \mathsf{X}_S$ is hard to invert, even given the coins s used to sample it. In the construction above, this may not be true. Indeed, given the seed s for the pseudo-random generator, we can no longer argue that inversion is as hard as for a truly uniform element. In fact, the PRG may be such that given s, it is easy to find s' such that $\mathsf{PRG}(s') = \mathsf{PRG}(s) - 1$ and thus easily invert. We observe that this can be circumvented if we make sure that PRG has a *discrete image* where a random image $\mathsf{PRG}(s)$ is likely to be isolated away from any other image. We show how to construct such PRGs from plain PRGs and pairwise-independent permutations.

In doubly enhanced TDPs, it is typically required that it is possible to sample an image-preimage pair (x, y) together with random coins used to sample the preimage y by the usual sampler. In our setting, we would like to sample an image $y = x_{\mathsf{PRG}(s)} \leftarrow \mathsf{X}_S$ together with randomness s and preimage $x_{\mathsf{PRG}(s)-1}$.

We only achieve a relaxed form of this requirement, where s is pseudo-random rather than truly random, even given the trapdoor, or the coins used to sample the function. The idea is to slightly change the pseudo-random generator PRG in the previous constructions in a way that exploits the specific structure of our TDP. We only change PRG on a sparse set of seeds that has negligible density, and thus previous properties are preserved (see more details in Section 4.3).

1.2.3 Injective One-Way Functions from iO

We now describe the main ideas behind constructing injective one-way function from iO and plain one-way functions. We rely on two-message statistically-binding commitment schemes [Nao91] and puncturable PRFs (both known from any one-way function). In the constructed family, every function $\mathsf{OWF}_{M_1,S}$ is associated with a first message M_1 for the commitment scheme and a pseudo-random function PRF_S. The public description of the function contains an obfuscated circuit $\widetilde{C}$ that on input x outputs a commitment $\mathsf{COM}_{M_1}(x; \mathsf{PRF}_S(x))$ with respect to the first commitment message M_1, plaintext x and randomness $\mathsf{PRF}_S(x)$. The fact that the function is injective (with overwhelming probability over M_1) follows directly from the statistical binding of the commitment. We focus on arguing one-wayness.

Our goal is to show that it is hard to recover a random x given $\widetilde{C}$ and $\widetilde{C}(x)$. We start by considering a hybrid circuit defined similarly to $\widetilde{C}$ except that it contains the punctured key $S\{x\}$ and given input x, it outputs a hardcoded commitment; since we did not change the functionality of the circuit indistinguishability follows by iO. Using pseudo-randomness at the punctured point x and the hiding of the commitment we can now argue that the hardcoded commitment hides x, replacing it with a commitment to some arbitrary plaintext, using true randomness. The problem is that now, even if we unpuncture $S\{x\}$, x itself still needs to appear in the clear as part of the code of the circuit in order to trigger the output of the hardcoded commitment.

Nevertheless, we may try to apply a similar strategy to the one previously used for our TDPs. Concretely, we note that x is only used to test if an input x' satisfies $x' = x$, and this comparison can be performed in an "encrypted form" — instead of hardcoding x in the clear we can hardcode $g(x)$ for some one-way function g and compare images instead of preimages. Unfortunately, to argue that this does not change functionality *the function g must itself be injective* which seems to bring us back to square one.

The key observation is that we may gain by using a function g that is only *sometimes injective*; namely, it is enough that g is simultaneously injective and hard to invert only on some noticeable subset of its domain. We show that such functions can be constructed from any one-way function. Now, leveraging the iO requirement only on the corresponding injective sub-domain, we can show that the above construction results in a *weak* one-way function that is *fully injective*; indeed, we only invoke sometimes-injective of g in the proof of one-wayness. Then, to obtain a (strong) injective one-way function, we can apply standard direct-product amplification [Yao82].

Constructing sometimes injective one-way functions. We outline the main idea behind constructing a sometimes-injective one-way function g, as above, based on any one way function f. First, consider for simplicity the case that the function f is r-regular. Roughly, the idea is extract the $\log(r)$ bits of randomness that remain in x conditioned on $f(x)$ and append them to the function output as in [HILL99]. However, due to their inherent entropy loss, standard randomness extractors cannot extract enough random bits to guarantee any meaningful injectiveness. Nevertheless, for our purpose, the extracted bits need not be statistically-close to uniform, they only need to preserve one-wayness. Accordingly, we use the *unpredictability extractors* of [DPW14], which allow extracting more bits so to guarantee injectiveness, while still preserving meaningful one-wayness.

To deal with f that is not regular, we may set r to be the most frequent regularity of f. This only shrinks the portion of the domain where f is both injective and hard to invert by some polynomial factor.

A uniform construction is obtained by choosing r at random.

2 Preliminaries

The cryptographic definitions in the paper follow the convention of modeling security against non-uniform adversaries. An efficient adversary $\mathcal{A}$ is modeled as a sequence of circuits $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, such that each circuit $\mathcal{A}_\lambda$ is of polynomial size $\lambda^{O(1)}$ with $\lambda^{O(1)}$ input and output bits; we shall also consider adversaries of some super polynomial size $t(\lambda) = \lambda^{\omega(1)}$. We often omit the subscript λ when it is clear from the context. The resulting hardness will accordingly be against non-uniform algorithms. The result can be cast into the uniform setting, with some adjustments to the analysis.

2.1 Indistinguishability Obfuscation

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

Definition 2.1 (Indistinguishability obfuscation [BGI$^+$01])**.** *A PPT algorithm $i\mathcal{O}$ is said to be an* indistinguishability obfuscator *for a class of circuits $\mathcal{C}$, if it satisfies:*

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

$$\Pr_{i\mathcal{O}}[\forall x : i\mathcal{O}(C)(x) = C(x)] = 1 \ .$$

2. **Indistinguishability:** *for any polysize distinguisher $\mathcal{D}$ there exists a negligible function $\mu(\cdot)$, 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:*

$$|\Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1]| \leq \mu(\lambda) \ ,$$

where the probability is over the coins of $\mathcal{D}$ and $i\mathcal{O}$.

We further say that $i\mathcal{O}$ is (t, δ)-secure, for some function $t(\cdot)$ and concrete negligible function $\delta(\cdot)$, if for all $t(\lambda)^{O(1)}$ distinguishers the above indistinguishability gap $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

2.2 Puncturable Pseudorandom Functions

We consider a simple case of the puncturable pseudo-random functions (PRFs) where any PRF may be punctured at a single point. The definition is formulated as in [SW14], and is satisfied by the GGM [GGM86] PRF [BW13, KPTZ13, BGI14],

Definition 2.2 (Puncturable PRFs)**.** *Let n, k be polynomially bounded length functions. An efficiently computable family of functions*

$$\mathcal{PRF} = \left\{\mathsf{PRF}_S : \{0,1\}^{n(\lambda)} \to \{0,1\}^\lambda : S \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N}\right\} \ ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{PRF}}$, 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 $S\{x^*\}$, so that the following conditions are satisfied:*

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

$$\Pr_{S\leftarrow\mathcal{K}_{\mathcal{PRF}}(1^\lambda)}\left[\forall x \neq x^* : \mathsf{PRF}_S(x) = \mathsf{PRF}_{S\{x^*\}}(x) : S\{x^*\} = \mathsf{Punc}(S, x^*)\right] = 1 \ .$$

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

$$|\Pr[\mathcal{D}(x^*, S\{x^*\}, \mathsf{PRF}_S(x^*)) = 1] - \Pr[\mathcal{D}(x^*, S\{x^*\}, u) = 1]| \leq \mu(\lambda) \ ,$$

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

We further say that $\mathcal{PRF}$ is (t, δ)-secure, for some function $t(\cdot)$ and concrete negligible function $\delta(\cdot)$, if for all $t(\lambda)^{O(1)}$ distinguishers the above indistinguishability gap $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

2.3 Injective One-Way Functions

We shall also rely on (possibly keyed) injective one-way functions.

Definition 2.3 (Injective OWF)**.** *Let k be polynomially bounded length function. An efficiently computable family of functions*

$$\mathcal{OWF} = \left\{ \mathsf{OWF}_K : \{0,1\}^\lambda \to \{0,1\}^* : K \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N} \right\} \ ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{OWF}}$, is an injective OWF if it satisfies

1. **Injectiveness:** *With overwhelming probability over the choice of $K \leftarrow \mathcal{K}_{\mathcal{OWF}}(1^\lambda)$, the function OWF_K is injective.*

2. **One-wayness:** *For any polysize inverter $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$, such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathcal{A}(K, \mathsf{OWF}_K(x)) = x : \begin{array}{c} K \leftarrow \mathcal{K}_{\mathcal{OWF}}(1^\lambda) \\ x \leftarrow \{0,1\}^\lambda \end{array} \right] \leq \mu(\lambda) \ .$$

We further say that $\mathcal{OWF}$ is (t, δ)-secure, for some function $t(\cdot)$ and concrete negligible function $\delta(\cdot)$, if for all $t(\lambda)^{O(1)}$ inverters the above inversion probability $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.

3 Injective One-Way Functions from iO

In this section, we construct injective one-way functions from iO and plain injective one-way functions. We start by defining and constructing *sometimes injective one-way functions*.

3.1 Sometimes Injective One-Way Functions

For a function $f : \{0,1\}^\lambda \to \{0,1\}^*$ and any input $x \in \{0,1\}^\lambda$, we denote by

$$\begin{aligned} &\mathbf{H}_f(x) := \log \left| \{x' : f(x') = f(x)\} \right| = \mathbf{H}_\infty \left(x' \leftarrow f^{-1}(f(x)) \right) \ , \\ &\mathbf{Inj}(f) := \{x : \mathbf{H}_f(x) = 0\} \end{aligned}$$

the min-entropy of a random preimage of $f(x)$, and the subset of inputs on which f is injective, respectively.

We next define sometimes-injective OWFs (SIOWFs). Roughly speaking, such functions are injective and hard to invert over a noticeable fraction of their domain.

