On the Impossibility of Basing Public-Coin One-Way Permutations on Trapdoor Permutations

Takahiro Matsuda

Research Institute for Secure Systems (RISEC),
National Institute of Advanced Industrial Science and Technology (AIST), Japan
t-matsuda@aist.go.jp

Abstract. One of the fundamental research themes in cryptography is to clarify what the minimal assumptions to realize various kinds of cryptographic primitives are, and up to now, a number of relationships among primitives have been investigated and established. Among others, it has been suggested (and sometimes explicitly claimed) that a family of one-way trapdoor permutations (TDP) is sufficient for constructing almost all the basic primitives/protocols in both "public-key" and "private-key" cryptography. In this paper, however, we show strong evidence that this is not the case for the constructions of a one-way permutation (OWP), one of the most fundamental primitives in private cryptography. Specifically, we show that there is no black-box construction of a OWP from a TDP, even if the TDP is *ideally secure*, where, roughly speaking, ideal security of a TDP corresponds to security satisfied by random permutations and thus captures major security notions of TDPs such as one-wayness, claw-freeness, security under correlated inputs, etc. Our negative result might at first sound unexpected because both OWP and (ideally secure) TDP are primitives that implement a "permutation" that is "one-way". However, our result exploits the fact that a TDP is a "secret-coin" family of permutations whose permutations become available only after some sort of key generation is performed, while a OWP is a publicly computable function which does not have such key generation process.

Keywords: black-box separation, trapdoor permutation, one-way permutation, family of one-way permutations.

1 Introduction

1.1 Background and Motivation

One of the fundamental research themes in cryptography is to clarify what the minimal assumptions to realize various kinds of cryptographic primitives are, and up to now, a number of relationships among primitives have been investigated and established. Clarifying these relationships gives us a lot of insights for how to construct and/or prove the security of cryptographic primitives, enables us to understand the considered primitives more deeply, and leads to systematizing the research area in cryptography.

In this paper, we focus on two central cryptographic primitives, a family of trapdoor permutations (TDP) and a one-way permutation (OWP). Among others, it has been suggested, and sometimes explicitly claimed (see, e.g. [9]), that a TDP is sufficient for constructing (almost) all basic primitives/protocols in both "public-key" and "private-key" cryptography. In particular, it has been shown that a TDP can be used for constructing a family of one-way trapdoor functions, public-key encryption schemes, key agreement protocols, private information retrieval, oblivious transfer, etc. Moreover, it has also been shown that a OWP is sufficient to construct most of private-key cryptographic primitives/protocols including symmetric key encryption schemes, message authentication codes, digital signature schemes [37], pseudorandom generators/functions/permutations [7, 47, 16, 32], bit commitment schemes [35], etc. (Some of them later turned out to be possible to construct from any one-way function, e.g. a pseudorandom generator from any one-way function [22].) These primitives can also be constructed from a TDP as well.

Somewhat surprisingly, however, the following simple but fundamental question has not been answered yet: "*Can we construct a OWP from a TDP?*" The main motivation of this paper is to clarify the answer to this question, in order to fully establish the relationships among these very basic and important primitives. One might think that the answer is trivially yes (and that this is obvious), because a TDP is trivially a family of one-way permutations if we keep trapdoors secret. However, we show strong evidence that the answer to the above question is *no* by showing that there is no *black-box construction* of a OWP from a TDP. Roughly, a black-box construction of a target primitive P from a building block primitive Q requires that the construction of P treats an instance of Q as a black-box (i.e. treats as an oracle) and furthermore that the reduction algorithm for the security proof treats an adversary that breaks the security of the construction of P (and the instance of Q) as a black-box. (The impossibility of the opposite direction, i.e. constructing a TDP from a OWP in a black-box way, is due to [25].)

Actually, to tackle the above question, we have to be careful about the difference between a "single" one-way permutation and a "family" of one-way permutations (one-way permutation family, OWPF).[1] Our black-box separation result mentioned above separates a "single" one-way permutation from a TDP. Furthermore, for OWPFs, we have to be also careful about the difference between the *public-coin* case and the *secret-coin* case. Informally, a OWPF is said to be *public-coin* if the randomness for choosing a permutation from the family can be revealed together with the description of the permutation. On the other hand, a OWPF is said to be *secret-coin* if the security (one-wayness) is not guaranteed if the randomness is revealed. (The distinction between public-coin primitives and secret-coin primitives is studied by Hsiao and Reyzin [24] for the case of collision-resistant hash function families.) With these categorizations, it is straightforward to see that any one-way TDP can always be seen as a secret-coin OWPF by regarding an evaluation-key (public-key) output from a key generation algorithm of the TDP as an index specifying a permutation in the family. However, the same OWPF derived from a TDP is *not* secure as a public-coin OWPF, because the randomness for choosing the evaluation-key (public-key) cannot be revealed: If revealed, then anyone can compute the corresponding trapdoor, which makes the permutation invertible. Furthermore, it is also straightforward to see that a single OWP is a special type of a public-coin OWPF (by implementing the permutations in the family with the given

[1] In order not to mix up with the difference between single function and function family of one-way permutations, when we just write "OWP", we always mean it is a "single" one-way permutation (i.e. not a family), and when we mean a family of OWPs, we write "OWP**F**".

single OWP). Here, what is not at all trivial is whether we can construct a public-coin OWPFs from a TDP in general. We also partially answer to this question in the negative.

1.2 Our Contribution

In this paper, we show that there is no black-box construction of a OWP from a TDP, even if the TDP is *ideally secure* [11, 29], where, roughly speaking, ideal security of a TDP corresponds to the security satisfied by random permutations (see Section 2.3 for the formal definition), and thus captures major security notions for a TDP such as one-wayness, claw-freeness [19], security under correlated inputs [42], etc. Therefore, our impossibility result rules out the black-box constructions of a OWP from TDP satisfying these security notions, and is strictly stronger than the result by Chang et al. [9] who showed the black-box separation of a OWP from a family of injective trapdoor functions. Our impossibility result might at first sound unexpected because both OWP and (one-way) TDP are primitives that implement a "permutation" that is "one-way". However, our result is established by exploiting the essential difference between a family of functions and a single function, that a TDP is a "secret-coin" family of permutations whose permutations become available only after some sort of key generation is performed, while a OWP is a publicly computable function which does not have such key generation process. (We explain the overview of the proof in Section 1.3.)

The type of black-box constructions that our main result rules out is called a *fully*-black-box construction in the taxonomy of Reingold et al. [41]. (The formal definition for a fully-black-box construction of a OWP from an ideal TDP is given in Section 3.) In fact, our result can be easily strengthened to rule out a *semi*-black-box construction, which is a less restrictive type than fully-black-box one, using the technique called "embedding" by Reingold et al. [41]. (We discuss this extension in Section 4.) Although the absence of (fully- and semi-)black-box constructions of a OWP from an ideal TDP does not necessarily mean that constructing a OWP from an ideal TDP is generally impossible, it should be emphasized that most of the known primitive-to-primitive constructions are fully-black-box, and thus the impossibility of black-box constructions is considered as a very strong evidence that "natural" and "efficient" constructions are impossible.

Our result also sheds light on the difference between "public-coin" and "secret-coin" OWPFs (their formal definitions can be found in Section 2.2). Whether a primitive remains secure in the sense of public-coin is usually related to whether we need some kind of trusted setup in a cryptographic protocol such as multi-party computation. Hsiao and Reyzin [24] conjectured that there is no (fully-)black-box construction of a public-coin OWPF from a secret-coin one. We partially answer to this conjecture: Specifically, we show that there is no black-box construction of a public-coin OWPF that satisfies a special property called *canonical domain sampling* (the formal definition is given in Section 2.2) from an ideal TDP (and especially from a secret-coin OWPF). This result is obtained as a corollary of our main result above by combining it with the result by Goldreich et al. [17] who showed that a OWP can be constructed, in a black-box manner, from a public-coin OWPF with the canonical domain sampling property. (See Section 4 for more details.) We note that the techniques we use to prove the black-box separation of a public-coin OWPF from a secret-coin one (and the black-box separation of a OWP from an ideal TDP) are different from those used by Hsiao and Reyzin in [24] (in fact, we use a part of the results in [24]).

Why Studying OWP vs. TDP? Historically, OWP and (public-coin/secret-coin) OWPF have much more often been treated as assumptions rather than as target primitives that are constructed from other primitives, and thus one may wonder why we should care the (im)possibility of constructing a OWP from TDP (or from other primitives).

Our opinion is that firstly, OWP, OWPF, and TDP are very basic primitives, and thus clarifying any of their properties as well as relations is important, and we believe that our results contribute to correctly understanding and firmly establishing relationships among these basic cryptographic primitives. Specifically, our results suggest that there is no simple hierarchy of black-box constructions even among very basic cryptographic primitives. Our results also clarify explicitly that there is a real difference among single function, public-coin and secret-coin families of functions in the case of permutations, which should be contrasted with the case of "functions" because the existence of a single one-way function is equivalent to the existence of a family of one-way functions (regardless of whether the family is secret-coin or public-coin). Furthermore, our results also show that it is not always the case that "public-key"-type primitives are stronger than "non-public-key"-type primitives (at least in the case of permutations). This should be again contrasted with the case of "functions", where there is a (trivial) black-box construction of a one-way function from basically all known "public-key"-type primitives (because key generation algorithms typically have to be a one-way function), but there does not exist a black-box construction for the opposite direction [25].

Secondly, there might actually be a cryptographic primitive that can be constructed from a OWP, but not from a TDP. One of such candidates may be a public-coin point obfuscation (an obfuscator for a point function) [1, 45]. Wee [45] showed that a point obfuscator can be constructed from a (very strong) OWP, while his point obfuscator does not seem to be proved secure if we replace the OWP in his construction with a permutation from a TDP together with its public-key (at least the "public-coin" property will be lost unless we assume some additional property for the TDP). We believe that there are much more (natural) examples of this sort, and that it is interesting to seek for such examples. (In particular, the difference between public-coin and secret-coin primitives will stand out more in the context of interactive protocols.)

1.3 Technical Overview

The main result of our paper builds on the results and techniques from several previous work [43, 26, 15, 24, 9, 30, 23], and our technical contribution lies in coming up with an appropriate combination of these results/techniques for achieving our purpose of separating OWP from (ideal) TDP.

We will use the "two oracle separation" paradigm [15, 24] (which is an extension of the one oracle separation [25, 41]) to show that there is no fully-black-box construction of a OWP from an ideal TDP. That is, we will use two oracles (more precisely, a random instance picked from all possible instances of oracles): the first oracle models a "building block" primitive (TDP in our case) and the second oracle is the "breaking" oracle that is useful for breaking all candidates of a target primitive (OWP in our case) but useless for breaking the security of the building block oracle. As the "building block" oracle, we use a random instance of a *TDP oracle* $\mathcal{T}$ that consists of suboracles $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ that essentially constitutes a (random) TDP, namely, $\mathcal{G}$ is the key generation, $\mathcal{E}$ is the evaluation of permutations, and $\mathcal{D}$ is the inversion of permutations. As the "breaking" oracle, we use the PSPACE oracle that has often been used in the literature of black-box separations, e.g. [25, 15, 9], mainly in order to guarantee that any computational hardness comes only from the building block oracle. If we pick $\mathcal{T}$ randomly, then $\mathcal{T}$ can be shown to be "ideally secure" even against computationally unbounded adversary that makes only polynomially many queries to $\mathcal{T}$. Since such adversary can simulate the PSPACE oracle by itself, it follows that an "ideally secure" TDP exists relative to $\mathcal{T}$ and PSPACE.