Definition 3.1 (Sometimes-Injective OWF)**.** *Let k be polynomially bounded length function. An efficiently computable family of functions*

$$\mathcal{SIOWF} = \left\{ \mathsf{SIOWF}_K : \{0,1\}^\lambda \to \{0,1\}^* : K \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N} \right\} \ ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{SIOWF}}$, is a sometimes injective OWF if for every key $K \in \{0,1\}^{k(\lambda)}$ there exists an injective subset $\mathbf{I}_K \subseteq \mathbf{Inj}(\mathsf{SIOWF}_K)$, satisfying the following conditions:

1. **Sometimes injectiveness:** *There exists a polynomial $p(\cdot)$ such that for any $\lambda \in \mathbb{N}$:*

$$\Pr\left[x \in \mathbf{I}_K : \begin{array}{c} K \leftarrow \mathcal{K}_{\mathcal{SIOWF}}(1^\lambda) \\ x \leftarrow \{0,1\}^\lambda \end{array} \right] \geq 1/p(\lambda) \ .$$

2. **One-wayness over injective subdomain:** *for any polysize inverter $\mathcal{A}$ there is a negligible function $\mu(\cdot)$ such that for any $\lambda \in \mathbb{N}$:*

$$\Pr\left[\mathcal{A}(K, \mathsf{SIOWF}_K(x)) = x \wedge x \in \mathbf{I}_K : \begin{array}{c} K \leftarrow \mathcal{K}_{\mathcal{SIOWF}}(1^\lambda) \\ x \leftarrow \{0,1\}^\lambda \end{array} \right] \leq \mu(\lambda) \ .$$

   We further say that $\mathcal{SIOWF}$ is t-secure, for some super-polynomial function $t(\cdot)$, if the one-wayness requirement holds for all $t(\lambda)^{O(1)}$ inverters.

The construction. Let $f : \{0,1\}^* \to \{0,1\}^*$ be any one-way function. We construct an SIOWF

$$\mathcal{SIOWF} = \left\{ \mathsf{SIOWF}_K : \{0,1\}^\lambda \to \{0,1\}^* : K \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N} \right\} \ ,$$

with a corresponding key sampler $\mathcal{K}_{\mathcal{OWF}}$ as follows:

- A random key $K := (S, e) \leftarrow \mathcal{K}_{\mathcal{SIOWF}}(1^\lambda)$ consists of a random $e \leftarrow [\lambda]$ and a random seed S for a hash function $\mathsf{h}_S : \{0,1\}^\lambda \to \{0,1\}^{e+1}$ drawn from a q-wise independent family, where we set $q = \lambda$ to be the security parameter.
- For $x \in \{0,1\}^\lambda$, the function is defined by $\mathsf{SIOWF}_K(x) := (f(x), \mathsf{h}_S(x))$.

Proposition 3.1. *$\mathcal{SIOWF}$ is a sometimes injective one-way function.*

Proof. Throughout, we denote by $E_e \subseteq \{0,1\}^\lambda$ the subset of values x such that $\mathbf{H}_f(x) \in [e-1, e)$. For $K = (S, e)$, we define $\mathbf{I}_{S,e} = E_e \cap \mathbf{Inj}(\mathsf{SIOWF}_{S,e}) \subseteq \mathbf{Inj}(\mathsf{SIOWF}_{S,e})$. We start by proving the following preliminary claim saying that the function is injective with high-probability over the set E_e.

Claim 3.1. *For any $\lambda \in \mathbb{N}, e \in [\lambda], x \in E_e$*

$$\Pr_S \left[x \in \mathbf{Inj}(\mathsf{SIOWF}_{S,e}) \right] \geq \frac{1}{2} \ .$$

Proof of Claim 3.1. Fix any $\lambda \in \mathbb{N}, e \in [\lambda], x \in E_e$, and let $y = f(x)$. Since the output of $\mathsf{SIOWF}_{S,e}(x)$ includes y, it suffices to show that x does not collide with any other $x' \in f^{-1}(y)$. By q-wise independence (in fact, pairwise is sufficient here), for any such x',

$$\Pr_S \left[\mathsf{h}_S(x) = \mathsf{h}_S(x') \right] \leq 2^{-e-1} \ .$$

Thus, the expected number of such x' that collide with x is:

$$2^{-e-1}(\left|f^{-1}(y)\right| - 1) \leq 2^{-e-1} \cdot 2^{\mathbf{H}_f(x)} \leq 1/2 \ ,$$

and the claim now follows by Markov's inequality. □

Sometimes injectiveness follows directly:

$$\Pr\left[x \in \mathbf{I}_{S,e} : \begin{array}{c}(S,e) \leftarrow \mathcal{K}_{\mathcal{SIOWF}}(1^\lambda) \\ x \leftarrow \{0,1\}^\lambda\end{array}\right] \geq$$
$$\Pr_{e,x}[x \in E_e] \cdot \min_{e, x \in E_e} \Pr_S[x \in \mathbf{Inj}(\mathsf{SIOWF}_{S,e})] \geq \frac{1}{2\lambda} \ .$$

where $\Pr_{e,x}[x \in E_e] = 1/\lambda$ since for any fixed x there is a unique $e \in [\lambda]$ such that $x \in E_e$, and $\min_{e,x\in E_e} \Pr_S[x \in \mathbf{Inj}(\mathsf{SIOWF}_{S,e})] \geq 1/2$ by the previous claim.

Next, we prove one-wayness over the injective subdomain. For this, we rely on a theorem from [DPW14] showing that any q-wise independent hash essentially preserves uninvertability.

Theorem 3.1 ([DPW14, Theorem 4.1] (restated))**.** *Let $\left\{\mathsf{h}_S : \{0,1\}^n \to \{0,1\}^m : S \in \{0,1\}^d\right\}$ be a q-wise independent hashing family. For any $D : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}$ and any random variable $X \in \{0,1\}^n$ with min-entropy $\mathbf{H}_\infty(X) \geq k$, if $\Pr[D(U,S) = 1] = \delta$, then $\Pr[D(\mathsf{h}_S(X), S) = 1] \leq O(q2^{m-k}) \max\{\delta, 2^{-q}\}$.*

Throughout the proof, let S, e, x be chosen uniformly at random from their domains.

$$\begin{aligned}
&\Pr_{S,e,x}[\mathcal{A}(S,e,f(x),\mathsf{h}_S(x)) = x \wedge x \in \mathbf{I}_{S,e}] \\
&\leq \Pr_{S,e,x}[\mathcal{A}(S,e,f(x),\mathsf{h}_S(x)) = x \wedge x \in E_e] \\
&\leq \Pr_{S,e,x}\left[\mathcal{A}(S,e,f(x),\mathsf{h}_S(x)) \in f^{-1}(f(x)) \wedge x \in E_e\right] \\
&= \Pr_{\substack{S,e,x \\ x' \leftarrow f^{-1}(f(x))}}\left[\mathcal{A}(S,e,f(x),\mathsf{h}_S(x')) \in f^{-1}(f(x)) \wedge x' \in E_e\right] && (1) \\
&= \Pr_{\substack{S,e,x \\ x' \leftarrow f^{-1}(f(x))}}\left[\mathcal{A}(S,e,f(x),\mathsf{h}_S(x')) \in f^{-1}(f(x)) \wedge x \in E_e\right] && (2) \\
&= \sum_{x^*, e^* \,:\, x^* \in E_{e^*}} \Pr_{x,e}[x = x^*, e = e^*] \Pr_{\substack{S \\ x' \leftarrow f^{-1}(f(x^*))}}\left[\mathcal{A}(S,e,f(x^*),\mathsf{h}_S(x')) \in f^{-1}(f(x^*))\right] \\
&\leq \sum_{x^*, e^* \,:\, x^* \in E_{e^*}} \Pr_{x,e}[x = x^*, e = e^*] O(\lambda) \max\left\{\Pr_{\substack{S \\ U \leftarrow \{0,1\}^{e+1}}}\left[\mathcal{A}(S,e,f(x^*),U) \in f^{-1}(f(x^*))\right], 2^{-\lambda}\right\} && (3) \\
&\leq O(\lambda) \Pr_{\substack{S,e,x \\ U \leftarrow \{0,1\}^{e+1}}}\left[\mathcal{A}(S,e,f(x),U) \in f^{-1}(f(x))\right] + O(\lambda)2^{-\lambda} \\
&\leq \mu(\lambda) && (4)
\end{aligned}$$

Equation (1) follows since we can think of sampling the pair $x, f(x)$ as equivalent to sampling $x', f(x)$ where $x \leftarrow \{0,1\}^\lambda, x' \leftarrow f^{-1}(f(x))$. Equation (2) follows from the definition of E_e, which implies that $x \in E_e$ if and only if $f^{-1}(f(x)) \subseteq E_e$. Equation (3) follows by applying Theorem 3.1 with the variable $x' \leftarrow f^{-1}(f(x))$ having entropy $k = \mathbf{H}_f(x) \geq e - 1$, the hash function h_S having output-length $m = e + 1$ and independence $q = \lambda$. To apply the theorem, we think of a distinguisher $D_{\mathcal{A},e,f(x)}$ that given (z, S) tests whether $\mathcal{A}(S, e, f(x), z)$ inverts $f(x)$. Lastly, by the one-wayness of f, there is some negligible $\mu(\lambda)$ such that equation (4) holds, as we wanted to show. □

Remark 3.1 (super-polynomial security). In the above construction, starting from a one-way function f that is t-secure directly yields t-security of $\mathcal{SIOWF}$.

3.2 Injective OWFs from iO and SIOWFs

We now construct a family of injective one-way functions based on iO and one-way functions. We first construct a weak but (fully) injective one-way function, and then use standard direct product amplification.

Ingridients. Let $i\mathcal{O}$ be an indistinguishability obfuscator for P/poly, and let $\mathcal{PRF}$ be a family of puncturable pseudo-random functions, where for $S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda)$, PRF_S maps $\{0,1\}^\lambda \to \{0,1\}^\lambda$. Let $(\mathsf{COM}_1, \mathsf{COM}_2)$ be a two message statistically-binding commitment scheme, where $\mathsf{COM}_1(1^\lambda)$ samples a first message M_1, and $\mathsf{COM}_2(x, M_1; r)$ computes a commitment M_2 to plaintext $x \in \{0,1\}^\lambda$, with respect to the first message M_1 and random coins $r \in \{0,1\}^\lambda$.

The function family. For $M_1 \leftarrow \mathsf{COM}_1(1^\lambda)$, $S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda)$, consider the circuit $C_{M_1,S} : \{0,1\}^\lambda \to \{0,1\}^*$ defined by

$$C_{M_1,S}(x) := \mathsf{COM}_2(x, M_1; \mathsf{PRF}_S(x)) \ ,$$

padded to some polynomial size $\ell(\lambda)$ to be determined later in the analysis.

The constructed family of one-way functions $\mathcal{OWF}$ consists of all obfuscations of such circuits:

1. A random key $\mathsf{OWF}_K \leftarrow \mathcal{K}_{\mathcal{OWF}}(1^\lambda)$ consists of an obfuscation $\widetilde{C} \leftarrow i\mathcal{O}(C_{M_1,S})$, for a first commitment message $M_1 \leftarrow \mathsf{COM}_1(1^\lambda)$ and PRF seed $S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda)$.
2. The function is given by $\mathsf{OWF}_K(x) = \widetilde{C}(x)$.

The fact that the construction gives an injective family follows directly from the statistical binding of the commitment. We next show that it is also weakly one-way.

Proposition 3.2. *Assume there exists a family $\mathcal{SIOWF}$ of sometimes-injective one-way functions. Then the above construction is a weak one-way function.*

Proof sketch. Let $\mathbf{I}_K$ and $p(\cdot)$ be as in Definition 3.1 such that $\mathcal{SIOWF}$ has an injective sub-domain $\mathbf{I}_K$ of density $1/p(\lambda)$. We show that any poly-size adversary $\mathcal{A}$ fails to invert the constructed $\mathcal{OWF}$ with probability at least $\frac{1}{p(\lambda)} - \mu(\lambda)$ for some negligible $\mu(\cdot)$. For this purpose we consider a sequence of hybrids.

Hyb_1**:** The real experiment. Here $\mathcal{A}$ is given as input $\widetilde{C}, \widetilde{C}(x)$ for a random input $x \leftarrow \{0,1\}^\lambda$ and random key $\widetilde{C} \leftarrow \mathcal{K}_{\mathcal{OWF}}(1^\lambda)$ and tries to obtain x.

Hyb_2**:** Here $\widetilde{C}$ is an obfuscation of an augmented circuit. In the new circuit, the PRF seed S is replaced with $S\{x\}$, which is punctured at x. In addition, $M_2 = \mathsf{COM}_2(x, M_1; \mathsf{PRF}_S(x))$ is hardwired as the output on input x (the input x itself is also hardwired). This circuit computes the same function as the previous $C_{M_1,S}$, thus by the iO guarantee, $\mathcal{A}$ obtains x with the same probability up to a negligible difference.

Hyb_3**:** Here $M_2 = \mathsf{COM}_2(x, M_1; r)$ is generated with truly uniform randomness r, rather than $\mathsf{PRF}_S(x)$. (This includes both the hardwired M_2 as well as the output of the function $\widetilde{C}(x) = M_2$ given to $\mathcal{A}$.) By pseudorandomness at punctured points, the probability of obtaining x is again maintained up to a negligible difference.

Hyb_4**:** Here $M_2 = \mathsf{COM}_2(0^\lambda, M_1; r)$ is a commitment to 0^λ, rather than to x. By the computational hiding of the commitment, the probability of obtaining x is again maintained up to a negligible difference.

Hyb_5**:** Here we unpuncture S. The point x itself is still hardwired into the circuit in the clear. This does not change functionality, and thus the probability that x is obtained is maintained, up to a negligible difference, by iO.

Hyb_6**:** In this hybrid, we also sample a key $K \leftarrow \mathcal{K}_{\mathcal{SIOWF}}(1^\lambda)$ for a sometimes injective one-way function. Then, whenever $x \in \mathbf{I}_K$, instead of storing x in the clear and comparing it to the input (in order to decide whether to return M_2), we store its image $\mathsf{SIOWF}_K(x)$. Comparison of x with an input x' is now done by first computing $\mathsf{SIOWF}(x')$ and then comparing the images. Since $x \in \mathbf{I}_K \subseteq \mathbf{Inj}(\mathsf{SIOWF}_K)$ this does not change functionality and the probability of obtaining x is preserved by iO.

Finally, we note that for some negligible $\mu(\lambda)$,

$$\begin{aligned}
&\Pr\left[\mathcal{A} \text{ obtains } x \text{ in } \mathsf{Hyb}_6\right] \\
&\leq \Pr\left[\mathcal{A} \text{ obtains } x \text{ in } \mathsf{Hyb}_6 \wedge x \in \mathbf{I}_K\right] + \Pr\left[x \notin \mathbf{I}_K\right] \\
&\leq \mu(\lambda) + 1 - \frac{1}{p(\lambda)} \ .
\end{aligned}$$

Above the bound on the first summand follows from one-wayness on the injective sub-domain, indeed, given $K, \mathsf{SIOWF}_K(x)$ such that $x \in \mathbf{I}_K$, the view of $\mathcal{A}$ can be efficiently simulated. The bound on the second summand follows from sometimes injectiveness.

The padding parameter. $\ell(\lambda)$ is chosen to be the maximum size among all circuits we went through in the analysis, so that iO can always be applied. □

4 Trapdoor Permutations from iO

In this section we define Trapdoor Permutations (TDPs) and their enhancements, and construct them from sub-exponentially-secure iO. At large the definitions follow [GR13], with some exceptions discussed below.

4.1 Standard TDPs

We start by defining standard (non-enhanced) TDPs.

Definition 4.1 (TDP)**.** *Let k be polynomially bounded length function. An efficiently computable family of functions*

$$\mathcal{TDP} = \left\{ \mathsf{TDP}_{PK} : D_{PK} \to D_{PK} : PK \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N} \right\} \ ,$$

associated with efficient (probabilistic) key and domain samplers $(\mathcal{K}, \mathcal{S})$, is a (standard) TDP if it satisfies

1. **Trapdoor invertibility:** *For any (PK, SK) in the support of $\mathcal{K}(1^\lambda)$, the function TDP_{PK} is a permutation of a corresponding domain D_{PK}. The inverse $\mathsf{TDP}^{-1}_{PK}(y)$ can be efficiently computed for any $y \in D_{PK}$, using the trapdoor SK.*

2. **Domain sampling:** *$\mathcal{S}(PK)$ samples a pseudo-uniform element in the domain D_{PK}; that is, for any polysize distinguisher $\mathcal{D}$, there exists a negligible $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\left| \begin{array}{c} \Pr\left[\mathcal{D}(r_{\mathcal{K}}, x) = 1 : \begin{array}{c} r_{\mathcal{K}} \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)} \\ (PK, SK) \leftarrow \mathcal{K}(1^\lambda; r_{\mathcal{K}}) \\ x \leftarrow \mathcal{S}(PK) \end{array} \right] \\ - \Pr\left[\mathcal{D}(r_{\mathcal{K}}, x) = 1 : \begin{array}{c} r_{\mathcal{K}} \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)} \\ (PK, SK) \leftarrow \mathcal{K}(1^\lambda; r_{\mathcal{K}}) \\ x \leftarrow D_{PK} \end{array} \right] \end{array} \right| \leq \mu(\lambda) \ .$$

3. **One-wayness:** *For any polysize inverter $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$, such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathcal{A}(PK, \mathsf{TDP}_{PK}(x)) = x : \begin{array}{c}(PK, SK) \leftarrow \mathcal{K}(1^\lambda) \\ x \leftarrow \mathcal{S}(PK)\end{array}\right] \leq \mu(\lambda) \ .$$

The above definition is similar to the one in [GR13] with the exception that $\mathcal{S}(PK)$ in [GR13] is required to sample a domain element that is statistically close to a uniform domain element, whereas we only require computational indistinguishability. Importantly, we require that computational-indistinguishability holds even given the random coins used to generate (PK, SK). This property is required in applications (e.g., the EGL oblivious transfer protocol) and follows automatically (and thus not required explicitly) in the case of statistical-indistinguishability.

Also, we note that like in trapdoor permutations with statistical (rather than computational) domain sampling, the one-wayness requirement can be restated in any of the following equivalent forms:

3.a. The pre-image x is sampled uniformly from the domain:

$$\Pr\left[\mathcal{A}(PK, \mathsf{TDP}_{PK}(x)) = x : \begin{array}{c}(PK, SK) \leftarrow \mathcal{K}(1^\lambda) \\ x \leftarrow D_{PK}\end{array}\right] \leq \mu(\lambda) \ .$$

3.b. The adversary inverts a random domain element x:

$$\Pr\left[\mathcal{A}(PK, x) = \mathsf{TDP}^{-1}_{PK}(x) : \begin{array}{c}(PK, SK) \leftarrow \mathcal{K}(1^\lambda) \\ x \leftarrow D_{PK}\end{array}\right] \leq \mu(\lambda) \ .$$

3.c. The adversary inverts a domain element sampled by $\mathcal{S}(PK)$:

$$\Pr\left[\mathcal{A}(PK, x) = \mathsf{TDP}^{-1}_{PK}(x) : \begin{array}{c}(PK, SK) \leftarrow \mathcal{K}(1^\lambda) \\ x \leftarrow \mathcal{S}(PK)\end{array}\right] \leq \mu(\lambda) \ .$$

4.1.1 The construction.

We now proceed to describe the construction of a TDP. The construction relies on super-polynomial hardness assumptions; for a convenient setting of parameters we assume that the underlying cryptographic primitives are sub-exponentially hard. In Section 4.4, we discuss relaxations to more mild (but still super-polynomial) hardness.