The difficult part of the proof is to show that any permutation $\mathsf{P}^{\mathcal{T}}$ is inverted, and thus a OWP does not exist relative to $\mathcal{T}$ and PSPACE. Here, we note that the evaluation-key space of $\mathcal{T}$ cannot be *dense* [20] (i.e. an inverse-polynomial fraction of strings are in the range of $\mathcal{G}$), because in this case, an evaluation-key ek of permutations in $\mathcal{E}$ could be picked without using $\mathcal{G}$, and thus implementing a permutation $\mathsf{P}^{\mathcal{T}}$ by the permutation (in $\mathcal{E}$) made available by this picked ek might lead to a OWP (even in the presence of the PSPACE oracle). To prevent this, we make the range of $\mathcal{G}$ sparse, and make $\mathcal{E}$ useless unless it is invoked with an honestly generated evaluation-key that is generated by making a query to $\mathcal{G}$. This guarantees that when calculating the permutation $\mathsf{P}^{\mathcal{T}}$, permutations in $\mathcal{E}$ become available only after making a query to $\mathcal{G}$ and obtaining an evaluation-key ek, *together with the corresponding trapdoor* td. Put differently, from the viewpoint of an entity computing the permutation $\mathsf{P}^{\mathcal{T}}$, every permutation in $\mathcal{E}$ associated with ek that becomes available during the computation of $\mathsf{P}^{\mathcal{T}}$ can be seen as an *invertible permutation*, because the entity must have known td corresponding to ek. This observation leads to the idea of simulating the TDP oracle $\mathcal{T}$ in $\mathsf{P}^{\mathcal{T}}$ with a *block cipher* oracle, which is a family of invertible permutations. More specifically, we introduce a new oracle $\mathcal{B}$, which we call *block cipher* oracle that models an ideally secure block cipher, and show that for any permutation $\mathsf{P}^{\mathcal{T}}$, there is another permutation $\widehat{\mathsf{P}}^{\mathcal{B}}$ such that inverting $\widehat{\mathsf{P}}^{\mathcal{B}}$ is as hard as inverting $\mathsf{P}^{\mathcal{T}}$. The idea and the technique of simulating a TDP oracle $\mathcal{T}$ (used in a constructed primitive) with a block cipher oracle is previously used by Lindell and Zarosim [30] who showed the black-box separation of an adaptively secure oblivious transfer protocol from a TDP. Furthermore, by using the result by Holenstein et al. [23] who showed that a random invertible permutation is simulatable by the fourteen-round Feistel-network construction of a permutation [32] in which each round function is an independent random function,[2] we can simulate the block cipher oracle $\mathcal{B}$ in the permutation $\widehat{\mathsf{P}}^{\mathcal{B}}$ with another oracle $\mathcal{R}$ (which we call *round function oracle*) that consists only of random functions (not permutations). More specifically, we show that for any permutation $\widehat{\mathsf{P}}^{\mathcal{B}}$, there is another permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$ such that inverting $\widetilde{\mathsf{P}}^{\mathcal{R}}$ is as hard as inverting $\widehat{\mathsf{P}}^{\mathcal{B}}$. Finally, using the previous results by Rudich [43], Kahn et al. [26], and Chang et al. [9] on the black-box separations of

[2] More precisely, [23] shows that the fourteen-round Feistel-network is *indifferentialble* [34] from an (invertible) random permutation. The statement that a constant-round Feistel-network was sufficient was originally suggested by Coron et al. [10]. However, it was pointed out in [23] that the original proof in [10] for six rounds had a gap and was not completed.

a OWP from random (injective) functions, we can show that there is a good inverter (which uses the PSPACE oracle) for any permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$.[3] Then, this inverter can be used to invert not only $\widetilde{\mathsf{P}}^{\mathcal{R}}$ but also $\mathsf{P}^{\mathcal{T}}$, and thus any permutation $\mathsf{P}^{\mathcal{T}}$ is inverted using the PSPACE oracle.

It is already known that a OWP is black-box separated from a one-way function (OWF) [43, 26] and that there is a black-box construction of a pseudorandom permutation, which is a standard security notion of a block cipher, from a OWF [22, 16, 32]. Therefore, one might wonder that if we give up the "ideal security" of a TDP and just consider one-way TDPs, then we may be able to conclude that there is no black-box construction of a OWP from a one-way TDP, as soon as we reduce a TDP-based permutation $\mathsf{P}^{\mathcal{T}}$ to a block-cipher-based permutation $\widehat{\mathsf{P}}^{\mathcal{B}}$. However, that a OWP is separated from a OWF in a black-box manner does not immediately mean that our block-cipher-based permutation $\widehat{\mathsf{P}}^{\mathcal{B}}$ cannot be proved one-way, because our block-cipher oracle $\mathcal{B}$ contains random permutations which may help $\widehat{\mathsf{P}}^{\mathcal{B}}$ to be one-way (with some clever use of permutations in $\mathcal{B}$). This is the main reason why we further reduce the block-cipher-based permutation $\widehat{\mathsf{P}}^{\mathcal{B}}$ to a random function-based permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$ by using the result of [23], so that random permutations in the oracle $\mathcal{B}$ do not help achieving a OWP any better than random "functions" in the oracle $\mathcal{R}$ do.

1.4 Related Work

Up to now, a number of black-box separations among various kinds of primitives have been established. For an excellent survey of the literature and the techniques of black-box separations, we refer the reader to [48]. Here, we review black-box separations related to OWPs and TDPs.

Regarding the black-box separations of a OWP from other primitives, it is known that it is separated from one-way functions [43, 26], from injective trapdoor functions and a private information retrieval protocols [9], and from length-increasing injective one-way functions (even if they are just 1-bit-increasing) [33].

On the other hand, recently, several black-box separation results have shown the limitations of a (one-way) TDP as a base primitive for constructing and/or proving the security of several "highly functional" cryptographic primitives or basic primitives with special functional/security properties. Those include the impossibility of constructing identity-based encryption [8], a wide class of predicate encryption [27], lossy trapdoor functions [42], trapdoor functions secure under correlated inputs [44], encryption schemes secure under key-dependent inputs [21], adaptively secure oblivious transfer protocols [30], non-interactive or perfectly binding commitment schemes secure under selective-opening attacks [2], verifiable random functions [12], a natural class of three-move blind signature schemes [13], succinct non-interactive argument systems [14], constant-round sequentially witness-hiding special-sound protocols for unique witness

[3] We note that a random function (which is length preserving) is indistinguishable from a random permutation for any (even computationally unbounded) algorithm that can make only polynomially many queries to the random function (even in the presence of the PSPACE oracle), but this fact does not mean that we can construct a OWP from a random function in a black-box way (in fact, it is not possible [43, 26, 9, 33]).

relations [39], and many of the cryptographic primitives that admit the so-called simulatable attacks [46]. We note that in fact, the results of [21, 2, 13, 14, 39, 46] rule out the possibility of constructions (and/or, security proofs) of the target primitives based not only on one-way TDP but also on much broader class of primitives or assumptions, such as all falsifiable assumptions [36].

Black-box separations for a particular construction that uses a TDP as a building block are also known. The unforgeability of the FDH signature scheme [4] cannot be based on an ideal TDP, if the TDP is treated as a black-box [11]. [6] shows a similar result for the PSS signature scheme, and [29] shows the impossibility of basing chosen ciphertext security of padding-based encryption schemes which include many known TDP-based encryption schemes such as the OAEP encryption scheme [3], on the (ideal) security of the building block TDP.

1.5 Paper Organization

The rest of this paper is organized as follows. In Section 2 we review some basic definitions and terminology. In Section 3, we show our main result on the black-box separation of a OWP from an ideal TDP, and we discuss further results, and the possibility of more general separation results in Section 4.

2 Preliminaries

In this section, we review the basic notation and the definitions of primitives.

Basic Notation. $\mathbb{N}$ denotes the set of natural numbers. For $n \in \mathbb{N}$, we define $[n] = \{1, \dots, n\}$. If x and y are strings, then "$|x|$" denotes the bit-length of x, and "$(x||y)$" denotes a concatenation of x and y. "$x \leftarrow y$" denotes an assignment of y to x. If S is a set then "$|S|$" denotes its size, and "$x \leftarrow_{\mathtt{R}} S$" denotes that x is chosen uniformly at random from S. "PPTA" denotes *probabilistic polynomial time algorithm*. If $\mathcal{A}$ is a probabilistic algorithm, then "$z \leftarrow_{\mathtt{R}} \mathcal{A}(x, y, \dots)$" means that $\mathcal{A}$ takes x, y, $\dots$ as input and outputs z, and "$z \leftarrow \mathcal{A}(x, y, \dots; r)$" means that $\mathcal{A}$ takes x, y, $\dots$ as input, uses r as an internal randomness, and outputs z. For an oracle algorithm $\mathcal{A}^{\mathcal{O}}$, we say that $\mathcal{A}^{\mathcal{O}}$ has query complexity q if $\mathcal{A}$ makes queries to the oracle $\mathcal{O}$ at most q times. "Perm_n" denotes the set of all permutations over $\{0, 1\}^n$. If f is a function and D is its domain, then we define $\mathsf{Range}(f) = \{f(x)|x \in D\}$.

A function $f : \mathbb{N} \to [0, 1]$ is said to be *negligible* if $f(k) < 1/p(k)$ for all positive polynomials $p(k)$ and all sufficiently large $k \in \mathbb{N}$, and a function $g : \mathbb{N} \to [0, 1]$ is said to be *overwhelming* if the function $f(k) = 1 - g(k)$ is negligible.

2.1 One-Way Permutations

Typically, security of a OWP is defined so that the security parameter k is its input length. However, since later we consider constructions of a OWP from another primitive, it will be convenient to consider the security parameter and the input length of the constructed permutation separately, so that the one-wayness advantage of an adversary and the input length of the constructed permutation are a function of the security parameter of the building block. Moreover, it is also convenient to identify a (one-way) permutation with a PPTA that computes it. Therefore, we take these approaches for the definition of a OWP.

Let $\ell = \ell(k)$ be a positive polynomial and P be a PPTA such that P is a permutation over $\{0,1\}^\ell$. We say that a PPTA P is a *one-way permutation (OWP) for length ℓ* if the following advantage function $\mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P},\mathcal{A},\ell}(k)$ is negligible for any PPTA adversary $\mathcal{A}$ (we assume that P is also given 1^k but omit to write it for simplicity):

$$\mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P},\mathcal{A},\ell}(k) = \Pr[x^* \leftarrow_{\mathtt{R}} \{0,1\}^\ell; y^* \leftarrow \mathsf{P}(x^*); x' \leftarrow_{\mathtt{R}} \mathcal{A}(1^k, y^*) : x' = x^*].$$

2.2 One-Way Permutation Families

A family of permutations (permutation family) PF consists of the following three PPTAs (Gen, Eval, Samp): Gen is the probabilistic evaluation-key generation algorithm which takes 1^k as input and outputs an evaluation-key ek. (An evaluation-key is also called an index.) Eval is the deterministic evaluation algorithm which takes ek and an element $x \in D_{ek}$ as input, and outputs $y \in D_{ek}$, where D_{ek} is the domain of $\mathsf{Eval}(ek, \cdot)$ that is determined by ek. Samp is the probabilistic sampling algorithm which takes ek as input, and outputs a (random) element $x \in D_{ek}$. As a correctness requirement, we require that for all $k \in \mathbb{N}$ and all $ek \leftarrow_{\mathtt{R}} \mathsf{Gen}(1^k)$, (i) $\mathsf{Samp}(ek)$ is a uniform distribution over D_{ek}, and (ii) $\mathsf{Eval}(ek, \cdot)$ is a permutation over D_{ek}.

We say that $\mathsf{PF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp})$ is a *one-way permutation family (OWPF)* if the following advantage function $\mathsf{Adv}^{\mathtt{OWPF}}_{\mathsf{PF},\mathcal{A}}(k)$ is negligible for any PPTA adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathtt{OWPF}}_{\mathsf{PF},\mathcal{A}}(k) = \Pr[ek \leftarrow_{\mathtt{R}} \mathsf{Gen}(1^k); x^* \leftarrow_{\mathtt{R}} \mathsf{Samp}(ek); y^* \leftarrow \mathsf{Eval}(ek, x^*);$$
$$x' \leftarrow_{\mathtt{R}} \mathcal{A}(ek, y^*) : x' = x^*].$$

If a permutation family PF remains one-way even when $\mathcal{A}$ is given the randomness r that is used to generate $ek = \mathsf{Gen}(1^k; r)$, then we call PF a *public-coin*[4] OWPF, and in order to distinguish it from an ordinary one, we call an ordinary OWPF a *secret-coin* OWPF.

Canonical Domain Sampling Property. We say that a OWPF PF has the *canonical domain sampling* property [17] if the following two conditions are satisfied:

1. **(Recognizable domain)** There exists a PPTA which, on input ek and x, tells if $x \in D_{ek}$ or not.
2. **(Dense domain)** There exist a polynomial time computable function $\ell = \ell(k)$ and a positive polynomial $p = p(k)$ so that $D_{ek} \subseteq \{0,1\}^\ell$ and $|D_{ek}| > 2^\ell / p$.

Goldreich et al. [17] showed that a OWP can be constructed in a black-box manner from a public-coin OWPF with the above property, and we briefly review their construction. Given a public-coin OWPF $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp})$ with the canonical domain

[4] Goldreich et al. [17] called this property "*augmented one-wayness*." Here we use the name due to Hsiao and Reyzin [24].

sampling property, where $\mathsf{Gen}(1^k)$ uses a $\lambda = \lambda(k)$-bit randomness, we construct a single permutation P for length $\lambda + \ell$ that works as follows: On input $(r_g \| z)$ such that $|r_g| = \lambda$ and $|z| = \ell$, P first calculates $ek \leftarrow \mathsf{Gen}(1^k; r_g)$, and then outputs $(r_g \| \mathsf{Eval}(ek, z))$ if $z \in D_{ek}$ or $(r_g \| z)$ otherwise. This P is indeed a permutation, and can be shown to be weakly one-way. Then, this weak one-wayness can be amplified by a standard technique (e.g. [47]) to obtain a OWP (with ordinary one-wayness).

2.3 Trapdoor Permutations

A family of trapdoor permutations (TDP) is a special class of secret-coin permutation family $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp})$ with the following additional properties: (1) The algorithm Gen is a deterministic polynomial-time algorithm that takes 1^k and a trapdoor $td \in \{0,1\}^k$ as input, and outputs a corresponding evaluation-key ek.[5] (This process is denoted by "$ek \leftarrow \mathsf{Gen}(1^k, td)$".) (2) There is a deterministic *inversion* algorithm Inv which takes $td \in \{0,1\}^k$ and an element $y \in D_{ek}$ as input (where $ek = \mathsf{Gen}(1^k, td)$), and outputs $x \in D_{ek}$ such that $\mathsf{Eval}(ek, x) = y$.