Ingredients. Fix any constant $\varepsilon < 1$, and let $T = T(\lambda) = 2^{\lambda^{\varepsilon/2}}$. We require the following primitives:

- $i\mathcal{O}$ is a $(\lambda, 2^{-\lambda^\varepsilon})$-secure indistinguishability obfuscator for P/poly.
- $\mathcal{PRF}$ is a $(\lambda, 2^{-\lambda^\varepsilon})$-secure family of puncturable pseudo-random functions, which for $\lambda \in \mathbb{N}$ maps $\mathbb{Z}_T$ to $\{0,1\}^\lambda$.
- $\mathcal{OWF}$ is a $(2^{\lambda^\varepsilon}, 2^{-\lambda^\varepsilon})$-secure family of injective one-way functions, which for $\lambda \in \mathbb{N}$ maps $\{0,1\}^\lambda$ to $\{0,1\}^*$. (Will only come up in the analysis, and not in the construction itself.)
- PRG is a (polynomially-secure) length-doubling pseudo-random generator.

The function family. The core of the construction will be obfuscations of circuits $(\mathsf{F}_S, \mathsf{X}_S)$ for computing the function forward and sampling domain elements, respectively. These obfuscations will be embedded in the function key PK and their corresponding secret S will be the trapdoor. The circuits are defined next.
For $S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda)$:

1. $\mathsf{F}_S(i,\sigma)$: takes as input $i \in \mathbb{Z}_T$ and $\sigma \in \{0,1\}^\lambda$ and checks whether $\sigma = \mathsf{PRF}_S(i)$. If so it returns $i+1, \mathsf{PRF}_S(i+1)$, where $i+1$ is computed modulo T. Otherwise it returns $\perp$.

2. $\mathsf{X}_S(s)$: takes as input a seed $s \in \{0,1\}^{\log \sqrt{T}}$ and outputs $(i,\sigma) = (\mathsf{PRG}(s), \mathsf{PRF}_S(\mathsf{PRG}(s)))$, where i is interpreted as a residue in $\mathbb{Z}_T$.

Both circuits are padded so that their total size is $\ell(\lambda)$, for a fixed polynomial $\ell(\cdot)$ specified later.
The constructed family $\mathcal{TDP}$ is now defined as follows.

1. A random key PK consists of obfuscations $\widetilde{\mathsf{F}} \leftarrow i\mathcal{O}(\mathsf{F}_S)$ and $\widetilde{\mathsf{X}} \leftarrow i\mathcal{O}(\mathsf{X}_S)$, for $S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda)$. The corresponding trapdoor SK is S.

2. The domain D_{PK} is $\{(i,\sigma) : i \in \mathbb{Z}_T, \sigma = \mathsf{PRF}_S(i)\}$.

3. To compute $\mathsf{TDP}_{PK}(i,\sigma)$, return $\widetilde{\mathsf{F}}(i,\sigma)$.

4. To compute $\mathsf{TDP}^{-1}_{PK}(i,\sigma)$ given SK, return $(i-1, \mathsf{PRF}_S(i-1))$, where $i-1$ is computed modulo T.

5. The domain sampler $\mathcal{S}(PK;s)$ takes as input PK and randomness $s \in \{0,1\}^{\log \sqrt{T}}$ and outputs $\widetilde{\mathsf{X}}(s)$.

Proposition 4.1. *The above construction of $\mathcal{TDP}$ is a trapdoor permutation.*

Proof. The fact that TDP is trapdoor-invertible follows readily from the construction. The fact that the domain sampler $\mathcal{S}(PK)$ samples domain elements that are computationally-indistinguishable from uniform domain elements, even given the coins of $\mathcal{K}$ used to generate (PK, SK), follows directly from the pseudo-randomness guarantee of PRG.

From hereon, we focus on showing one-wayness. It would be simplest to work with the formulation (3.b) of the one-wayness requirement. Concretely, fix any polysize $\mathcal{A}$, we show that there exists a negligible $\mu(\cdot)$ such that for every $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{PRF}_S(i-1) \leftarrow \mathcal{A}(\widetilde{\mathsf{F}}, \widetilde{\mathsf{X}}, i, \mathsf{PRF}_S(i)) : \begin{array}{r} S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda) \\ \widetilde{\mathsf{F}} \leftarrow i\mathcal{O}(\mathsf{F}_S) \\ \widetilde{\mathsf{X}} \leftarrow i\mathcal{O}(\mathsf{X}_S) \\ i \leftarrow \mathbb{Z}_T \end{array}\right] \leq \mu(\lambda) \ .$$

We show that except with sub-exponentially-small probability $\mathcal{A}(\widetilde{\mathsf{F}}, \widetilde{\mathsf{X}}, i, \mathsf{PRF}_S(i))$ cannot output σ^* such that $\widetilde{\mathsf{F}}(i-1,\sigma^*) \neq \perp$, which is equivalent to showing that $\sigma^* \neq \mathsf{PRF}_S(i-1)$. We prove this via a sequence of indistinguishable hybrid experiments where the obfuscated $\widetilde{\mathsf{F}}$ is gradually augmented to return $\perp$ on an increasing interval, until it eventually returns $\perp$ on some interval $[i-u, i-1]$ (for every possible signature), meaning in particular that $\mathcal{A}(\widetilde{\mathsf{F}}, \widetilde{\mathsf{X}}, i, \mathsf{PRF}_S(i))$ cannot find an accepting signature σ^* for $i-1$. Throughout the hybrids we change the obfuscated circuits and assume that they are always padded so that their total size is $\ell(\lambda)$, for a fixed polynomial $\ell(\cdot)$ specified later.

Hyb_1**:** The original experiment.

Hyb_2**:** Here $\widetilde{\mathsf{F}}$ is an obfuscation of a circuit $\mathsf{F}^{(2)}_{i,v,S,K'}$. The circuit has hardwired a key $K' \leftarrow \mathcal{K}_{\mathcal{OWF}}(1^{\lambda'})$ for and injective OWF defined on inputs of length $\lambda' = \log \sqrt[4]{T}$, and a random image $v = \mathsf{OWF}_{K'}(u)$, for $u \leftarrow \{0,1\}^{\lambda'} \cong \mathbb{Z}_{\sqrt[4]{T}}$. The circuit behaves like F, with the exception that given any input (k,σ) such that $k \in [i - \sqrt[4]{T}, i-1]$ and $\mathsf{OWF}_{K'}(i-k) = v$, the circuit returns $\perp$.

$\mathsf{Hyb}_{3,j}, j \in [0, \sqrt[4]{T}-1]$**:** Here $\widetilde{\mathsf{F}}$ is an obfuscation of a circuit $\mathsf{F}^{(3,j)}_{i,u,S}$. The circuit has a random index $u \leftarrow \mathbb{Z}_{\sqrt[4]{T}}$. On any input (k,σ), it returns $\perp$ if $k \in [i-u, i-u+j]$, where we truncate j so that $j = \min\{j, u-1\}$. On any other input it behaves just like F_S.

$\mathsf{Hyb}_{4,j}, j \in [0, \sqrt[4]{T}-1]$**:** Here $\widetilde{\mathsf{F}}$ is an obfuscation of a circuit $\mathsf{F}^{(4,j)}_{i,u,S\{i-u+j\},\sigma_{i-u+j}}$. The circuit is the same as $\mathsf{F}^{(3,j)}_{i,u,S}$, only that it has a punctured PRF key $S\{i-u+j\}$, and the value $\sigma_{i-u+j} = \mathsf{PRF}_S(i-u+j)$ is hardwired. In addition, $\widetilde{\mathsf{X}}$ is an obfuscation of a circuit $\mathsf{X}^{(4,j)}_{S\{i-u+j\}}$. The circuit is the same as X_S, only that it has the punctured $S\{i-u+j\}$, and whenever $\mathsf{PRF}_S(i-u+j)$ is required the circuit returns $\bot$ (no value is hardwired instead).

$\mathsf{Hyb}_{5,j}, j \in [0, \sqrt[4]{T}-1]$**:** Here $\widetilde{\mathsf{F}}$ is an obfuscation of a circuit $\mathsf{F}^{(5,j)}_{i,u,S\{i-u+j\},\sigma_{i-u+j}}$. The circuit is the same as $\mathsf{F}^{(4,j)}_{i,u,S\{i-u+j\},\sigma_{i-u+j}}$, only that the hardwired σ_{i-u+j} is not set to $\mathsf{PRF}_S(i-u+j)$, but sampled uniformly at random from $\{0,1\}^\lambda$.

$\mathsf{Hyb}_{6,j}, j \in [0, \sqrt[4]{T}-1]$**:** Here $\widetilde{\mathsf{F}}$ is an obfuscation of a circuit $\mathsf{F}^{(6,j)}_{i,u,S,v,K}$. The circuit is the same as $\mathsf{F}^{(5,j)}_{i,u,S\{i-u+j\},\sigma_{i-u+j}}$, only that instead of storing σ_{i-u+j} in the clear $v = \mathsf{OWF}_K(\sigma_{i-u+j})$ is stored, and comparison of σ and σ_{i-u+j} is done by comparing $\mathsf{OWF}_K(\sigma)$ and $\mathsf{OWF}_K(\sigma_{i-u+j})$. Here $K \leftarrow \mathcal{K}_{\mathcal{OWF}}(1^\lambda)$ is a key for an injective OWF from the family $\mathcal{OWF}$. Also, the PRF seed S is no longer punctured. In addition, $\widetilde{\mathsf{X}}$ is again an obfuscation of X_S (where S is no longer punctured).

We prove the following:

Claim 4.1. *For any polysize distinguisher $\mathcal{D}$, all $\lambda \in \mathbb{N}$, and all $j \in [0, \sqrt[4]{T(\lambda)}-1]$:*

1. $|\Pr[\mathcal{D}(\mathsf{Hyb}_1)=1] - \Pr[\mathcal{D}(\mathsf{Hyb}_2)=1]| \leq 2^{-\Omega(\lambda^{\varepsilon^2})}$,
2. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_2)=1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,0})=1]\right| \leq 2^{-\Omega(\lambda^{\varepsilon})}$,
3. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{3,j})=1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{4,j})=1]\right| \leq T^{-1/2} + 2^{-\Omega(\lambda^{\varepsilon})}$,
4. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{4,j})=1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{5,j})=1]\right| \leq 2^{-\Omega(\lambda^{\varepsilon})}$,
5. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{5,j})=1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{6,j})=1]\right| \leq T^{-1/2} + 2^{-\Omega(\lambda^{\varepsilon})}$,
6. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{6,j})=1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,j+1})=1]\right| \leq 2^{-\Omega(\lambda^{\varepsilon})}$,

where the view of $\mathcal{D}$ in each hybrid consists of the corresponding obfuscated $\widetilde{\mathsf{F}}$, $\widetilde{\mathsf{X}}$ and $(i, \mathsf{PRF}_S(i))$.

Proving the above claim will conclude the proof of Proposition 4.1 since it implies that

$$\Pr\left[\begin{array}{l} \sigma \leftarrow \mathcal{A}(\widetilde{\mathsf{F}}, \widetilde{\mathsf{X}}, i, \mathsf{PRF}_S(i)) \\ \mathsf{F}(i-1,\sigma) \neq \bot \end{array} : \begin{array}{r} S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda) \\ \widetilde{\mathsf{F}} \leftarrow i\mathcal{O}(\mathsf{F}_S) \\ \widetilde{\mathsf{X}} \leftarrow i\mathcal{O}(\mathsf{X}_S) \\ i \leftarrow \mathbb{Z}_T \end{array}\right] \leq$$

$$\Pr\left[\begin{array}{l} \sigma \leftarrow \mathcal{A}(\widetilde{\mathsf{F}}, \widetilde{\mathsf{X}}, i, \mathsf{PRF}_S(i)) \\ \mathsf{F}(i-1,\sigma) \neq \bot \end{array} : \begin{array}{r} S \leftarrow \mathcal{K}_{\mathcal{PRF}}(1^\lambda) \\ \widetilde{\mathsf{F}} \leftarrow \boxed{i\mathcal{O}(\mathsf{F}^{(3,\sqrt[4]{T})}_{i,S,u})} \\ \widetilde{\mathsf{X}} \leftarrow i\mathcal{O}(\mathsf{X}_S) \\ i \leftarrow \mathbb{Z}_T \end{array}\right]$$

$$+ \lambda^{-\omega(1)} + 2^{-\Omega(\lambda^{\varepsilon^2})} + \sqrt[4]{T} \cdot (T^{-1/2} + 2^{-\Omega(\lambda^{\varepsilon})}) =$$

$$0 + \lambda^{-\omega(1)} + 2^{-\Omega(\lambda^{\varepsilon^2})} + 2^{\lambda^{\frac{\varepsilon}{2}}/4} \cdot (2^{-\lambda^{\frac{\varepsilon}{2}}/2} + 2^{-\Omega(\lambda^{\varepsilon})}) =$$

$$\lambda^{-\omega(1)} \ ,$$

where the first to last equality follows from the fact that $\mathsf{F}^{(3,\sqrt[4]{T})}_{S,u}(i-1,\sigma) = \bot$ for any σ.

Proof of Claim 4.1. We prove each of the items in the claim. The proof is at most part similar to the one in [BPR15], with several exceptions.

Proof of 1 and 6. Recall that here we need to show that

1. $|\Pr[\mathcal{D}(\mathsf{Hyb}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1]| \leq 2^{-\Omega(\lambda^{\varepsilon^2})}$,
6. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{6,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,j+1}) = 1]\right| \leq 2^{-\Omega(\lambda^{\varepsilon})}$.

In both cases, one obfuscated program differs from the other on exactly a single point, which is the unique (random) preimage of the corresponding image v (in the first case $v = \mathsf{OWF}_{K'}(u)$, and in the second $v = \mathsf{OWF}_K(\sigma_{i-u+j})$).

To prove the claim, we rely on a lemma proven in [BCP14] that roughly shows that, for circuits that only differ on a single input, iO implies what is known as *differing input obfuscation* [BGI+01], where it is possible to efficiently extract from any iO distinguisher an input on which the underlying circuits differ.

Lemma 4.1 (special case of [BCP14])**.** *Let $i\mathcal{O}$ be a (t, δ)-secure indistinguishability obfuscator for P/poly. There exists a PPT oracle-aided extractor $\mathcal{E}$, such that for any $t^{O(1)}$-size distinguisher $\mathcal{D}$, and two equal size circuits C_0, C_1 differing on exactly one input x^*, the following holds. Let C'_0, C'_1 be padded versions of C_0, C_1 of size $s \geq 3 \cdot |C_0|$.*

$$\begin{aligned} &\textit{If} && |\Pr[\mathcal{D}(i\mathcal{O}(C'_0) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C'_1) = 1]| = \eta \geq \delta(s)^{o(1)} \ , \\ &\textit{then} && \Pr\left[x^* \leftarrow \mathcal{E}^{\mathcal{D}(\cdot)}(1^{1/\eta}, C_0, C_1)\right] \geq 1 - 2^{-\Omega(s)} \ . \end{aligned}$$

Using the lemma, we show that if either item 2 or 7 do not hold, we can invoke the distinguisher $\mathcal{D}$ to invert the underlying one-way function. The argument is similar in both cases up to different parameters; for concreteness, we focus on the first.

Assume that for infinitely many $\lambda \in \mathbb{N}$, $\mathcal{D}$ distinguishes Hyb_0 from Hyb_1 with gap $\eta(\lambda) = 2^{-o(\lambda^{\varepsilon^2})}$. Then, by averaging, with probability $\eta(\lambda)/2$ over the choice of (u, K'), $\mathcal{D}$ distinguishes the two distributions conditioned on these choices with gap $\eta(\lambda)/2$. Thus, we can invoke the extractor $\mathcal{E}$ given by Lemma 4.1 to invert the one-way function family $\mathcal{OWF}$ with probability $\frac{\eta(\lambda)}{2} \cdot (1 - 2^{-\Omega(\lambda)}) \geq 2^{-o(\lambda^{\varepsilon^2})}$ in time $t_{\mathcal{E}}(\lambda) \cdot t_{\mathcal{D}}(\lambda) \leq \eta(\lambda)^{-O(1)} \cdot \lambda^{O(1)} = 2^{O(\lambda^{\varepsilon^2})}$. Note that, indeed, given the image and the one-way function key, the inverter can construct the corresponding circuits efficiently. Recall that OWF'_K is defined on inputs of size $\lambda' = \log \sqrt[4]{T} = \lambda^{\varepsilon/2}/4$, and is $(2^{-\lambda'^{\varepsilon}}, 2^{\lambda'^{\varepsilon}})$-secure. Thus we get a contradiction to its one-wayness.

Proof of 2. Recall that here we need to show that

2. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,0}) = 1]\right| \leq 2^{-\Omega(\lambda^{\varepsilon})}$.

Here the obfuscated $\widetilde{\mathsf{F}}$ compute the exact same function in both hybrids. Specifically, for any input (k, σ), a comparison in the clear of $i - k$ and u is replaced by comparison of their corresponding values $\mathsf{OWF}_{K'}(i - k)$ and $\mathsf{OWF}_{K'}(u)$ under an injective one-way function. Thus, the required indistinguishability follows from the $(\lambda, 2^{-\lambda^{\varepsilon}})$-security of $i\mathcal{O}$.

Proof of 3. Recall that here we need to show that

3. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{3,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{4,j}) = 1]\right| \leq T^{-1/2} + 2^{-\Omega(\lambda^{\varepsilon})}$.

Here also, the obfuscated $\widetilde{\mathsf{F}}$ computes the exact same function in both hybrids. Specifically, rather than computing $\sigma_{i-u+j} = \mathsf{PRF}_S(i - u + j)$ using the PRF key S, the value σ_{i-u+j} is hardwired and directly compared to σ. For any other index, the punctured key $S\{i - u + j\}$ is used.

We now claim that the obfuscated $\widetilde{\mathsf{X}}$ also computes the same function in both hybrids with overwhelming probability $1 - T^{-1/2}$. Indeed, since X_S only computes PRF_S on values in the image of PRG, the probability that X_S and $\mathsf{X}^{(4,j)}_{S\{i-u+j\}}$ do not compute the same function can be bounded by the probability that $i - u + j$ is not in the image of PRG. Recall that i is sampled uniformly from $\mathbb{Z}_T$; thus, $i - u + j$ is also uniformly random in $\mathbb{Z}_T$, and we can bound the probability that it is in the image of $\mathsf{PRG} : \mathbb{Z}_{\sqrt{T}} \to \mathbb{Z}_T$ by $\sqrt{T} \cdot T^{-1} = T^{-1/2}$.

The required indistinguishability now follows from iO security.

Proof of 4. Recall that here we need to show that

4. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{4,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{5,j}) = 1]\right| \leq 2^{-\Omega(\lambda^\varepsilon)}$.

The only difference between the two obfuscated circuit distributions is that in the first the hardwired value σ_{i-u+j} in $\widetilde{\mathsf{F}}$ is $\mathsf{PRF}_S(i - u + j)$, whereas in the second it is sampled independently uniformly at random. Indistinguishability follows from the $(2^{\lambda^\varepsilon}, 2^{-\lambda^\varepsilon})$-pseudo-randomness at the punctured point guarantee. Note that, indeed, given punctured key $S\{i - u + j\}$ and σ_{i-u+j}, a distinguisher can construct the corresponding circuits $\widetilde{\mathsf{F}}, \widetilde{\mathsf{X}}$ efficiently.

Proof of 5. Recall that here we need to show that

5. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{5,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{6,j}) = 1]\right| \leq T^{-1/2} + 2^{-\Omega(\lambda^\varepsilon)}$.

Here also, the two obfuscated $\widetilde{\mathsf{F}}$ in both hybrids compute the exact same function. First, the comparison of σ and σ_{i-u+j} is replaced by comparison of their corresponding values under an injective one-way function. In addition, the punctured key $S\{i - u + j\}$ is replaced with a non-punctured key S. This does not affect functionality as the two keys compute the same function on all points except $i - u + j$, and the circuit in the two hybrids treats any input $i - u + j, \sigma$, independently of the PRF key.