Hard Games and Ideal Security. In this paper, we consider "ideal security" of a TDP, following [11, 29]. Roughly, ideal security of a TDP corresponds to security satisfied by random permutations.

Let G be a PPTA (called a challenger) that can exchange messages with another algorithm (called an adversary) $\mathcal{A}$ by a shared communication tape. We say that G defines a game regarding random permutations if both G and $\mathcal{A}$ have access to t independent random permutations $\pi_1, \dots, \pi_t$ over $\{0,1\}^k$, where $t = t(k)$ is a polynomial determined by G, G interacts with $\mathcal{A}$, and finally outputs a decision bit d. This process is denoted by "$d \leftarrow_{\mathtt{R}} \mathsf{Expt}^{\mathsf{G}^{\pi_1(\cdot),\dots,\pi_t(\cdot)}}_{\mathrm{RP},\mathcal{A}^{\pi_1(\cdot),\dots,\pi_t(\cdot)}}(k)$." (Here, "RP" stands for "random permutations.") We say that the adversary $\mathcal{A}$ wins the game G if $d = 1$.

Informally, an oracle PPTA G defines a δ-hard game regarding random permutations, where $0 \le \delta < 1$, if no oracle algorithm $\mathcal{A}$ can win the game G regarding random permutations with probability significantly better than δ. Typically, $\delta = 0$ for "search games" (e.g. one-wayness experiment) or $\delta = 1/2$ for "distinguishing games" (e.g. security experiment for a pseudorandom generator). We define the advantage of an adversary $\mathcal{A}$ in a game G as follows:

$$\mathsf{Adv}^{\mathsf{G}}_{\mathrm{RP},\mathcal{A}}(k) = \Pr[\pi_1, \dots, \pi_t \leftarrow_{\mathtt{R}} \mathsf{Perm}_k; d \leftarrow_{\mathtt{R}} \mathsf{Expt}^{\mathsf{G}^{\pi_1(\cdot),\dots,\pi_t(\cdot)}}_{\mathrm{RP},\mathcal{A}^{\pi_1(\cdot),\dots,\pi_t(\cdot)}}(k) : d = 1].$$

Then, we define the δ-hardness of the game G as follows.

Definition 1. *We say that a game* G *is* δ-hard (for some $0 \le \delta \le 1$) for adversaries with polynomial query complexity *if for any (even computationally unbounded) algorithm $\mathcal{A}$ whose query complexity is at most polynomial, there is a negligible function $\mu(k)$ such that* $\mathsf{Adv}^{\mathsf{G}}_{\mathrm{RP},\mathcal{A}}(k) - \delta \le \mu(k)$. *We call "$\delta(\mathsf{G})$" the* hardness *of the game* G *and is the smallest value such that* G *is δ-hard for adversaries with polynomial query complexity.*

[5] It is usual to define the Gen algorithm as a probabilistic algorithm so that it takes 1^k as input, and outputs a pair (ek, td). However, in terms of existence, a TDP with such definition is equivalent to one defined in this paper, because without loss of generality we can identify the randomness r for generating $(ek, td) \leftarrow \mathsf{Gen}(1^k; r)$ with the trapdoor of a TDP.

We stress that unlike [11, 29], our definition of the hardness $\delta(\mathsf{G})$ of a game G regarding random permutations is with respect to *computationally unbounded* adversaries, and the restriction on an adversary is only on its query complexity, rather than its running time. Though this requirement for hard games is stronger than the ones used in [11, 29] (and thus potentially harder to achieve), most security games that are δ-hard for all PPTAs remain δ-hard for computationally unbounded adversaries with polynomial query complexity. Examples include one-wayness, claw-freeness [19], and security under $t(k)$-correlated inputs [42] for any predetermined polynomial $t(k)$. See also [29, Table 1] for other types of security games that can be captured by δ-hard games. We note that, since G does not have access to inversions of permutations, our definition of hard games does not capture adaptive one-wayness [38, 28].

A game for a TDP is then defined by replacing the random permutations in a δ-hard game with instantiations of permutations in the TDP. More specifically, we define the advantage of an adversary $\mathcal{A}$ in a game G for a TDP $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp}, \mathsf{Inv})$ as follows:

$$\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP},\mathcal{A}}(k) = \Pr\left[\begin{array}{c} td_1, \dots, td_t \leftarrow_{\mathrm{R}} \{0,1\}^k; ek_i \leftarrow \mathsf{Gen}(1^k, td_i) \text{ for } i \in [t] \\ d \leftarrow_{\mathrm{R}} \mathsf{Expt}^{\mathsf{G}^{\mathsf{Eval}(ek_1,\cdot),\dots,\mathsf{Eval}(ek_t,\cdot)}}_{\mathsf{TDP},\mathcal{A}(ek_1,\dots,ek_t)}(k) \end{array} : d = 1\right]$$

Note that in the above experiment, the interface of G is exactly the same as that of a game defined for random permutations. However, the interface of $\mathcal{A}$ is changed. Unlike the games regarding random permutations, we do not provide $\mathcal{A}$ with oracle access to $\mathsf{Eval}(ek_i, \cdot)$'s because it gets evaluation keys $\{ek_i\}$ and thus can compute each $\mathsf{Eval}(ek_i, \cdot)$ by itself.

Definition 2. *We say that* TDP *is* secure for game G *if for all PPTAs $\mathcal{A}$, there is a negligible function $\mu(k)$ such that* $\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP},\mathcal{A}}(k) - \delta(\mathsf{G}) \le \mu(k)$. *Furthermore, we say that* TDP *is an* ideal TDP *if it is secure for all games.*

Note that the definition of the hard games for a TDP considers only PPTA adversaries, although the hardness $\delta(\mathsf{G})$ is defined with respect to (computationally unbounded) adversaries with polynomial query complexity.

It has been observed in [11] that ideal security is too strong to be satisfied by TDPs implemented by PPTAs. However, we will show the *impossibility* of constructing a OWP from an ideal TDP in a black-box manner, and thus ruling out a black-box construction from a TDP with such strong security makes our result *stronger*.

3 Black-box Separation of OWP from Ideal TDP

In this section, we show our main result: there is no black-box construction of a OWP from an ideal TDP.

We first recall the formal definition of the type of black-box constructions that we will rule out, which is called a *fully*-black-box construction (reduction) in the taxonomy of Reingold et al. [41]. (The definition can be easily adapted to other primitives.)

Definition 3. *We say that there exists a fully-black-box construction of a OWP from an ideal TDP, if there exist a positive polynomial $\ell = \ell(k)$, an oracle PPTA* P *(called* *"*construction*"), and an oracle PPTA $\mathcal{R}$ (called "*reduction*") such that for all tuples of algorithms* $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp}, \mathsf{Inv})$ *that implement a TDP with security parameter k and all algorithms $\mathcal{A}$ (where each algorithm in* TDP *and $\mathcal{A}$ are of arbitrary complexity) the following two conditions hold:*

(Correctness): $\mathsf{P}^{\mathsf{TDP}}$ *is a permutation over $\{0,1\}^{\ell}$.*
(Security): *If* $\mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P}^{\mathsf{TDP}},\mathcal{A},\ell}(k)$ *is non-negligible, then so is* $\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP},\mathcal{R}^{\mathcal{A},\mathsf{TDP}}}(k) - \delta(\mathsf{G})$ *for some game* G*.*

The main result in this paper is the following.

Theorem 1. *There is no fully-black-box construction of a OWP from an ideal TDP.*

Recall that the security games for most of the security notions of a TDP, such as (ordinary) one-wayness, security under $t(k)$-correlated-inputs [42] for any predetermined polynomial $t = t(k)$, and claw-freeness [19], can be captured by the δ-hard games. Since "a (fully-)black-construction of a primitive from another primitive" is a transitive relation, we obtain the following as a corollary of Theorem 1.

Corollary 1. *There is no fully-black-box construction of a OWP from a one-way TDP*[6]*, a TDP secure under t-correlated-input for any predetermined polynomial t, or a claw-free TDP.*

To prove Theorem 1, we will use the following "two oracle separation" technique [15, 24] (which is an extension from the "one oracle separation" by [25, 41]). Specifically, to prove Theorem 1, it is sufficient to show the following lemma.

Lemma 1. *(adapted from [15, 24].) Let* PSPACE *be an oracle for a* PSPACE*-complete problem. Assume there exist a set $\mathbb{O}$ of oracles and a tuple of oracle PPTAs* $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp}, \mathsf{Inv})$ *that satisfy the following three conditions:*

(1): $\mathsf{TDP}^{\mathcal{O}} = (\mathsf{Gen}^{\mathcal{O}}, \mathsf{Eval}^{\mathcal{O}}, \mathsf{Samp}^{\mathcal{O}}, \mathsf{Inv}^{\mathcal{O}})$ *is correct as a TDP for all $\mathcal{O} \in \mathbb{O}$.*
(2): For any game G *and for any oracle PPTA $\mathcal{A}$,* $\mathbf{E}_{\mathcal{O}\leftarrow_{\mathsf{R}}\mathbb{O}}[\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathsf{PSPACE}}}(k)] - \delta(\mathsf{G})$ *is negligible.*
(3): For any positive polynomial $\ell = \ell(k)$ and for any oracle PPTA P*, if* $\mathsf{P}^{\mathcal{O}}$ *is a permutation over $\{0,1\}^{\ell}$ for all $\mathcal{O} \in \mathbb{O}$, then there exists an oracle PPTA $\mathcal{A}$ such that* $\mathbf{E}_{\mathcal{O}\leftarrow_{\mathsf{R}}\mathbb{O}}[\mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathsf{PSPACE}},\ell}(k)]$ *is overwhelming.*

Then, there is no fully-black-box construction of a OWP from an ideal TDP.

In order to use Lemma 1 for showing our main result, we define the set $\mathbb{T}$ of "TDP" oracles $\mathcal{T}$ below, which will be used as $\mathbb{O}$ in the above lemma. Next, in Section 3.1, we show Lemmas 2 and 3 which guarantee that there is a tuple of oracle PPTAs $\mathsf{TDP} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Samp}, \mathsf{Inv})$ such that $\mathbb{T}$ and TDP satisfy the conditions (1) and (2) of the above lemma, respectively. Then, in Section 3.2, we show Lemma 4 which guarantees that the set $\mathbb{T}$ satisfies the condition (3) of the above lemma. Theorem 1 follows by combining these lemmas.

[6] Actually, permutations in our TDP have a trivial domain $\{0,1\}^k$ and thus the TDP satisfies *doubly enhanced one-wayness* [18]. Furthermore, given a $2k$-bit string ek, whether $\mathcal{E}(ek, \cdot)$ defines a permutation can also be checked easily by checking the result of $\mathcal{E}(ek, 0^k)$, and thus it also satisfies the *certified* property [5]. Thus, our result also rules out constructions from a one-way TDP with these properties.

TDP Oracle $\mathcal{T}$. The *TDP oracle $\mathcal{T}$* models an ideally secure TDP whose evaluation-key space is sparse. Formally, a TDP oracle $\mathcal{T}$ consists of the following three suboracles $(\mathcal{G}, \mathcal{E}, \mathcal{D})$:

$\mathcal{G} : \{0,1\}^k \to \{0,1\}^{2k}$: (Corresponding to the key generation for the TDP) This is an injective function that takes $td \in \{0,1\}^k$ as input, and returns $ek \in \{0,1\}^{2k}$.

$\mathcal{E} : \{0,1\}^{2k} \times \{0,1\}^k \to \{0,1\}^k \cup \{\perp\}$: (Corresponding to evaluation) For every $ek \in \mathsf{Range}(\mathcal{G})$, $\mathcal{E}(ek, \cdot)$ is a permutation over $\{0,1\}^k$, and for every $ek \notin \mathsf{Range}(\mathcal{G})$ and every $\alpha \in \{0,1\}^k$, $\mathcal{E}(ek, \alpha) = \perp$.

$\mathcal{D} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^k$: (Corresponding to inversion) This function takes $td \in \{0,1\}^k$ and $\beta \in \{0,1\}^k$ as input, and returns $\alpha \in \{0,1\}^k$ such that $\mathcal{E}(\mathcal{G}(td), \alpha) = \beta$.

We denote by $\mathbb{T}$ the set consisting of all possible TDP oracles $\mathcal{T}$ that satisfy the above syntax.

3.1 Ideal Trapdoor Permutation Based on $\mathcal{T}$

Here, we show that there exists an ideal TDP that uses a TDP oracle $\mathcal{T} = (\mathcal{G}, \mathcal{E}, \mathcal{D}) \in \mathbb{T}$. Consider the following tuple $\mathsf{TDP}^{\mathcal{T}} = (\mathsf{Gen}^{\mathcal{T}}, \mathsf{Eval}^{\mathcal{T}}, \mathsf{Samp}^{\mathcal{T}}, \mathsf{Inv}^{\mathcal{T}})$ of oracle PPTAs, which are constructed straightforwardly from $\mathcal{T}$:

- $\mathsf{Gen}^{\mathcal{T}}(1^k, td)$: Compute $ek \leftarrow \mathcal{G}(td)$ and output the evaluation-key ek.
- $\mathsf{Eval}^{\mathcal{T}}(ek, x)$: Compute $y \leftarrow \mathcal{E}(ek, x)$ and output y. (We define the domain D_{ek} of $\mathsf{Eval}^{\mathcal{T}}(ek, \cdot)$ to be $\{0,1\}^k$ for all $ek \in \mathsf{Range}(\mathcal{G})$.)
- $\mathsf{Samp}^{\mathcal{T}}(ek)$: Pick $x \in \{0,1\}^k$ uniformly at random, and output x. (Note that this algorithm does not use $\mathcal{T}$ at all.)
- $\mathsf{Inv}^{\mathcal{T}}(td, y)$: Compute $x \leftarrow \mathcal{D}(td, y)$ and output x.

Regarding $\mathsf{TDP}^{\mathcal{T}}$ described above, the following two lemmas can be shown:

Lemma 2. *For any $\mathcal{T} \in \mathbb{T}$, $\mathsf{TDP}^{\mathcal{T}}$ is correct as a TDP.*

Lemma 3. *For all games G and any oracle PPTA adversary $\mathcal{A}$, there exists a negligible function $\mu(k)$ such that $\mathbf{E}_{\mathcal{T} \leftarrow_{\mathsf{R}} \mathbb{T}}[\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathcal{T}}, \mathcal{A}^{\mathcal{T}, \mathsf{PSPACE}}}(k)] - \delta(\mathsf{G}) \le \mu(k)$.*

Lemma 2 is immediate from the definition of the TDP oracle $\mathcal{T}$. The formal proof of Lemma 3 is given in the full version (but we will give a proof sketch below). Note that if we pick $\mathcal{T} = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ uniformly from $\mathbb{T}$, then $\mathcal{G}$ is a random injective function that is length-doubling, and every permutation $\mathcal{E}(ek, \cdot)$ with $ek \in \mathsf{Range}(\mathcal{G})$ is an independent random permutation. Kiltz and Pietrzak [29] showed that a similar construction of a TDP oracle whose "key generation oracle" is also a random permutation is ideally secure even against computationally unbounded adversary that makes only polynomially many queries. Our proof of Lemma 3 is similar to theirs.

Proof Sketch of Lemma 3. Fix an arbitrary δ-hard game G, and let $t = t(k)$ be a polynomial implicitly determined by G. Fix also an arbitrary PPTA adversary $\mathcal{A}$.

The expectation (over the choice of $\mathcal{T}$) of the advantage of the adversary $\mathcal{A}$ attacking $\mathsf{TDP}^{\mathcal{T}} = (\mathsf{Gen}^{\mathcal{T}}, \mathsf{Eval}^{\mathcal{T}}, \mathsf{Samp}^{\mathcal{T}}, \mathsf{Inv}^{\mathcal{T}})$ in the game G (in the presence of the PSPACE oracle) can be written as follows:

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\mathcal{T}\leftarrow_{\mathrm{R}}\mathbb{T}}\left[\ \mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathcal{T}},\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}}}(k)\ \right] \\
&= \mathop{\mathbf{E}}_{\mathcal{T}\leftarrow_{\mathrm{R}}\mathbb{T}}\left[\Pr\left[\begin{array}{c} td_1^*,\dots,td_t^* \leftarrow_{\mathrm{R}} \{0,1\}^k;\ ek_i^* \leftarrow \mathsf{Gen}^{\mathcal{T}}(1^k, td_i^*) \text{ for } i\in[t]; \\ d \leftarrow_{\mathrm{R}} \mathsf{Expt}^{\mathsf{G}^{\mathsf{Eval}^{\mathcal{T}}(ek_1^*,\cdot),\dots,\mathsf{Eval}^{\mathcal{T}}(ek_t^*,\cdot)}}_{\mathsf{TDP}^{\mathcal{T}},\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}}(ek_1^*,\dots,ek_t^*)}(k) \end{array} : d=1\right]\right] \\
&= \Pr\left[\begin{array}{c} \mathcal{T}\leftarrow_{\mathrm{R}}\mathbb{T};\ td_1^*,\dots,td_t^* \leftarrow_{\mathrm{R}} \{0,1\}^k;\ ek_i^* \leftarrow \mathcal{G}(td_i^*) \text{ for } i\in[t]; \\ d \leftarrow_{\mathrm{R}} \mathsf{Expt}^{\mathsf{G}^{\mathcal{E}(ek_1^*,\cdot),\dots,\mathcal{E}(ek_t^*,\cdot)}}_{\mathsf{TDP}^{\mathcal{T}},\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}}(ek_1^*,\dots,ek_t^*)}(k) \end{array} : d=1\right].
\end{aligned}$$

Let us denote by $\widetilde{\mathsf{Expt}}^{\mathsf{G}}_{\mathsf{TDP}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}}}(k)$ the experiment in the probability in the last equation.

Now, consider the following two games.

Game 1: This is the ordinary δ-hard game G for $\mathsf{TDP}^{\mathcal{T}}$, in which sampling of the oracle $\mathcal{T}$ from $\mathbb{T}$ is also taken into account, i.e. $\widetilde{\mathsf{Expt}}^{\mathsf{G}}_{\mathsf{TDP}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}}}(k)$.

Game 2: Same as Game 1, except that $\mathcal{A}$'s queries of the following types are answered with $\bot$: (i) a $\mathcal{G}$-query td_i^* for some $i \in [t]$, and (ii) a $\mathcal{D}$-query $(td_i^*, *)$ for some $i \in [t]$.

For $i \in \{1,2\}$, let Succ_i be the event that $\mathcal{A}$ wins (i.e. $d = 1$ occurs) in Game i. By definition we have $\mathbf{E}_{\mathcal{T}\leftarrow_{\mathrm{R}}\mathbb{T}}[\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathcal{T}},\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}}}(k)] = \Pr[\mathsf{Succ}_1]$. Furthermore, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{\mathcal{T}\leftarrow_{\mathrm{R}}\mathbb{T}}[\mathsf{Adv}^{\mathsf{G}}_{\mathsf{TDP}^{\mathcal{T}},\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}}}(k)] - \delta(\mathsf{G}) &= \Pr[\mathsf{Succ}_1] - \delta(\mathsf{G}) \\
&\le |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]| + \Pr[\mathsf{Succ}_2] - \delta(\mathsf{G}). \quad (1)
\end{aligned}$$

In the full version, we will show how to upperbound each term in the right hand side of the inequality (1), which will prove Lemma 3. Below we explain the sketches for how to show these.

$|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ can be shown to be negligible, because the adversary $\mathcal{A}$, who can make only polynomially many queries, cannot tell the difference between Game 1 and Game 2 (except with negligible probability). More specifically, Game 1 and Game 2 differ only in the response to $\mathcal{A}$'s $\mathcal{G}$-queries and $\mathcal{D}$-queries that contain the preimages $\{td_i^*\}_{i\in[t]}$ of the evaluation keys $\{ek_i^*\}_{i\in[t]}$, and thus in order for $\mathcal{A}$ to distinguish these games, $\mathcal{A}$ has to find one of $\{td_i^*\}_{i\in[t]}$. However, intuitively, finding any of the preimages $\{td_i^*\}_{i\in[t]}$ is hard because the TDP oracle $\mathcal{T}$ is chosen randomly and especially the function $\mathcal{G}$ is a random injective function, and we will formally show that this intuition works.

$\Pr[\mathsf{Succ}_2] - \delta(\mathsf{G})$ can be shown to be negligible, roughly because Game 2 can be perfectly simulated by another computationally unbounded adversary $\mathcal{S}$ with polynomial query complexity that interacts with the PPTA (challenger) G for *random permutations* (not for the TDP $\mathsf{TDP}^{\mathcal{T}}$), in such a way that $\mathsf{Adv}^{\mathsf{G}}_{\mathrm{RP},\mathcal{S}}(k) = \Pr[\mathsf{Succ}_2]$. But by the assumption that G is a δ-hard game, $\mathsf{Adv}^{\mathsf{G}}_{\mathrm{RP},\mathcal{S}}(k) - \delta(\mathsf{G}) = \Pr[\mathsf{Succ}_2] - \delta(\mathsf{G})$ is negligible.

This completes the proof sketch of Lemma 3. ☐

3.2 Breaking Any Candidate of One-Way Permutation Based on $\mathcal{T}$

Here, we show that any candidate of a OWP $\mathsf{P}^{\mathcal{T}}$ based on a TDP oracle $\mathcal{T} \in \mathbb{T}$ can be broken by some oracle PPTA almost perfectly (using the PSPACE oracle). Specifically, this subsection is devoted to proving the following lemma.

Lemma 4. *Let $\ell = \ell(k)$ be a positive polynomial and P be an oracle PPTA such that $\mathsf{P}^{\mathcal{T}}$ is a permutation over $\{0,1\}^{\ell}$ for all $\mathcal{T} \in \mathbb{T}$. Then there exists an oracle PPTA $\mathcal{A}$ such that $\mathbf{E}_{\mathcal{T} \leftarrow_{\mathtt{R}} \mathbb{T}}[\mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P}^{\mathcal{T}}, \mathcal{A}^{\mathcal{T}, \mathsf{PSPACE}}, \ell}(k)]$ is overwhelming.*

To prove Lemma 4, we need some further notations, two other oracles than $\mathcal{T}$, and several intermediate lemmas. Thus, we first introduce them, and in the last of this subsection show the proof of Lemma 4. The intuitive explanation on how the above lemma is proved can be found in Section 1.3.

Further Notations. For notational convenience, we introduce two notations. Let $\mathbb{O}$ be a set of oracles $\mathcal{O}$, $\ell = \ell(k)$ be a positive polynomial, and P and $\mathcal{A}$ be oracle PPTAs. If $\mathsf{P}^{\mathcal{O}}$ is a permutation over $\{0,1\}^{\ell}$ for all oracles $\mathcal{O} \in \mathbb{O}$, then we denote by $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{O}}, \mathcal{A}^{\mathbb{O}, \mathsf{PSPACE}}, \ell}(k)$ the following experiment:

$$[\ \mathcal{O} \leftarrow_{\mathtt{R}} \mathbb{O};\ x^* \leftarrow_{\mathtt{R}} \{0,1\}^{\ell};\ y^* \leftarrow \mathsf{P}^{\mathcal{O}}(x^*);\ x' \leftarrow_{\mathtt{R}} \mathcal{A}^{\mathcal{O}, \mathsf{PSPACE}}(1^k, y^*)\ ].$$

Note that $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{O}}, \mathcal{A}^{\mathbb{O}, \mathsf{PSPACE}}, \ell}(k)$ includes sampling an oracle $\mathcal{O}$ from $\mathbb{O}$.

Then, we define $\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{O}}, \mathcal{A}^{\mathbb{O}, \mathsf{PSPACE}}, \ell}(k) := \mathbf{E}_{\mathcal{O} \leftarrow_{\mathtt{R}} \mathbb{O}}[\ \mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P}^{\mathcal{O}}, \mathcal{A}^{\mathcal{O}, \mathsf{PSPACE}}, \ell}(k)\ ]$, i.e.,

$$\begin{aligned} &\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{O}}, \mathcal{A}^{\mathbb{O}, \mathsf{PSPACE}}, \ell}(k) \\ &= \Pr[\mathcal{O} \leftarrow_{\mathtt{R}} \mathbb{O}; x^* \leftarrow_{\mathtt{R}} \{0,1\}^{\ell}; y^* \leftarrow \mathsf{P}^{\mathcal{O}}(x^*); x' \leftarrow_{\mathtt{R}} \mathcal{A}^{\mathcal{O}, \mathsf{PSPACE}}(1^k, y^*) : x' = x^*]. \end{aligned}$$

(Our goal in this subsection is to show that $\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}}, \mathcal{A}^{\mathbb{T}, \mathsf{PSPACE}}, \ell}(k)$ is overwhelming.)

Block Cipher Oracle $\mathcal{B}$. Here we introduce a "block cipher" oracle $\mathcal{B}$ which models an ideally secure block cipher (or, keyed invertible permutation) whose key space is sparse. Formally, a block cipher oracle $\mathcal{B}$ consists of the following three suboracles $(\widehat{\mathcal{G}}, \mathcal{P}, \mathcal{P}^{-1})$:

$\widehat{\mathcal{G}} : \{0,1\}^k \to \{0,1\}^{2k}$: (Corresponding to the key generation for the block cipher) This is an injective function that takes $td \in \{0,1\}^k$ as input, and returns $ek \in \{0,1\}^{2k}$.

$\mathcal{P} : \{0,1\}^{2k} \times \{0,1\}^k \to \{0,1\}^k \cup \{\bot\}$: (Corresponding to encryption) For every $ek \in \mathsf{Range}(\widehat{\mathcal{G}})$, $\mathcal{P}(ek, \cdot)$ is a permutation over $\{0,1\}^k$, and for every $ek \notin \mathsf{Range}(\widehat{\mathcal{G}})$ and every $\alpha \in \{0,1\}^k$, $\mathcal{P}(ek, \alpha) = \bot$.