Also, $\widetilde{\mathsf{X}}$ now obfuscates the unpunctured version X_S instead of $\mathsf{X}^{(4,j)}_{S\{i-u+j\}}$. As before this does not change functionality with overwhelming probability $1 - T^{-1/2}$.

Overall, the required indistinguishability follows from iO.

This concludes the proof of the Claim 4.1 and Proposition 4.1. □

The padding parameter $\ell(\lambda)$. We choose $\ell(\lambda)$ so that each of the circuits $\widetilde{\mathsf{F}}^{\cdots}_{\cdots}$ considered above can be implemented by a circuit of size at most $\ell(\lambda)/3$. (The extra $1/3$ slack is taken to satisfy Lemma 4.1 in the analysis below.)

□

4.2 Enhanced TDPs

We next define enhanced TDPs. These are basically (standard) TDPs where it is possible to obliviously sample hard-to-invert images; concretely, given $x \leftarrow \mathcal{S}(PK; r_{\mathcal{S}})$, it is hard to find $\mathsf{TDP}^{-1}_{PK}(x)$, even given the coins $r_{\mathcal{S}}$ of $\mathcal{S}$.

Definition 4.2 (Enhanced TDP)**.** *A TDP family $\mathcal{TDP}$ is said to be enhanced if for any polysize inverter $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$, such that for all $\lambda \in \mathbb{N}$,*

$$\Pr\left[\mathcal{A}(PK, r_{\mathcal{S}}) = \mathsf{TDP}^{-1}_{PK}(x) : \begin{array}{c}(PK, SK) \leftarrow \mathcal{K}(1^\lambda) \\ r_{\mathcal{S}} \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)} \\ x \leftarrow \mathcal{S}(PK; r_{\mathcal{S}})\end{array}\right] \leq \mu(\lambda) \ .$$

4.2.1 Enhancing the previous construction.

We now describe how to enhance the construction presented in the previous section.

Is the previous TDP already enhanced? We start by noting that the family $\mathcal{TDP}$ constructed in the previous section may not be enhanced. Specifically, recall that the randomness $r_{\mathcal{S}}$ used by $\mathcal{S}$ in the construction is a seed s for PRG which is extended to an index i, and the corresponding domain element is $(i, \mathsf{PRF}_S(i))$. Note that it may very well be that given the seed s such that $i = \mathsf{PRG}(s)$, it is not hard to find another seed s' such that $\mathsf{PRG}(s') = i - 1$. In this case, the inverter may invoke the sampler $\mathcal{S}$ with this randomness and invert $(i, \mathsf{PRF}_S(i))$.

Looking more closely into the analysis, in the previous section, we could replace i with a truly random index, which with high-probability had no images of PRG in its close surrounding, due to the sparseness of PRG's image. This no longer works, as given the seed s used to generate i, we can no longer replace it with a truly random index.

Discrete-image PRGs. To circumvent the above, we rely on a pseudo-random generator *with discrete image*, meaning that with overwhelming probability over the choice of the seed s, the corresponding image $\mathsf{PRG}(s)$ has no other image $\mathsf{PRG}(s')$ in its close surrounding. We show how to construct such pseudo-random generators from plain pseudo-random generators. More accurately, we construct a family of pseudo-random generators indexed by some public seed h, where the discrete image requirement holds with overwhelming probability for a random seed h.

Definition 4.3 (Discrete-image PRG)**.** *Let k and ℓ be polynomially bounded length functions. An efficiently computable family of functions*

$$\mathcal{PRG} = \left\{ \mathsf{PRG}_h : \{0,1\}^{\lambda} \to \{0,1\}^{\ell(\lambda)} : h \in \{0,1\}^{k(\lambda)}, \lambda \in \mathbb{N} \right\} \ ,$$

associated with an efficient (probabilistic) key sampler $\mathcal{K}_{\mathcal{PRG}}$, is a discrete-image PRG if it satisfies:

1. **Pseudo-randomness:** *For any polysize distinguisher $\mathcal{D}$ there is a negligible μ such that for any $\lambda \in \mathbb{N}$:*

$$\left| \begin{array}{c} \Pr\left[\mathcal{D}(h, \mathsf{PRG}_h(s)) = 1 : \begin{array}{c} h \leftarrow \mathcal{K}_{\mathcal{PRG}}(1^{\lambda}) \\ s \leftarrow \{0,1\}^{\lambda} \end{array} \right] \\ - \Pr\left[\mathcal{D}(h, u) = 1 : \begin{array}{c} h \leftarrow \mathcal{K}_{\mathcal{PRG}}(1^{\lambda}) \\ u \leftarrow \{0,1\}^{\ell(\lambda)} \end{array} \right] \end{array} \right| \leq \mu(\lambda) \ .$$

2. **Discrete image:** *for any $\lambda \in \mathbb{N}$ and any $t \in \mathbb{Z}_{2^{\ell(\lambda)}} \setminus \{0\}$:*

$$\Pr\left[\exists s' \neq s : \mathsf{PRG}_h(s) - \mathsf{PRG}_h(s') = t \bmod 2^{\ell(\lambda)} : \begin{array}{c} h \leftarrow \mathcal{K}_{\mathcal{PRG}}(1^{\lambda}) \\ s \leftarrow \{0,1\}^{\lambda} \end{array} \right] \leq 2^{-\ell(\lambda)+\lambda} \ .$$

A construction of discrete-image PRGs. Let $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{\ell(\lambda)}$ be a (plain) pseudo-random generator, and let

$$\mathcal{H}_{\lambda} = \left\{ h : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell(\lambda)} : \lambda \in \mathbb{N} \right\} \ ,$$

be a family of pair-wise independent permutations. We construct a discrete-image family

$$\mathcal{PRG} = \left\{ \mathsf{PRG}_h : \{0,1\}^{\lambda} \to \{0,1\}^{\ell(\lambda)} \right\} \ ,$$

as follows.

- The public seed h is a random hash in the family $\mathcal{H}_{\lambda}$.

- The generator is given by

$$\mathsf{PRG}_h(s) := h(\mathsf{PRG}(s)) \ .$$

Claim 4.2. *$\mathcal{PRG}$ is a discrete-image pseudo-random generator.*

Proof. The pseudo-randomness property follows directly from the fact that PRG is a pseudo-random generator and h is an efficiently computable permutation.

To prove discrete-image, it suffices to show that for any fixed $s \in \{0,1\}^\lambda$ and any $t \in \mathbb{Z}_{2^{\ell(\lambda)}} \setminus \{0\}$,

$$\Pr\left[\exists s' \neq s : \mathsf{PRG}_h(s) - \mathsf{PRG}_h(s') = t \bmod 2^{\ell(\lambda)} : h \leftarrow \mathcal{K}_{\mathcal{PRG}}(1^\lambda)\right] \leq 2^{-\ell(\lambda)+\lambda} \ .$$

Indeed, by pairwise-independence, conditioning on the value of $\mathsf{PRG}_h(s) = h(\mathsf{PRG}(s))$, for every $s' \in \{0,1\}^\lambda$ such that $\mathsf{PRG}(s') \neq \mathsf{PRG}(s)$, the value $h(\mathsf{PRG}(s'))$ is uniformly random in $\mathbb{Z}_{2^{\ell(\lambda)}}$ and thus $h(\mathsf{PRG}(s')) = h(\mathsf{PRG}(s)) + t \bmod 2^{\ell(\lambda)}$ with probability at most $2^{-\ell(\lambda)}$. Taking union-bound over all $s' \in \{0,1\}^\lambda$, the claim follows. □

The augmented construction. The construction of enhanced TDPs is now identical to the one in Section 4.1, except that we augment the obfuscated domain sampling circuit X_S to a circuit $\mathsf{X}_{S,h}$ that also has hardwired a random public seed h for a discrete-image PRG. The new sampling circuit is now defined as the previous ones, except that instead of using a plain $\mathsf{PRG} : \mathbb{Z}_{\sqrt{T}} \to \mathbb{Z}_T$ we use the discrete image $\mathsf{PRG}_h : \mathbb{Z}_{\sqrt{T}} \to \mathbb{Z}_T$.

Proposition 4.2. *The augmented construction is an enhanced trapdoor permutation.*

Proof sketch. The proof is identical to that of Proposition 4.1 with two exceptions to the proof of one-wayness. Whereas in Proposition 4.1, we consider, in Hyb_1 an adversary that tries to invert $(i, \mathsf{PRF}_S(i))$ for a truly uniform $i \leftarrow \mathbb{Z}_T$. Now, $i \leftarrow \mathsf{PRG}_h(s) \in \mathbb{Z}_T$ is a pseudo-random element, and the adversary also obtains the seed s, which are the coins of the sampler $\mathcal{S}(PK)$.

The second difference is when switching between $\mathsf{X}_{S,h}$ and $\mathsf{X}_{S\{i-u+j\},h}$ (in the proofs of items 3 and 5). In Proposition 4.1, we relied on the fact that i is uniformly random and thus $i - u + j \mod T$ is not in the image of PRG with probability $T^{-1/2}$, implying that puncturing does not affect functionality and letting us invoke the iO guarantee. Now i is no longer random, but the same holds based on the discrete image property of PRG_h (when choosing $t = u - j \mod T$). □

4.3 Doubly Enhanced TDPs

We now define doubly-enhanced TDPs. These are enhanced TDPs where given the key PK, it is possible to sample coins $r_{\mathcal{S}}$ together with a preimage x of $y = \mathcal{S}(PK, r_{\mathcal{S}})$. In [GR13], it is required that $r_{\mathcal{S}}$ is distributed as uniformly random coins for $\mathcal{S}$. We relax this requiring that $r_{\mathcal{S}}$ is only pseudo-random even given the randomness used to sample (PK, SK). Indeed, this relaxation suffices for applications of doubly-enhanced TDPs such as non-interactive zero-knowledge.