$\mathcal{P}^{-1} : \{0,1\}^{2k} \times \{0,1\}^k \to \{0,1\}^k \cup \{\bot\}$: (Corresponding to decryption) For every $ek \in \mathsf{Range}(\widehat{\mathcal{G}})$, $\mathcal{P}^{-1}(ek, \cdot)$ is the inversion of $\mathcal{P}(ek, \cdot)$, and for every $ek \notin \mathsf{Range}(\widehat{\mathcal{G}})$ and every $\beta \in \{0,1\}^k$, $\mathcal{P}^{-1}(ek, \beta) = \bot$.

We denote by $\mathbb{B}$ the set consisting of all possible block cipher oracles $\mathcal{B}$ that satisfy the above syntax.

Relationship between $\mathcal{T}$ and $\mathcal{B}$. We will use the following simple fact shown by Lindell and Zarosim [30].

Lemma 5. *([30]) Let ϕ be the mapping that maps a block cipher oracle $\mathcal{B} = (\widehat{\mathcal{G}}, \mathcal{P}, \mathcal{P}^{-1}) \in \mathbb{B}$ to a tuple of oracles $\phi(\mathcal{B}) = (\mathcal{G}, \mathcal{E}, \mathcal{D})$, where the suboracles $\mathcal{G}$, $\mathcal{E}$, and $\mathcal{D}$ are defined in the following way: For all $td \in \{0,1\}^k$, $ek \in \{0,1\}^{2k}$, $\alpha \in \{0,1\}^k$ and $\beta \in \{0,1\}^k$, we let*

$$\mathcal{G}(td) := \widehat{\mathcal{G}}(td), \quad \mathcal{E}(ek, \alpha) := \mathcal{P}(ek, \alpha), \quad \textit{and} \quad \mathcal{D}(td, \beta) := \mathcal{P}^{-1}(\widehat{\mathcal{G}}(td), \beta).$$

Then, ϕ is a bijection from $\mathbb{B}$ to $\mathbb{T}$.

Round Function Oracle $\mathcal{R}$. Here, we introduce a "round function" oracle $\mathcal{R}$ which models a set of "round functions" in the Feistel-network construction of permutations [32] (whose evaluation key space is sparse). Formally, a round function oracle $\mathcal{R}$ consists of the following two suboracles $(\widetilde{\mathcal{G}}, \mathcal{F})$:

$\widetilde{\mathcal{G}} : \{0,1\}^k \to \{0,1\}^{2k}$: (Corresponding to the key generation for each round function) This is an injective function that takes $td \in \{0,1\}^k$ as input, and returns $ek \in \{0,1\}^{2k}$.

$\mathcal{F} : [14] \times \{0,1\}^{2k} \times \{0,1\}^{k/2} \to \{0,1\}^{k/2} \cup \{\bot\}$: (Corresponding to the round functions in the Feistel-network). For every index $i \in [14]$ and $ek \in \mathsf{Range}(\widetilde{\mathcal{G}})$, $\mathcal{F}(i, ek, \cdot)$ is a function from $k/2$ bit to $k/2$ bit, and for every $ek \notin \mathsf{Range}(\widetilde{\mathcal{G}})$ and every $(i, \gamma) \in [14] \times \{0,1\}^{k/2}$, $\mathcal{F}(i, ek, \gamma) = \bot$.

We denote by $\mathbb{R}$ the set consisting of all possible round function oracles $\mathcal{R}$ that satisfy the above syntax.

Relationship between $\mathcal{B}$ and $\mathcal{R}$. Holenstein et al. [23] showed that the random oracle model and the ideal cipher model are equivalent. (The statement itself was posed by Coron et al. [10].) More concretely, they proved that a random invertible permutation can be simulated by the fourteen-round Feistel-network construction of a permutation in which each round function is an independent random function. (Technically, this means that the latter is *indifferentiable* [34] from the former.) Based on their result, we can also construct oracle PPTAs C and S such that $(\mathsf{C}^{\mathcal{R}}, \mathcal{R})$ and $(\mathcal{B}, \mathsf{S}^{\mathcal{B}})$ are indistinguishable.

More formally, consider the following PPTA C that, given access to $\mathcal{R} = (\widetilde{\mathcal{G}}, \mathcal{F}) \in \mathbb{R}$, tries to simulate a block cipher oracle $\mathsf{C}^{\mathcal{R}} = (\widehat{\mathcal{G}}, \mathcal{P}, \mathcal{P}^{-1})$ as follows:

$\widehat{\mathcal{G}}(\cdot)$: Define $\widehat{\mathcal{G}}(\cdot) = \widetilde{\mathcal{G}}(\cdot)$.

$\mathcal{P}(\cdot,\cdot)$: On input $(ek, \alpha) \in \{0,1\}^{2k} \times \{0,1\}^k$, check if $ek \in \mathsf{Range}(\widetilde{\mathcal{G}})$ by making an $\mathcal{F}$-query $(1, ek, 0^{k/2})$. If the answer from $\mathcal{F}$ is $\bot$ (meaning $ek \notin \mathsf{Range}(\widetilde{\mathcal{G}})$), then return $\bot$. Otherwise, regard α as $\alpha = (L_0||R_0)$ so that $|L_0| = |R_0| = k/2$. Then, for each $i \in [14]$, compute $L_i \leftarrow R_{i-1}$ and $R_i \leftarrow \mathcal{F}(i, ek, R_{i-1}) \oplus L_{i-1}$, and finally output $\beta \leftarrow (L_{14}||R_{14})$.

$\mathcal{P}^{-1}(\cdot,\cdot)$: On input $(ek, \beta) \in \{0,1\}^{2k} \times \{0,1\}^k$, check if $ek \in \mathsf{Range}(\widetilde{\mathcal{G}})$ as above. If $ek \notin \mathsf{Range}(\widetilde{\mathcal{G}})$, then return $\bot$. Otherwise, compute and output the inversion of $\mathcal{P}(ek, \cdot)$ using $\mathcal{F}$.

Constructed as above, it is guaranteed that $\mathsf{C}^{\mathcal{R}} \in \mathbb{B}$ for all $\mathcal{R} \in \mathbb{R}$, because the Feistel-network construction yields a permutation no matter what round functions are used. Moreover, the result in [23] yields the following.

Lemma 6. *(follows from [23].) Let* C *be the oracle PPTA as above. Then, for any polynomial* $q = q(k)$*, there exists an oracle PPTA* S *such that for all (computationally unbounded) oracle algorithms* $\mathcal{D}$ *making at most* q *queries, the following difference is negligible:*

$$|\Pr_{\mathcal{R} \leftarrow_{\mathrm{R}} \mathbb{R}}[\mathcal{D}^{\mathsf{C}^{\mathcal{R}},\mathcal{R}}(1^k) = 1] - \Pr_{\mathcal{B} \leftarrow_{\mathrm{R}} \mathbb{B}}[\mathcal{D}^{\mathcal{B},\mathsf{S}^{\mathcal{B}}}(1^k) = 1]|.$$

TDP Oracle $\mathcal{T}$ *Can Be Simulated.* Here, we show that if there exists a TDP-based permutation $\mathsf{P}^{\mathcal{T}}$, then so does a "random function"-based permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$ such that inverting $\widetilde{\mathsf{P}}^{\mathcal{R}}$ is as hard as inverting $\mathsf{P}^{\mathcal{T}}$. Furthermore, the latter is true even in the presence of PSPACE oracle.

Lemma 7. *Let* $\ell = \ell(k)$ *be a positive polynomial and* P *be an oracle PPTA such that* $\mathsf{P}^{\mathcal{T}}$ *is a permutation over* $\{0,1\}^{\ell}$ *for all* $\mathcal{T} \in \mathbb{T}$*. Then, there exists another oracle PPTA* $\widetilde{\mathsf{P}}$ *that satisfies the following two properties: (1) For all* $\mathcal{R} \in \mathbb{R}$*,* $\widetilde{\mathsf{P}}^{\mathcal{R}} \in \mathsf{Perm}_{\ell}$*. (2) For any oracle PPTA* $\widetilde{\mathcal{A}}$*, there exist another oracle PPTA* $\mathcal{A}$ *and a negligible function* $\mu(k)$ *such that* $\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}},\ell}(k) \geq \widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k) - \mu(k)$*.*

Proof of Lemma 7. (The intuitive explanation can be found in Section 1.3.) Let ℓ and P be as stated in the lemma. First, define the "intermediate" oracle PPTA $\widehat{\mathsf{P}}$ by $\widehat{\mathsf{P}}^{\mathcal{B}}(\cdot) = \mathsf{P}^{\phi(\mathcal{B})}(\cdot)$, where ϕ is the bijection from $\mathbb{B}$ to $\mathbb{T}$ due to Lemma 5. This construction of $\widehat{\mathsf{P}}$ also guarantees that $\mathsf{P}^{\mathcal{T}}(\cdot) = \widehat{\mathsf{P}}^{\phi^{-1}(\mathcal{T})}(\cdot)$ where ϕ^{-1} is the inversion function of ϕ (i.e. ϕ^{-1} is also a bijection from $\mathbb{T}$ to $\mathbb{B}$). Next, define the oracle PPTA $\widetilde{\mathsf{P}}$ by $\widetilde{\mathsf{P}}^{\mathcal{R}}(\cdot) = \widehat{\mathsf{P}}^{\mathsf{C}^{\mathcal{R}}}(\cdot)$, where C is the oracle PPTA due to Lemma 6. Then, since $\mathsf{P}^{\mathcal{T}} \in \mathsf{Perm}_{\ell}$ for all $\mathcal{T} \in \mathbb{T}$, we have $\widehat{\mathsf{P}}^{\mathcal{B}} \in \mathsf{Perm}_{\ell}$ for all $\mathcal{B} \in \mathbb{B}$. This in turn guarantees that $\widetilde{\mathsf{P}}^{\mathcal{R}} \in \mathsf{Perm}_{\ell}$ for all $\mathcal{R} \in \mathbb{R}$, because $\mathsf{C}^{\mathcal{R}} \in \mathbb{B}$ for all $\mathcal{R} \in \mathbb{R}$. Therefore, $\widetilde{\mathsf{P}}$ satisfies the property (1).

Next, we show that $\widetilde{\mathsf{P}}$ satisfies the property (2). Let $\widetilde{\mathcal{A}}$ be an arbitrary oracle PPTA adversary that runs in the experiment $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k)$ and makes in total $q = q(k)$ oracle queries. Note that since $\widetilde{\mathcal{A}}$ is a PPTA, q is a polynomial. Let S be the simulator corresponding to the polynomial q, which is guaranteed to exist by Lemma 6, and define an oracle PPTA $\widehat{\mathcal{A}}^{(\cdot),(\cdot)}$ (which expects to have access to an oracle $\mathcal{B} \in \mathbb{B}$ and the PSPACE oracle) by $\widetilde{\mathcal{A}}^{\mathsf{S}^{(\cdot)},(\cdot)}$. That is, given access to any $\mathcal{B} \in \mathbb{B}$ and the PSPACE oracle, $\widehat{\mathcal{A}}^{\mathcal{B},\mathsf{PSPACE}}$ and $\widetilde{\mathcal{A}}^{\mathsf{S}^{\mathcal{B}},\mathsf{PSPACE}}$ behave identically. Since both $\widetilde{\mathcal{A}}$ and S are oracle PPTAs, $\widehat{\mathcal{A}}$ is also an oracle PPTA and thus makes at most polynomially many queries.

Then, consider the following sequence of games.

Game 1 This is the ordinary experiment $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k)$ that $\widetilde{\mathcal{A}}$ runs in. That is:

$$[\mathcal{R} \leftarrow_{\mathrm{R}} \mathbb{R};\ x^* \leftarrow_{\mathrm{R}} \{0,1\}^{\ell};\ y^* \leftarrow \widetilde{\mathsf{P}}^{\mathcal{R}}(x^*);\ x' \leftarrow_{\mathrm{R}} \widetilde{\mathcal{A}}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y^*)].$$

Game 2 This game is defined as follows:

$$[\mathcal{B} \leftarrow_{\mathrm{R}} \mathbb{B};\ x^* \leftarrow_{\mathrm{R}} \{0,1\}^{\ell};\ y^* \leftarrow \widehat{\mathsf{P}}^{\mathcal{B}}(x^*);\ x' \leftarrow_{\mathrm{R}} \widehat{\mathcal{A}}^{\mathcal{B},\mathsf{PSPACE}}(1^k, y^*)].$$

Game 3 This game is defined as follows:
$[\mathcal{T} \leftarrow_{\mathtt{R}} \mathbb{T};\ x^* \leftarrow_{\mathtt{R}} \{0,1\}^\ell;\ y^* \leftarrow \mathsf{P}^{\mathcal{T}}(x^*);\ x' \leftarrow_{\mathtt{R}} \widehat{\mathcal{A}}^{\phi^{-1}(\mathcal{T}),\mathsf{PSPACE}}(1^k, y^*)]$.

Game 4 Same as Game 3, except that when $\widehat{\mathcal{A}}$ makes a $\mathcal{P}$-query (ek, α) or a $\mathcal{P}^{-1}$-query (ek, β) such that ek is not an answer to some of $\widehat{\mathcal{A}}$'s previous $\widehat{\mathcal{G}}$-queries, the query is answered with $\bot$.