Definition 4.4 (Doubly-enhanced TDP)**.** *An enhanced TDP family $\mathcal{TDP}$ is said to be doubly-enhanced there exists a sampler $\mathcal{R}$ satisfying the following two requirements.*

1. **Correlated preimage sampling.** *For any PK in the support of $\mathcal{K}(1^\lambda)$:*

$$(x, r_{\mathcal{S}}) \leftarrow \mathcal{R}(PK) \text{ such that } \mathsf{TDP}_{PK}(x) = \mathcal{S}(PK, r_{\mathcal{S}}) \ .$$

2. **Pseudorandomness.** *For any polysize distinguisher $\mathcal{D}$ there is a negligible μ such that for any $\lambda \in \mathbb{N}$:*

$$\left| \begin{array}{l} \Pr\left[\mathcal{D}(x, r_{\mathcal{S}}, r_{\mathcal{K}}) = 1 : \begin{array}{c} PK \leftarrow \mathcal{K}(1^\lambda, r_{\mathcal{K}}) \\ (x, r_{\mathcal{S}}) \leftarrow \mathcal{R}(PK) \end{array}\right] \\ - \Pr\left[\mathcal{D}(x, r_{\mathcal{S}}, r_{\mathcal{K}}) = 1 : \begin{array}{c} PK \leftarrow \mathcal{K}(1^\lambda; r_{\mathcal{K}}) \\ r_{\mathcal{S}} \leftarrow \{0,1\}^{\mathrm{poly}(\lambda)} \\ y \leftarrow \mathcal{S}(PK, r_{\mathcal{S}}) \\ x \leftarrow \mathsf{TDP}_{PK}^{-1}(y) \end{array}\right] \end{array} \right| \leq \mu(\lambda) \ .$$

4.3.1 Doubly enhancing the previous construction.

To make the previous construction doubly enhanced we show how to slightly augment the discrete-image PRG used in the construction on some sparse subset of seeds (thus not hurting previous properties), while taking advantage of the particular structure of our TDP.

Concretely, we augment the code of PRG_h to compute a new PRG_h^* as follows. Let $\mathsf{PRG}' : \mathbb{Z}_{\sqrt[4]{T}} \to \mathbb{Z}_{\sqrt{T}}$ be a length doubling pseudorandom generator that expands small seeds $s' \in \mathbb{Z}_{\sqrt[4]{T}}$ to longer seeds $s \in \mathbb{Z}_{\sqrt{T}}$ for PRG_h. PRG_h^* acts as follows:

Given a (private) seed $s \in \mathbb{Z}_{\sqrt{T}}$ as input, parse it as $(s', r') \in \mathbb{Z}_{\sqrt[4]{T}} \times \mathbb{Z}_{\sqrt[4]{T}}$.

1. If $r' = 0$, compute $\mathsf{PRG}'(s')$, and output $\mathsf{PRG}_h^*(s', r') := \mathsf{PRG}_h(\mathsf{PRG}'(s')) - 1 \mod T$.
2. Otherwise, output as before $\mathsf{PRG}_h^*(s', r') = \mathsf{PRG}_h(s', r')$.

The augmented construction. The construction of doubly-enhanced TDPs is now identical to the one of enhanced TDPs, except that we instantiate the pseudo-random generator with the new $\mathcal{PRG}^* = \{\mathsf{PRG}_h^*\}$.

Proposition 4.3. *The augmented construction is a doubly-enhanced trapdoor permutation.*

Proof sketch. First notice that we did not harm the pseudo-randomness and discrete-image properties of the original family $\mathcal{PRG}$. Indeed, the augmented PRG_h^* only behaves differently from PRG_h on the set $\{s = (s', r') : r' = 0\}$, which has negligible density $T^{-1/4}$. The pseudo-randomness and discrete-image properties, however, are defined for a uniformly random $(s', r') \in \mathbb{Z}_{\sqrt[4]{T}} \times \mathbb{Z}_{\sqrt[4]{T}}$, and thus remain unaffected.

We can now define the sampler $\mathcal{R}(PK)$:

1. Pick a random (short) seed $s' \leftarrow \mathbb{Z}_{\sqrt[4]{T}}$.
2. Compute $r_{\mathcal{S}} = \mathsf{PRG}'(s') \in \mathbb{Z}_{\sqrt[4]{T}} \times \mathbb{Z}_{\sqrt[4]{T}}$ and $r_{\mathcal{S}}^x = (s', 0) \in \mathbb{Z}_{\sqrt[4]{T}} \times \mathbb{Z}_{\sqrt[4]{T}}$.
3. Return $(x, r_{\mathcal{S}})$ where $x = \mathcal{S}(PK; r_{\mathcal{S}}^x)$.

The pseudorandomness of $r_{\mathcal{S}}$, conditioned on $(r_{\mathcal{K}}, x)$, follows directly from the pseudo-randomness guarantee of PRG'. We now note that x is the preimage of $y = \mathcal{S}(PK, r_{\mathcal{S}})$. We shall assume for simplicity that $\mathsf{PRG}'(s')$ never outputs $s = (s''; r'')$ such that $r'' = 0$ (PRG' can always be augmented to satisfy this property). Then, by construction $x = (i - 1, \mathsf{PRF}_S(i - 1))$ where $i = \mathsf{PRG}_h(\mathsf{PRG}'(s'))$ and $y = (i, \mathsf{PRF}_S(i))$.

This completes the proof. □

4.4 Relaxing Subexponential Security

In all constructions above, we assumed all cryptographic primitives are sub-exponentially hard. We now explain how this can be relaxed, and what are the tradeoffs between the hardness of the different primitives. Let $f(\cdot), g(\cdot), h(\cdot)$ be sub-linear functions and assume that $\mathcal{OWF}$ is $(2^{f(\lambda)}, 2^{-f(\lambda)})$-secure, $\mathcal{PRF}$ is $(\lambda, 2^{-g(\lambda)})$-secure, and $i\mathcal{O}$ is $(\lambda, 2^{-h(\lambda)})$-secure. We can restate Claim 4.1 as follows.

Claim 4.3 (Claim 4.1 generalized)**.** *For any polysize distinguisher $\mathcal{D}$, all $\lambda \in \mathbb{N}$, and all $j \in [0, \sqrt[4]{T}]$:*

1. $|\Pr[\mathcal{D}(\mathsf{Hyb}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1]| \leq 2^{-\Omega(f(\log \sqrt[4]{T}))} + 2^{-\Omega(h(\lambda))}$,
2. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,0}) = 1]\right| \leq 2^{-\Omega(h(\lambda))}$,
3. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{3,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{4,j}) = 1]\right| \leq T^{-1/2} + 2^{-\Omega(h(\lambda))}$,
4. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{4,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{5,j}) = 1]\right| \leq 2^{-\Omega(g(\lambda))}$,
5. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{5,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{6,j}) = 1]\right| \leq T^{-1/2} + 2^{-\Omega(h(\lambda))}$,
6. $\left|\Pr[\mathcal{D}(\mathsf{Hyb}_{6,j}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,j+1}) = 1]\right| \leq 2^{-\Omega(f(\lambda))} + 2^{-\Omega(h(\lambda))}$.

The overall inversion probability can be bounded by

$$2^{-\Omega(f(\log \sqrt[4]{T}))} + \sqrt[4]{T} \cdot (T^{-1/2} + 2^{-\Omega(f(\lambda))} + 2^{-\Omega(g(\lambda))} + 2^{-\Omega(h(\lambda))}) \ .$$

In particular, letting $m(\lambda) = \min\{f(\lambda), g(\lambda), h(\lambda)\}$, we can guarantee hardness of the resulting TDP as long as

1. $T(\lambda) = \lambda^{-\omega(1)}$.
2. $m(\lambda) = \omega(\log(T))$.
3. $f(\log \sqrt[4]{T}) = \omega(\log \lambda)$.

For instance, for any constant $\varepsilon < 1$, we can set

- $T = 2^{(\log \lambda)^{2/\varepsilon}}$,
- $f(\lambda) = \lambda^{\varepsilon}$ ($\mathcal{OWF}$ is still sub-exponential),
- $g(\lambda) = h(\lambda) = (\log \lambda)^{2+2/\varepsilon}$ ($\mathcal{PRF}$ and $i\mathcal{O}$ are quasi-polynomial).

Alternatively, we can set

- $T = 2^{2^{(\log \lambda)^{\varepsilon}}}$,
- $f(\lambda) = g(\lambda) = h(\lambda) = 2^{(\log \lambda)^{\frac{1+\varepsilon}{2}}}$ (all primitives are only $2^{\lambda^{o(1)}}$-secure).

Acknowledgements

We thank Mark Zhandry for bringing to our attention the question of injective OWFs from indistinguishability obfuscation. We thank Zhe Zhang and John Steinberger for pointing out a gap in the proof of Proposition 3.1 in a previous version of this work.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *TCC*, 2015.

[AC08] Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 793–802, 2008.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Crypto*, 2015.

[BCC+14] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 71–89, 2014.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 278–291, 1993.

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

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

[BGK+13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. Cryptology ePrint Archive, Report 2013/631, 2013. `http://eprint.iacr.org/`.

[Blu81] Manuel Blum. Coin flipping by telephone. In *Proceedings of the 18th Annual International Cryptology Conference*, pages 11–15, 1981.

[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/`.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *FOCS*, 2015.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 1–25, 2014.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT (2)*, pages 280–300, 2013.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *Crypto*, 2015.

[CLTV14] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. Cryptology ePrint Archive, Report 2014/882, 2014. `http://eprint.iacr.org/`.

[DPW14] Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 93–110, 2014.

[Gen14] Craig Gentry. Computing on the edge of chaos: Structure and randomness in encrypted computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:106, 2014.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.

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

[GK88] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof for a problem equivalent to discrete logarithm. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 57–70, 1988.

[GLSW14] Craig Gentry, Allison B. Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. *IACR Cryptology ePrint Archive*, 2014:309, 2014.

[Gol11] Oded Goldreich. Candidate one-way functions based on expander graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 76–87. 2011.

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

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[JP00] Ari Juels and Marcus Peinado. Hiding cliques for cryptographic security. *Des. Codes Cryptography*, 20(3):269–280, 2000.

[KLW14] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014. `http://eprint.iacr.org/`.