For $i \in [4]$, let Succ_i be the event that $x' = x^*$ occurs in Game i. Then we have

$$\widetilde{\mathsf{Adv}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k) = \Pr[\mathsf{Succ}_1] \le \sum_{i\in[3]} |\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]| + \Pr[\mathsf{Succ}_4]. \quad (2)$$

To complete the proof, we upperbound each term in the above inequality.

Claim 1 $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ *is negligible.*

Proof of Claim 1. We show that we can construct a *computationally unbounded* oracle algorithm (distinguisher) $\mathcal{D}$ that, using $\widehat{\mathsf{P}}$ and $\widetilde{\mathcal{A}}$ as its subroutines, makes at most q queries, and satisfies

$$|\Pr_{\mathcal{R}\leftarrow_{\mathtt{R}}\mathbb{R}}[\mathcal{D}^{\mathsf{C}^{\mathcal{R}},\mathcal{R}}(1^k) = 1] - \Pr_{\mathcal{B}\leftarrow_{\mathtt{R}}\mathbb{B}}[\mathcal{D}^{\mathcal{B},\mathsf{S}^{\mathcal{B}}}(1^k) = 1]| = |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|. \quad (3)$$

$\mathcal{D}$ is given access to two oracles $(\mathcal{O}_1, \mathcal{O}_2)$, which is either $(\mathsf{C}^{\mathcal{R}}, \mathcal{R})$ or $(\mathcal{B}, \mathsf{S}^{\mathcal{B}})$, and runs as follows:

$\mathcal{D}^{\mathcal{O}_1,\mathcal{O}_2}(1^k)$**:** $\mathcal{D}$ picks $x^* \leftarrow_{\mathtt{R}} \{0,1\}^\ell$, computes $y^* \leftarrow \widehat{\mathsf{P}}^{\mathcal{O}_1}(x^*)$, and then simulates $\widetilde{\mathcal{A}}^{\mathcal{O}_2,\mathsf{PSPACE}}(1^k, y^*)$. Note that $\mathcal{D}$ is computationally unbounded, and thus can simulate the PSPACE oracle perfectly for $\widetilde{\mathcal{A}}$.
When $\widetilde{\mathcal{A}}$ terminates with output x', $\mathcal{D}$ checks whether $x' = x^*$. If this is the case, then $\mathcal{D}$ outputs 1, otherwise outputs 0, and terminates.

The above completes the description of $\mathcal{D}$. Note that the number of queries that $\mathcal{D}$ makes is at most the number of queries made by $\widetilde{\mathcal{A}}$, and thus is at most q.

Now, consider the case when $(\mathcal{O}_1, \mathcal{O}_2) = (\mathsf{C}^{\mathcal{R}}, \mathcal{R})$. Then it is clear that $\mathcal{D}$ simulates Game 1 perfectly for $\widetilde{\mathcal{A}}$. In particular, in this case we have $\widehat{\mathsf{P}}^{\mathcal{O}_1}(x^*) = \widehat{\mathsf{P}}^{\mathsf{C}^{\mathcal{R}}}(x^*) = \widetilde{\mathsf{P}}^{\mathcal{R}}(x^*)$, and $\widetilde{\mathcal{A}}$ is given access to $\mathcal{O}_2 = \mathcal{R}$ and PSPACE as in Game 1. Under this situation, the probability that $\mathcal{D}$ outputs 1 is exactly the same as the probability that $\widetilde{\mathcal{A}}$ succeeds in outputting the preimage x^* under $\widetilde{\mathsf{P}}^{\mathcal{R}}$ in Game 1, i.e. $\Pr_{\mathcal{R}\leftarrow_{\mathtt{R}}\mathbb{R}}[\mathcal{D}^{\mathsf{C}^{\mathcal{R}},\mathcal{R}}(1^k) = 1] = \Pr[\mathsf{Succ}_1]$.

Next, consider the case when $(\mathcal{O}_1, \mathcal{O}_2) = (\mathcal{B}, \mathsf{S}^{\mathcal{B}})$. Recall that we defined $\widehat{\mathcal{A}}^{\mathcal{B},\mathsf{PSPACE}}$ by $\widetilde{\mathcal{A}}^{\mathsf{S}^{\mathcal{B}},\mathsf{PSPACE}}$, and thus $\widetilde{\mathcal{A}}^{\mathcal{O}_2,\mathsf{PSPACE}} = \widetilde{\mathcal{A}}^{\mathsf{S}^{\mathcal{B}},\mathsf{PSPACE}} = \widehat{\mathcal{A}}^{\mathcal{B},\mathsf{PSPACE}}$. Recall also that $\mathcal{D}$ can simulate PSPACE perfectly by its computationally unbounded power. Therefore, in this case $\mathcal{D}$ perfectly simulates Game 2 for $\widehat{\mathcal{A}}$. In particular, in this case we have $\widehat{\mathsf{P}}^{\mathcal{O}_1}(x^*) = \widehat{\mathsf{P}}^{\mathcal{B}}(x^*)$, and $\widehat{\mathcal{A}}$'s oracle queries are perfectly answered as in Game 2, using $\mathcal{O}_1 = \mathcal{B}$ and $\mathcal{D}$'s computationally unbounded power. Therefore the probability that $\mathcal{D}$ outputs 1 is exactly the same as the probability that $\widehat{\mathcal{A}}$ outputs x^* in Game 2, i.e. $\Pr_{\mathcal{B}\leftarrow_{\mathtt{R}}\mathbb{B}}[\mathcal{D}^{\mathcal{B},\mathsf{S}^{\mathcal{B}}}(1^k) = 1] = \Pr[\mathsf{Succ}_2]$.

In summary, our distinguisher $\mathcal{D}$ makes in total q queries and satisfies the equation (3). Thus, Lemma 6 guarantees that $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ is upperbounded to be negligible. This completes the proof of Claim 1. $\square$

Claim 2 $\Pr[\mathsf{Succ}_2] = \Pr[\mathsf{Succ}_3]$.

Proof of Claim 2. Recall that due to Lemma 5, ϕ (and thus ϕ^{-1}) is a bijection between $\mathbb{B}$ and $\mathbb{T}$. Therefore, the uniform distribution over $\mathbb{B}$ is equivalent to the distribution of $\phi^{-1}(\mathcal{T})$ when $\mathcal{T} \leftarrow_{\mathtt{R}} \mathbb{T}$. Moreover, $\mathsf{P}^{\mathcal{T}}(\cdot) = \widehat{\mathsf{P}}^{\phi^{-1}(\mathcal{T})}(\cdot)$ for all $\mathcal{T} \in \mathbb{T}$ by definition. These imply that from $\widehat{\mathcal{A}}$'s view point, all values in Game 2 and those in Game 3 are distributed identically, and thus $\Pr[\mathsf{Succ}_2] = \Pr[\mathsf{Succ}_3]$. This completes the proof of Claim 2. $\square$

Claim 3 $|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]|$ *is negligible.*

Proof Sketch of Claim 3. For $i \in \{3, 4\}$, let Find_i be the event that in Game i, $\widehat{\mathcal{A}}$ makes at least one $\mathcal{P}$- or $\mathcal{P}^{-1}$-query such that ek is not an answer to some of previous $\widehat{\mathcal{A}}$'s $\widehat{\mathcal{G}}$-queries and $ek \in \mathsf{Range}(\mathcal{G})$. Note that Game 3 and Game 4 proceed identically until Find_3 or Find_4 occurs in the corresponding games. Therefore, we have

$$|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]| \le \Pr[\mathsf{Find}_3] = \Pr[\mathsf{Find}_4].$$

Hence, to prove the claim it is sufficient to bound $\Pr[\mathsf{Find}_4]$.

Recall that in Game 4 (and in Game 3) the oracle $\mathcal{T} \in \mathbb{T}$ is picked uniformly, and thus $\mathcal{G}$ oracle is a random injective function which is length-doubling. Therefore, the probability that Find_4 occurs is exactly the same as the probability that an oracle algorithm with polynomial query complexity, which is given access to a random length-doubling injective function and the corresponding "membership" function for its range (this membership function tells if a given value is in the range of the injective function), finds a "fresh" element that is not obtained by actually making a query to the function but belongs to its range. However, it is easy to prove that such a probability is negligible (as long as the query complexity of the algorithm is at most polynomial), and this in turn bounds $\Pr[\mathsf{Find}_4]$ to be negligible. (The formal proof is provided in the full version.) This completes the proof sketch of Claim 3. $\square$

Claim 4 *There exists an oracle PPTA $\mathcal{A}$ such that* $\Pr[\mathsf{Succ}_4] = \widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}},\ell}(k)$.

Proof of Claim 4. Using the oracle PPTA $\widehat{\mathcal{A}}$ as a building block, we construct an oracle PPTA $\mathcal{A}$ that runs in $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}},\ell}(k)$: $\mathcal{A}$ is given $(1^k, y^*)$ as input, where $y^* = \mathsf{P}^{\mathcal{T}}(x^*)$ for a randomly chosen $x^* \in \{0,1\}^{\ell}$ and $\mathcal{T} \in \mathbb{T}$, given access to $\mathcal{T}$ and PSPACE, and runs as follows:

$\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}}(1^k, y^*)$: $\mathcal{A}$ generates an empty list L used to store "known" $\mathcal{G}$-query/answer pairs, and then runs $\widehat{\mathcal{A}}(1^k, y^*)$.

$\mathcal{A}$ responds to the queries from $\widehat{\mathcal{A}}$ as follows:

- For a $\widehat{\mathcal{G}}$-query td, $\mathcal{A}$ forwards it to $\mathcal{G}$, receives ek from $\mathcal{G}$, and returns this ek to $\widehat{\mathcal{A}}$. $\mathcal{A}$ also stores the pair (td, ek) into the list L.

– For a $\mathcal{P}$-query (ek, α), if there is no entry of the form $(*, ek)$ in L, then $\mathcal{A}$ responds with $\perp$. Otherwise, $\mathcal{A}$ makes a $\mathcal{E}$-query (ek, α), receives β from $\mathcal{E}$, and finally returns this β to $\widehat{\mathcal{A}}$.
– For a $\mathcal{P}^{-1}$-query (ek, β), if there is no entry of the form $(*, ek)$ in L, then $\mathcal{A}$ responds with $\perp$. Otherwise, $\mathcal{A}$ retrieves td that corresponds to ek from L, makes a $\mathcal{D}$-query (td, β), receives α from $\mathcal{D}$, and finally returns this α to $\widehat{\mathcal{A}}$.
– For a PSPACE-query, $\mathcal{A}$ answers to it by using $\mathcal{A}$'s own PSPACE oracle.

When $\widehat{\mathcal{A}}$ terminates with output x', $\mathcal{A}$ also terminates with output this x'.

It is easy to see that $\mathcal{A}$ perfectly simulates Game 4 for $\widehat{\mathcal{A}}$ in which the oracles given access to $\widehat{\mathcal{A}}$ are $\phi^{-1}(\mathcal{T})$ (that works as specified in Game 4) and PSPACE. Under this situation, when $\widehat{\mathcal{A}}$ succeeds in outputting the value x^* such that $\widehat{\mathsf{P}}^{\phi^{-1}(\mathcal{T})}(x^*) = y^*$, since $\mathsf{P}^{\mathcal{T}}(\cdot) = \widehat{\mathsf{P}}^{\phi^{-1}(\mathcal{T})}(\cdot)$ for all $\mathcal{T} \in \mathbb{T}$ by definition, $\mathcal{A}$ also succeeds in outputting the preimage under $\mathsf{P}^{\mathcal{T}}$. Therefore, we have $\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}}, \mathcal{A}^{\mathbb{T}, \mathsf{PSPACE}}}(k) = \Pr[\mathsf{Succ}_4]$. This completes the proof of Claim 4. $\square$

Claims 1 to 4 imply that for any oracle PPTA $\widetilde{\mathcal{A}}$, there exist an oracle PPTA $\mathcal{A}$ and a negligible function $\mu(k)$ such that $\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}}, \mathcal{A}^{\mathbb{T}, \mathsf{PSPACE}}, \ell}(k) \geq \widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}}, \widetilde{\mathcal{A}}^{\mathbb{R}, \mathsf{PSPACE}}, \ell}(k) - \mu(k)$, and thus the property (2) is satisfied as well. This completes the proof of Lemma 7. $\square$

"Mimicking" Algorithm N *and Good Inverter* Q *for* N. The combination of the results by Rudich [43] and Kahn et al. [26] shows that any permutation which has oracle access to a set of random functions can be inverted using the PSPACE oracle. On the other hand, Lemma 7 shows that for any TDP-based permutation $\mathsf{P}^{\mathcal{T}}$, there is another "random function"-based permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$ such that if $\widetilde{\mathsf{P}}^{\mathcal{R}}$ can be inverted using the PSPACE oracle, then so can be $\mathsf{P}^{\mathcal{T}}$. Here, it seems that by combining the results [43, 26] and Lemma 7 we can invert the "random function"-based permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$ using the PSPACE oracle. However, there is a subtle issue here: The suboracle $\mathcal{F}$ in a round function oracle $\mathcal{R}$ is not a pure random function, even if $\mathcal{R}$ is sampled randomly from the set $\mathbb{R}$. Specifically, $\mathcal{F}$ returns an "invalid" symbol $\perp$ for some inputs, and thus we cannot directly use the results [43, 26].