[KMN+14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. *IACR Cryptology ePrint Archive*, 2014:347, 2014.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM Conference on Computer and Communications Security*, pages 669–684, 2013.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NR02] Moni Naor and Omer Reingold. Constructing pseudo-random permutations with a prescribed structure. *J. Cryptology*, 15(2):97–102, 2002.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 187–196, 2008.

[Rab79] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report LCR/TR-212, MIT Laboratory of Computer Science, 1979.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Eurocrypt*, 2015.

A Overview of the Proof in [BPR15]

As explained in the introduction, the proof that our basic TDP construction (without any domain sampler) is one-way follows closely the one in [BPR15]. For completeness, we include here an overview of the proof taken almost verbatim from [BPR15].

We start by recalling the basic TDP construction. A given permutation is over the set of nodes $\{x_i\}_{i\in\mathbb{Z}_T}$. Each node x_i is a pair $(i, \mathsf{PRF}_S(i))$ where $\mathsf{PRF}_S : \mathbb{Z}_T \to \{0,1\}^\lambda$ is sampled from a family of pseudo-random functions, and $T \in \mathbb{N}$ is super-polynomial in the security parameter λ. The permutation is define by the cycle

$$x_1 \to x_2 \to \cdots \to x_T \to x_1 \ .$$

The public key describing the permutation consists the obfuscated program $\widetilde{\mathsf{F}}$ that maps x_i to x_{i+1} (where $i+1$ is computed modulo T) and outputs $\perp$ on any other input. The trapdoor is simply the seed S of the pseudo-random function that allows us to efficiently invert the permutation.

Inverting the permutation on x_i requires finding the end of the path starting at x_i and ending at x_{i-1}. Intuitively, this path can be thought of as an authenticated chain where a signature σ corresponding to some pair (j, σ) cannot be obtained without first obtaining all previous signatures on the path.

To prove that finding the signature $\mathsf{PRF}_S(i-1)$ of the node x_{i-1} is hard, we show that the obfuscated $\widetilde{\mathsf{F}}$ is computationally indistinguishable from a circuit that on input x_{i-1} returns $\perp$, rather than x_i as $\widetilde{\mathsf{F}}$ would. This implies that an efficient algorithm would not be able to obtain x_{i-1} from either one of the circuits, or it could distinguish the two. We next go in more detail into how indistinguishability of these two circuits is shown.

For every $\alpha, \beta \in \mathbb{Z}_T$ we consider the circuit $\widetilde{\mathsf{F}}_{\alpha,\beta}$ that is identical to $\widetilde{\mathsf{F}}$, except that for every j in the range from α to β (wrapping around T in case that $\alpha > \beta$) $\widetilde{\mathsf{F}}_{\alpha,\beta}$ on the input x_j outputs $\perp$. The argument proceeds in two steps. First, we show that for a *random* $u \in \mathbb{Z}_T$, the obfuscation $\widetilde{\mathsf{F}}_{u,u}$ is computationally indistinguishable from $\widetilde{\mathsf{F}}$. Intuitively this "splits" the authenticated chain into two parts: from x_i to x_u and from x_{u+1} to x_{i-1}. While given $\widetilde{\mathsf{F}}_{u,u}$ and x_i it is possible to compute additional signatures in the first part of the chain, we show that is it hard to find a signature for any j in the second part of the chain. More concretely, in the second step, we prove that the obfuscated circuits $\widetilde{\mathsf{F}}_{u,u}$ and $\widetilde{\mathsf{F}}_{u,i-1}$ are computationally indistinguishable by a sequence of hybrids. For every j in the range between u and $i-2$, we show that the obfuscations $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ are computationally indistinguishable. In total, we have at most T hybrids; relying on injective one-way functions and iO with super-polynomial hardness (related to T), we show that each two obfuscations are $T^{-\Theta(1)}$-indistinguishable. Overall we deduce that the obfuscations $\widetilde{\mathsf{F}}$ and $\widetilde{\mathsf{F}}_{u,i-1}$ are also computationally indistinguishable as required.

To summarize, the hardness proof follows two steps:

1. **Split the chain into two parts:** For a random $u \in \mathbb{Z}_T$, prove that $\widetilde{\mathsf{F}}$ and $\widetilde{\mathsf{F}}_{u,i-1}$ are indistinguishable.

2. **Erase second part:** For every j in the range between u and $i-2$ prove that $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ are $T^{-\Theta(1)}$-indistinguishable.

We next explain how the two steps described above are proven. For simplicity, we shall assume the existence of a length-doubling pseudo-random generator $\mathsf{PRG} : \mathbb{Z}_T \to \mathbb{Z}_{T^2}$ that is injective; in the body, we relax this assumption and rely only on injective one-way functions. The two steps rely the ideas of *hidden triggers and* punctured programs introduced by Sahai and Waters [SW14].

First step. To prove that $\widetilde{\mathsf{F}}$ is indistinguishable from $\widetilde{\mathsf{F}}_{u,u}$, we first note that $\widetilde{\mathsf{F}}$ is indistinguishable from an obfuscation $\widetilde{\mathsf{F}}^v(j, \sigma)$ of a circuit that has an extra "if statement":

1. if $\sigma = \mathsf{PRF}_S(j)$ and $\mathsf{PRG}(j) = v$, return $\perp$.

2. Otherwise return x_{j+1} (as $\widetilde{\mathsf{F}}$ would);

here v is chosen at random from the range $\mathbb{Z}_{T^2}$ of PRG. Observe that, with overwhelming probability $1 - \frac{1}{T}$, v is not in the image of PRG, the condition in (1) is never met, and the alternative behavior is never triggered. Thus, $\widetilde{\mathsf{F}}$ and $\widetilde{\mathsf{F}}^v$ compute the same function and their obfuscations are indistinguishable. Next, relying on the pseudo-randomness guarantee of PRG, we can indistinguishably replace the uniformly random v with a pseudo-random value $\mathsf{PRG}(u)$. It is left to note that, because PRG is injective, $\widetilde{\mathsf{F}}^v$, with $v = \mathsf{PRG}(u)$, computes the exact same function as $\widetilde{\mathsf{F}}_{u,u}$. Indeed, both compute the same function as $\widetilde{\mathsf{F}}$ except on x_u, where an alternative behavior is triggered and $\perp$ is returned.

Second step. To prove that $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ are indistinguishable, we require that PRF_S is *puncturable*. This means that for every input $k \in \mathbb{Z}_T$, we can sample a punctured $\mathsf{PRF}_{S\{k\}}$ that agrees with PRF_S on all inputs $\ell \neq k$, but computationally hides any information on the value $\mathsf{PRF}_S(k)$; namely $\mathsf{PRF}_S(k)$ is pseudo-random, even given the program $\mathsf{PRF}_{S\{k\}}$. Such puncturable pseudo-random functions are known to exist based on any one-way function [BW13, KPTZ13, BGI14].

Now, note that $\widetilde{\mathsf{F}}_{u,j}$ and $\widetilde{\mathsf{F}}_{u,j+1}$ differ only on input x_{j+1}: while the first returns x_{j+2}, the second returns $\perp$. In particular, the two circuits must hide x_{j+1} to guarantee indistinguishability. What enables hiding the value x_{j+1} is that both circuits never output $\mathsf{PRF}_S(j+1)$, but rather, on input x_j both return $\perp$. The value $\mathsf{PRF}_S(j+1)$ is only used to test if an input $(j+1, \sigma)$ satisfies $\sigma = \mathsf{PRF}_S(j+1)$. Relying on puncturing, this comparison can be performed in "encrypted form" while hiding $\mathsf{PRF}_S(j+1)$.

Concretely, we first move from $\widetilde{\mathsf{F}}_{u,j}$ to $\widetilde{\mathsf{F}}^{(1)}_{u,j}$ that has a punctured $\mathsf{PRF}_{S\{j+1\}}$. The circuit has $\sigma_{j+1} = \mathsf{PRF}_S(j+1)$ hardwired, and given $(j+1, \sigma)$ it directly compares σ to σ_{j+1}. The circuit computes the same function, and indistinguishability holds by iO. Then, relying on pseudo-randomness at the punctured point $j+1$, we move to $\widetilde{\mathsf{F}}^{(2)}_{u,j}$ where σ_{j+1} is replaced with a truly random value in $\{0,1\}^\lambda$. Next, we move to $\widetilde{\mathsf{F}}^{(3)}_{u,j}$ where the comparison of σ_{j+1} and σ is not done in the clear, but rather under an injective (length-doubling) pseudo-random generator $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$; in particular, σ_{j+1} is not stored in the clear, but rather $\mathsf{PRG}(\sigma_{j+1})$ is stored. This does not change functionality and thus indistinguishability is again preserved. Now, using pseudo-randomness of PRG, we move to yet another circuit $\widetilde{\mathsf{F}}^{(4)}_{u,j}$, where $\mathsf{PRG}(\sigma_{j+1})$ is replaced by a truly random string $v \in \{0,1\}^{2\lambda}$. Finally, note that as in the proof of the first step, v is not in the image of PRG with overwhelming probability $1 - 2^{-\lambda}$. Thus we can indistinguishably change the circuit to return $\perp$ when given the input x_{j+1}. We can then reverse the above steps and go back to computing $\sigma_{j+1} = \mathsf{PRF}_S(j+1)$, using the non-punctured PRF_S.

To guarantee the required indistinguishability gap between the different hybrids, we should take care in choosing the parameters T, the output length λ of PRF_S, and the hardness of each of the cryptographic primitives. Choosing these parameters yields different hardness tradeoffs (further discussed in Section 4.4).

B On non-interactive commitments from iO

As mentioned above, our result allows to remove additional assumptions in iO based construction of several fundamental primitives including non-interactive commitments. Here we elaborate on this implication.

In the construction of non-interactive commitments based on injective one-way functions [Blu81] it is crucial that a malicious sender cannot send a malformed function key defining a function that is not injective. We observe however, that it is sufficient to require that every key in the support of the key-generation algorithm defines an injective function. If the function family has this property, we can modify the decommitment algorithm and have sender present its random coins, proving that the key is indeed in the support of the key-generation algorithm. While our injective one-way function do not satisfy the above property, our trapdoor permutations do.