For convenience, let us refer to a query to the suboracle $\mathcal{F}$ in a round function oracle $\mathcal{R} \in \mathbb{R}$ as *invalid* if the answer to the query is $\perp$, and an oracle algorithm N that expects to access to an oracle $\mathcal{R} \in \mathbb{R}$ as *legal* if $\mathsf{N}^{\mathcal{R}}$ never makes an invalid query for all $\mathcal{R} \in \mathbb{R}$ and for all inputs.

To resolve the subtlety on invalid queries, we will use the approach by Chang et al. [9]: we show two lemmas that enable us to finally show that a TDP-based permutation can be inverted almost perfectly. The first lemma below (Lemma 8) roughly states that for a permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$ based on a round function oracle $\mathcal{R}$, there is a "mimicking" algorithm $\mathsf{N}^{\mathcal{R}}$ which is legal and, for most inputs, computes almost the same result as $\widetilde{\mathsf{P}}^{\mathcal{R}}$ for most oracles $\mathcal{R} \in \mathbb{R}$.

Lemma 8. *Let $\ell = \ell(k) > 0$ be a polynomial and $\widetilde{\mathsf{P}}$ be an oracle PPTA such that $\widetilde{\mathsf{P}}^{\mathcal{R}}$ is a permutation over $\{0,1\}^{\ell}$ for all $\mathcal{R} \in \mathbb{R}$. Then, there exists an oracle PPTA N (that expects to access to an oracle from $\mathbb{R}$) with the following properties: (i) N is legal, and (ii) For sufficiently large k's, for at least $1 - 2 \cdot 2^{-k/6}$ fraction of strings $y \in \{0,1\}^{\ell}$, $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = (\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y)$ holds for at least $1 - 2^{-k/3}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$.*

The formal proof proceeds closely to that of [9, Claim 3 and Lemma 3], and is given in the full version. We give a proof sketch.

Proof Sketch of Lemma 8. Let ℓ and $\widetilde{\mathsf{P}}$ be as stated in the lemma. Using $\widetilde{\mathsf{P}}$ as a subroutine, we construct the oracle PPTA N that satisfies the properties (i) and (ii). N takes a string $x \in \{0,1\}^\ell$ as input, has access to an oracle $\mathcal{R} \in \mathbb{R}$, and runs as follows:

$\mathsf{N}^{\mathcal{R}}(x)$: N firstly generates an empty list L into which "known" evaluation-keys $ek \in \mathsf{Range}(\widetilde{\mathcal{G}})$ will be stored, and then runs $\widetilde{\mathsf{P}}(x)$. N responds to queries from $\widetilde{\mathsf{P}}$ as follows:

- When $\widetilde{\mathsf{P}}$ makes a $\widetilde{\mathcal{G}}$-query $td \in \{0,1\}^k$, N forwards it to $\widetilde{\mathcal{G}}$, receives a result ek from $\widetilde{\mathcal{G}}$, and returns this ek to $\widetilde{\mathsf{P}}$. N also stores ek into the list L.
- When $\widetilde{\mathsf{P}}$ makes a $\mathcal{F}$-query $(i, ek, \gamma) \in [14] \times \{0,1\}^{2k} \times \{0,1\}^{k/2}$, N responds with $\perp$ if $ek \notin L$. Otherwise, N forwards (i, ek, γ) to $\mathcal{F}$, receives an answer $\delta \in \{0,1\}^{k/2}$ from $\mathcal{F}$, and returns δ to $\widetilde{\mathsf{P}}$.

When $\widetilde{\mathsf{P}}$ terminates with output y, N also terminates with output y.

The above completes the description of N. Note that N is legal, because N's $\mathcal{F}$-queries always satisfy $ek \in \mathsf{Range}(\widetilde{\mathcal{G}})$. Hence, the property (i) is satisfied.

To show that the above N satisfies the property (ii), we will show the following two claims that together imply what we want (the formal proofs are given in the full version), and hence enable us to complete the proof of Lemma 8:

Claim 5 *For any string $x \in \{0,1\}^\ell$, $\Pr_{\mathcal{R} \leftarrow_{\mathsf{R}} \mathbb{R}}[\mathsf{N}^{\mathcal{R}}(x) \neq \widetilde{\mathsf{P}}^{\mathcal{R}}(x)] \leq 2^{-k/2}$ holds for sufficiently large k's.*

Claim 6 *For sufficiently large k's, the following holds. There are at least $1 - 2 \cdot 2^{-k/6}$ fraction of strings $y \in \{0,1\}^\ell$ such that $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = (\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y)$ holds for at least $1 - 2^{-k/3}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$.*

Claim 5 can be shown in a similar manner to the negligible upperbound of $\Pr[\mathsf{Find}_4]$ in the proof of Claim 3. Specifically, it is clear from the description of N that for any input $x \in \{0,1\}^\ell$, the output of N and that of $\widetilde{\mathsf{P}}$ agree unless $\widetilde{\mathsf{P}}$ makes a $\mathcal{F}$-query $(*, ek, *)$ such that ek is not an answer to $\widetilde{\mathsf{P}}$'s previous $\widetilde{\mathcal{G}}$-queries. Therefore, "$\mathsf{N}^{\mathcal{R}}(x) \neq \widetilde{\mathsf{P}}^{\mathcal{R}}(x)$" must mean that $\widetilde{\mathsf{P}}$ makes such a $\mathcal{F}$-query. However, if $\mathcal{R}$ is chosen uniformly, $\widetilde{\mathcal{G}}$ is a random length-doubling injective function, and thus the probability of $\widetilde{\mathsf{P}}$ finding a "fresh" element that belongs to $\mathsf{Range}(\widetilde{\mathcal{G}})$ is exponentially small. (Here, $\mathcal{F}$ works as the "membership" oracle regarding the range of $\widetilde{\mathcal{G}}$, but it does not help much.)

For showing Claim 6, consider the Boolean matrix $M = \left(M_{(y,\mathcal{R})}\right)$ whose rows are indexed by $y \in \{0,1\}^\ell$ and whose columns are indexed by $\mathcal{R} \in \mathbb{R}$, so that $M_{(y,\mathcal{R})} = 1$ if and only if $(\mathsf{N}^{\mathcal{R}})^{-1}(y) \neq (\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y)$. By Claim 5, we know that for sufficiently large k's, we have that for each $x \in \{0,1\}^\ell$, $\mathsf{N}^{\mathcal{R}}(x) \neq \widetilde{\mathsf{P}}^{\mathcal{R}}(x)$ holds for at most $2^{-k/2}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$. Since any such pair $(x, \mathcal{R})$ contributes at most two 1's to the matrix M (namely, to the entries $M_{(\mathsf{N}^{\mathcal{R}}(x),\mathcal{R})}$ and $M_{(\widetilde{\mathsf{P}}^{\mathcal{R}}(x),\mathcal{R})}$), the total fraction of 1's in M is at most $2 \cdot 2^{-k/2}$. That is, $\Pr_{y \leftarrow_{\mathsf{R}} \{0,1\}^\ell, \mathcal{R} \leftarrow_{\mathsf{R}} \mathbb{R}}[M_{(y,\mathcal{R})} = 1] \leq 2 \cdot 2^{-k/2}$. Then, a simple counting argument yields Claim 6.

This completes the proof sketch of Lemma 8. □

We note that even if $\widetilde{\mathsf{P}}^{\mathcal{R}}$ is a permutation, $\mathsf{N}^{\mathcal{R}}$ in Lemma 8 is not guaranteed to be a permutation (although $\mathsf{N}^{\mathcal{R}}$ is very close to a permutation), and this is the main reason why we cannot directly use the results from [43, 26]. A similar situation was encountered in [9] where the authors could not directly apply the results from [43, 26] to show the separation of a OWP from a trapdoor function.

Fortunately, we can use the next lemma, which is implied by the one shown and used in [9, Section 3.2] (which is in turn based on [43, 26]). The following lemma roughly says that if most of the images under a legal oracle algorithm $\mathsf{N}^{\mathcal{R}}$ have a unique preimage, (and in particular these properties are satisfied by the algorithm $\mathsf{N}^{\mathcal{R}}$ in Lemma 8), then there is an oracle algorithm $\mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}$ that can invert $\mathsf{N}^{\mathcal{R}}$ almost always, using the PSPACE oracle.

Lemma 9. *(follows from [9, Lemma 4].) Let $\ell = \ell(k)$ be a positive polynomial. There exists a constant $\lambda > 0$ such that for every legal oracle PPTA $\mathsf{N}^{(\cdot)} : \{0,1\}^\ell \to \{0,1\}^\ell$ (that expects to access to an oracle from $\mathbb{R}$), there is another oracle PPTA Q with the following property: For any $\epsilon < \lambda$ and any $y \in \{0,1\}^\ell$, if the size of the set $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = \{x \in \{0,1\}^\ell | \mathsf{N}^{\mathcal{R}}(x) = y\}$ is one for $1-\epsilon$ fraction of oracles $\mathcal{R} \in \mathbb{R}$, then $\mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y) = (\mathsf{N}^{\mathcal{R}})^{-1}(y)$ holds for $1-\sqrt{\epsilon}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$.*

Inverting Any Permutation Based on $\mathcal{T}$: Proof of Lemma 4. Now, we are ready to prove Lemma 4. Let ℓ and P be as stated in the lemma. By lemma 7, for this P, there is an oracle PPTA $\widetilde{\mathsf{P}}$ such that $\widetilde{\mathsf{P}}^{\mathcal{R}} \in \mathsf{Perm}_\ell$ for all $\mathcal{R} \in \mathbb{R}$. Then, Lemma 8 tells us that for this $\widetilde{\mathsf{P}}$, there exists an oracle PPTA N that satisfies the properties (i) and (ii). Since $\widetilde{\mathsf{P}}^{\mathcal{R}}$ is a permutation for all $\mathcal{R} \in \mathbb{R}$, the size of the set $(\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y) = \{x \in \{0,1\}^\ell | \widetilde{\mathsf{P}}^{\mathcal{R}}(x) = y\}$ is one for all $y \in \{0,1\}^\ell$ and all $\mathcal{R} \in \mathbb{R}$. Thus, if $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = (\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y)$, the size of the set $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = \{x \in \{0,1\}^\ell | \mathsf{N}^{\mathcal{R}}(x) = y\}$ must also be one. By the property (ii) of N in Lemma 8, for at least $1 - 2 \cdot 2^{-k/6}$ fraction of strings $y \in \{0,1\}^\ell$, the size of the set $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = \{x \in \{0,1\}^\ell | \mathsf{N}^{\mathcal{R}}(x) = y\}$ is one for at least $1 - 2^{-k/3}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$.

Set $\epsilon' = 2^{-k/3}$. For any constant $\lambda > 0$, $\epsilon' < \lambda$ holds for all sufficiently large k's, and thus this ϵ' can be used as the ϵ in Lemma 9. Call $y \in \{0,1\}^\ell$ *good* if $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = (\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y)$ holds for $1 - 2^{-k/3}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$. By definition, if y is good, then it is guaranteed that the size of the set $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = \{x \in \{0,1\}^\ell | \mathsf{N}^{\mathcal{R}}(x) = y\}$ is one for at least $1 - \epsilon' = 1 - 2^{-k/3}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$, and it is also guaranteed that $\Pr_{y \leftarrow_{\mathtt{R}} \{0,1\}^\ell}[y \text{ is good}] \geq 1 - 2 \cdot 2^{-k/6}$ holds. Furthermore, by using N and ϵ', Lemma 9 implies that there is an oracle PPTA Q such that for sufficiently large k's and for all good y's, $\mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(y) = (\mathsf{N}^{\mathcal{R}})^{-1}(y)$ holds for $1 - \sqrt{\epsilon'}$ fraction of oracles $\mathcal{R} \in \mathbb{R}$. Recall that for $y \in \{0,1\}^\ell$ and $\mathcal{R} \in \mathbb{R}$ such that $(\mathsf{N}^{\mathcal{R}})^{-1}(y) = (\widetilde{\mathsf{P}}^{\mathcal{R}})^{-1}(y)$ and $\mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y) = (\mathsf{N}^{\mathcal{R}})^{-1}(y) = x$, it holds that $\widetilde{\mathsf{P}}^{\mathcal{R}}(x) = y$, i.e. Q succeeds in calculating the preimage x of y under the permutation $\widetilde{\mathsf{P}}^{\mathcal{R}}$. Therefore, considering sufficiently large k's, we have

$$\Pr[\mathcal{R} \leftarrow_{\mathtt{R}} \mathbb{R}; x \leftarrow_{\mathtt{R}} \mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y) : \mathsf{N}^{\mathcal{R}}(x) = \widetilde{\mathsf{P}}^{\mathcal{R}}(x) = y | y \text{ is good}]$$
$$\geq 1 - \sqrt{\epsilon'} = 1 - 2^{-k/6}.$$

Now, define an oracle PPTA adversary $\widetilde{\mathcal{A}}$, which runs in $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k)$, by $\widetilde{\mathcal{A}}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y^*) = \mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y^*)$. Since x^* is chosen uniformly from $\{0,1\}^\ell$ in $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k)$ and $\widetilde{\mathsf{P}}^{\mathcal{R}}$ is a permutation, $y^* = \widetilde{\mathsf{P}}^{\mathcal{R}}(x^*)$ is distributed uniformly over $\{0,1\}^\ell$. Therefore, for sufficiently large k's, we have:

$$
\begin{aligned}
&\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k)\\
&= \Pr[\mathcal{R} \leftarrow_{\mathtt{R}} \mathbb{R}; x^* \leftarrow_{\mathtt{R}} \{0,1\}^\ell; y^* \leftarrow \widetilde{\mathsf{P}}^{\mathcal{R}}(x^*); x' \leftarrow_{\mathtt{R}} \widetilde{\mathcal{A}}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y^*) : x' = x^*]\\
&\geq \Pr[\mathcal{R} \leftarrow_{\mathtt{R}} \mathbb{R}; y^* \leftarrow_{\mathtt{R}} \{0,1\}^\ell; x' \leftarrow_{\mathtt{R}} \mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y^*) : \mathsf{N}^{\mathcal{R}}(x') = \widetilde{\mathsf{P}}^{\mathcal{R}}(x') = y^*]\\
&\geq \Pr[\mathcal{R} \leftarrow_{\mathtt{R}} \mathbb{R}; x' \leftarrow_{\mathtt{R}} \mathsf{Q}^{\mathcal{R},\mathsf{PSPACE}}(1^k, y^*) : \mathsf{N}^{\mathcal{R}}(x') = \widetilde{\mathsf{P}}^{\mathcal{R}}(x') = y^* | y^* \text{ is good}]\\
&\quad \times \Pr_{y^* \leftarrow \{0,1\}^\ell}[y^* \text{ is good}]\\
&\geq (1 - 2^{-k/6}) \cdot (1 - 2 \cdot 2^{-k/6}) \geq 1 - 3 \cdot 2^{-k/6}.
\end{aligned}
$$

Finally, by the property (2) of P in Lemma 7, for this $\widetilde{\mathcal{A}}$, there exist an oracle PPTA adversary $\mathcal{A}$, which runs in $\widetilde{\mathsf{Expt}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}},\ell}(k)$, and a negligible function $\mu(k)$ such that for sufficiently large k's:

$$
\widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}}}(k) \geq \widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\widetilde{\mathsf{P}}^{\mathbb{R}},\widetilde{\mathcal{A}}^{\mathbb{R},\mathsf{PSPACE}},\ell}(k) - \mu(k) \geq 1 - 3 \cdot 2^{-k/6} - \mu(k).
$$

What we have shown thus far is that there exists an oracle PPTA $\mathcal{A}$ such that $\mathbf{E}_{\mathcal{T} \leftarrow_{\mathtt{R}} \mathbb{T}}[\mathsf{Adv}^{\mathtt{OWP}}_{\mathsf{P}^{\mathcal{T}},\mathcal{A}^{\mathcal{T},\mathsf{PSPACE}},\ell}(k)] = \widetilde{\mathsf{Adv}}^{\mathtt{OWP}}_{\mathsf{P}^{\mathbb{T}},\mathcal{A}^{\mathbb{T},\mathsf{PSPACE}}}(k)$ is overwhelming. The above can be shown for all positive polynomials $\ell(k)$ and any PPTA P such that $\mathsf{P}^{\mathcal{T}} \in \mathsf{Perm}_\ell$ for all $\mathcal{T} \in \mathbb{T}$. This completes the proof of Lemma 4. □

4 Towards More General Separations

Broader Class of Permutations and Permutation Families. As in the previous black-box separation results of a OWP from other basic primitives [43, 26, 9, 33], our separation results rule out a black-box construction of a OWP which is *defined over strings* (i.e. the domain is a set of strings of a fixed length determined by the security parameter). However, we can consider a more general form of a permutation whose domain is not just a set of strings but an arbitrary set D, and which has a corresponding sampling algorithm Samp to sample an element from the domain D (although such formulation of a OWP is not standard). Furthermore, as a more natural and closely related primitive to a OWP, we can also consider a public-coin OWPF.

Therefore, a natural question regarding our result will be: "*Can our impossibility result be extended to also rule out a black-box construction of a OWP with such general form of domain or of a public-coin OWPF?*"

We note that previously to our work, Hsiao and Reyzin [24] conjectured that there is no black-box construction of a public-coin OWPF from a secret-coin OWPF. We can partially answer to the above question in the positive due to the result by Goldreich et al. [17], who showed that there is a (fully-)black-box construction of a OWP from a public-coin OWPF with the *canonical domain sampling* property (see Section 2.2 for the definition and a brief review of the construction of [17]). This result, combined with Theorem 1, yields the following corollary.

Corollary 2. *There is no fully-black-box construction of a public-coin OWPF with canonical domain sampling from an ideal TDP.*

It seems to us that if we consider another restricted type of a constructed public-coin permutation family such that the sampling algorithm Samp of the constructed permutation family does not use the algorithms of a building block TDP, then we can rule out a black-box construction of such public-coin OWPF from an ideal TDP, with essentially the same approach used to show Theorem 1 (although we have not formally checked this). This is because if Samp of the constructed public-coin permutation family does not depend on the TDP used as a building block, then whenever we use a same evaluation key ek, the domain D_{ek} of a permutation $\mathsf{Eval}(ek, \cdot)$ remains the same, and thus slight modifications of Lemmas 7 to 9 seem to work accordingly.

Other than these observations, so far we do not know how to rule out the possibility of constructing a public-coin OWPF from a TDP (or even from an ordinary secret-coin OWPF) in general, and thus we would like to leave it as an interesting open problem. Goldreich et al. [17] showed that under the standard RSA assumption or a discrete logarithm assumption in the integer group $\mathbb{Z}_p^*$ (with some appropriate condition on p), we can construct a public-coin OWPF with the canonical domain sampling property, and hence a OWP. However, they noted that how to construct a OWP or a public-coin OWPF under the standard factoring assumption is still open. Tackling the above open problem of clarifying whether there exists a black-box construction of a public-coin OWPF from a secret-coin OWPF will also contribute to this problem: If it turns out to be possible (which we think is unlikely), then we can use the Rabin TDP [40] as a building block to construct a public-coin OWPF, while if it is not possible, one has to essentially use some specific algebraic property to build a public-coin OWPF under the factoring assumption.

Stronger Separation. So far, all our results are impossibility of a fully-black-box construction, which is the most restrictive type of black-box constructions. With a slight modification, however, our separation results can be strengthened to show that there is no *semi*-black-box construction (in the taxonomy of Reingold et al. [41]) of a OWP (and a public-coin OWPF with canonical domain sampling) from an ideal TDP. Specifically, to show such a result, we need to show a "single" oracle which simultaneously implements an ideal TDP and PSPACE. However, our TDP oracle $\mathcal{T}$ can be easily modified to such an oracle by using the "embedding" technique due to Reingold et al. [41]. We discuss more details in the full version.

Acknowledgement The author would like to thank Jacob Schuldt for many insightful comments and discussions on several drafts of this paper. The author would also like to thank the members of the study group "Shin-Akarui-Angou-Benkyou-Kai" and the anonymous reviewers for their helpful comments and suggestions.

References

1. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S.P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, LNCS 2139, pp. 1-18, 2001.
2. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT 2009*, LNCS 5479, pp. 1-35, 2009.
3. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT 1994*, LNCS 950, pp. 92-111, 1995.
4. M. Bellare and P. Rogaway. The exact security of digital signatures – how to sign with RSA and Rabin. In *EUROCRYPT 1996*, LNCS 1070, pp. 399-416, 1996.
5. M. Bellare and M. Yung. Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation. *J. of Cryptology*, 9(3):149-166, 1996.
6. R. Bhattacharyya and A. Mandal. On the impossibility of instantiating PSS in the standard model. In *PKC 2011*, LNCS 6571, pp. 351-368, 2011.
7. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Computing*, 13(4):850-864, 1984.
8. D. Boneh, P.A. Papakonstantinou, C. Rackoff, Y. Vahlis, and B. Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *FOCS 2008*, pp. 283-292, 2008.
9. Y.-C. Chang, C.-Y. Hsiao, and C.-J. Lu. The impossibility of basing one-way permutations on central cryptographic primitives. *J. of Cryptology*, 19(1):97-114, 2006.
10. J.-S. Coron, J. Patarin, and Y. Seurin. The random oracle model and the ideal cipher model are equivalent. In *CRYPTO 2008*, LNCS 5157, pp. 1-20, 2008.
11. Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of the full domain hash. In *CRYPTO 2005*, LNCS 3621, pp. 449-466, 2005.
12. D. Fiore and D. Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. In *TCC 2012*, LNCS 7194, pp. 636-653, 2012.
13. M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT 2010*, LNCS 6110, pp. 197-215, 2010.
14. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC 2011*, pp. 99-108, 2011.
15. Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS 2001*, pp. 126-135, 2001.
16. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792-807, 1986.
17. O. Goldreich, L.A. Levin, and N. Nisan. On constructing 1-1 one-way functions. In *Studies in Complexity and Cryptography*, LNCS 6650, pp. 12-25, 2011.
18. O. Goldreich and R.D. Rothblum. Enhancements of trapdoor permutations. *J. of Cryptology*, 26(3):484-512, 2013.
19. S. Goldwasser, S. Micali, and R. Rivest. A digital signature schemes secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281-308, 1988.
20. I. Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In *TCC 2004*, LNCS 2951, pp. 394-409, 2004.
21. I. Haitner and T. Holenstein. On the (im)possibility of key dependent encryption. In *TCC 2009*, LNCS 5444, pp. 202-219, 2009.
22. J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudorandom generator from any one-way function. *SIAM J. Computing*, 28(4):1364-1396, 1999.
23. T. Holenstein, R. Künzler, and S. Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In *STOC 2011*, pp. 89-98, 2011.

24. C.-Y. Hsiao and L. Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In *CRYPTO 2004*, LNCS 3152, pp. 92-105, 2004.
25. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC 1989*, pp. 44-61, 1989.
26. J. Kahn, M. Saks, and C. Smyth. A dual version of Reimer's inequality and a proof of Rudich's conjecture. In *CoCo 2000*, pp. 98-103, 2000.
27. J. Katz and A. Yerukhimovich. On black-box constructions of predicate encryption from trapdoor permutations. In *ASIACRYPT 2009*, LNCS 5912, pp. 197-213, 2009.
28. E. Kiltz, P. Mohassel, and A. O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In *EUROCRYPT 2010*, LNCS 6110, pp. 673-692, 2010.
29. E. Kiltz and K. Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In *EUROCRYPT 2009*, LNCS 5479, pp. 389-406, 2009.
30. Y. Lindell and H. Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer, 2009. Full version of [31]. Available at u.cs.biu.ac.il/˜zarosih/papers/adaptive-full version.pdf.
31. Y. Lindell and H. Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. In *TCC 2009*, LNCS 5444, pp. 183-201, 2009.
32. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Computing*, 17(2):373-386, 1988.
33. T. Matsuda and K. Matsuura. On black-box separations among injective one-way functions. In *TCC 2011*, LNCS 6597, pp. 597-614, 2011.
34. U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions and applications to the random oracle methodology. In *TCC 2004*, LNCS 2951, pp. 21-39, 2004.
35. M. Naor. Bit commitment using pseudorandomness. *J. of Cryptology*, 4(2):151-158, 1991.
36. M. Naor. On cryptographic assumptions and challenges. In *CRYPTO 2003*, LNCS 2729, pp. 96-109, 2003.
37. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *STOC 1989*, pp. 33-43, 1989.
38. O. Pandey, R. Pass, and V. Vaikuntanathan. Adaptive one-way functions and applications. In *CRYPTO 2008*, LNCS 5157, pp. 57-74, 2008.
39. R. Pass. Limits of provable security from standard assumptions. In *STOC 2011*, pp. 109-118, 2011.
40. M.O. Rabin. Digitalized signatures as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, January 1979.
41. O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC 2004*, LNCS 2951, pp. 1-20, 2004.
42. A. Rosen and G. Segev. Chosen-ciphertext security via correlated products. In *TCC 2009*, LNCS 5444, pp. 419-436, 2009.
43. S. Rudich. Limits on the provable consequences of one-way functions, 1988. PhD thesis, University of California at Berkeley.
44. Y. Vahlis. Two is a crowd? a black-box separation of one-wayness and security under correlated inputs. In *TCC 2010*, LNCS 5978, pp. 165-182, 2010.
45. H. Wee. On obfuscating point functions. In *STOC 2005*, pp. 523-532, 2005.
46. D. Wichs. Barriers in cryptography with weak, correlated and leaky sources. In *Proc of ITCS 2013*, pp. 111-126, 2013.
47. A.C.-C. Yao. Theory and application of trapdoor functions. In *FOCS 1982*, pp. 80-91, 1982.
48. A. Yerukhimovich. A study of separation in cryptography: New results and new models, 2011. PhD thesis, the University of Maryland. Available at www.cs.umd.edu/˜arkady/thesis/thesis.pdf.