On Black-Box Separations among Injective One-Way Functions

Takahiro Matsuda* and Kanta Matsuura

The University of Tokyo, Japan {tmatsuda,kanta}@iis.u-tokyo.ac.jp

Abstract. A one-way permutation (OWP) is one of the most fundamental cryptographic primitives, and can be used as a building block for most of basic symmetric-key cryptographic primitives. However, despite its importance and usefulness, previous black-box separation results have shown that constructing a OWP from another primitive seems hopeless, unless building blocks already achieve "one-way" property and "permutation" property simultaneously. In this paper, in order to clarify more about the constructions of a OWP from other primitives, we study the construction of a OWP from primitives that are very close to a OWP. Concretely, as a negative result, we show that there is no fully black-box construction of a OWP from a length-increasing injective one-way function (OWF), even if the latter is just 1-bit-increasing and achieves strong form of one-wayness which we call *adaptive one-wayness*. As a corollary, we show that there is no fully black-box construction of a OWP from a regular OWF with regularity greater than 1. Since a permutation is length-preserving and injective, and is a regular OWF with regularity 1, our negative result indicates that to construct a OWP from another primitive is quite difficult, even if we use very close primitives to a OWP as building blocks. Moreover, we extend our separation result of a OWP from a length-increasing injective OWF, and show a certain restrictive form of black-box separations among injective OWFs in terms of how much a function stretches its input. This result shows a hierarchy among injective OWFs (including a OWP).

Keywords: black-box separation, injective one-way function, one-way permutation, adaptive one-wayness.

1 Introduction

A one-way permutation (OWP) is one of the most fundamental cryptographic primitives[1]. It has been shown that OWPs are sufficient for constructing most of basic "symmetric-key" primitives, which include, e.g. pseudorandom generators [18, 2], pseudorandom functions [4], symmetric-key encryption schemes and message authentication codes. While most of primitives implied by a OWP are

* Takahiro Matsuda is supported by JSPS Research Fellowships for Young Scientists.

[1] Unless otherwise stated explicitly, whenever we say "OWP", we mean a single OWP, not a family of OWPs

later shown to be implied by an ordinary one-way function (OWF) (e.g. a pseudorandom generator from any OWF [7]), it is usual that a primitive built from a OWP is more efficient than the one built from a general OWF. Therefore, a OWP is still quite an important primitive, and constructions of a OWP from other (simpler) primitives (possibly efficiently) is worth studying. However, how to construct a OWP from other primitives has not been studied well, compared to constructions of other primitives that use a OWP as a building block. In this paper, we focus on constructions of OWPs.

There are several negative results on this, in terms of black-box constructions[2]. The combination of the results by Rudich [14] and Kahn et al. [10] shows that there is no black-box construction of a OWP from a OWF. Chang et al. [3] later extend it and show that there is no black-box construction of a OWP from (a family of) trapdoor functions or private information retrieval protocols. These negative results indicate that it is quite difficult to construct a OWP from other primitives. There are also some positive results. However, to the best of our knowledge, the only positive results about the construction of a OWP are the constructions from primitives which already have "one-way" property and "permutation" property simultaneously: Yao [18] shows that the existence of a permutation which is weakly one-way implies that of a (normal) OWP. Goldreich et al. [5] show that if a family of OWPs satisfies some special properties, it can be used to construct a single OWP.

Taking into account these negative and positive results on the constructions of OWPs, a natural question that arises here is: *Which properties of a OWP make it hard to construct a OWP from other primitives?* To clarify this, in this paper, we focus on a special type of OWFs, a length-increasing injective OWF, and tackle the problem of whether we can construct a OWP from it. Recall that a permutation is a function which is both length-preserving and injective. Therefore, a length-increasing injective OWF is one of the primitives that is extremely close to a OWP. Regarding this problem, our answer is negative.

The problem on a OWP versus a length-increasing injective OWF can be generalized into the following form: *Let m, ℓ be integers with $m > \ell \geq 0$. Can we construct an ℓ-bit-increasing injective OWF from an m-bit-increasing injective OWF?* (Note that if $m \leq \ell$, then the answer to the question is trivially yes.) We also tackle this problem and show a partial negative answer.

1.1 Our Contribution

In this paper, we show that even if we use a very close primitive to a OWP, it is impossible to construct a OWP in a black-box manner. More concretely,

[2] Roughly speaking, a construction of a primitive from another is black-box if the constructed primitive does not use the code of the building block primitive, and the reduction algorithm for the security proof does not use the code of an adversary attacking the constructed primitive. In this paper, we only talk about the so-called *fully black-box* constructions (reductions) defined in [13]. See [13] for other types of black-box constructions/reductions.

we show in Section 3 that there is no fully black-box construction[3] of a OWP from a length-increasing injective OWF, even if the latter is just 1-bit-increasing and achieves stronger type of one-wayness which we call *adaptive one-wayness* (see below). We note that this separation result is not implied by the previous results [14, 10, 3] on a black-box separation of a OWP from other primitives. Actually, one of the results in [3] implies the separation of a OWP from an injective OWF whose range is sparse (e.g. length-tripling functions). However, our impossibility holds as long as the building block injective OWF is length-increasing, regardless of the sparseness of the range. An immediate corollary of the separation result is the non-existence of a fully black-box construction of a OWP from a regular OWF[4] for any regularity greater than 1. Note that a OWP is also a regular OWF with regularity 1. Therefore, our negative results suggest that constructing a OWP is quite difficult (or maybe impossible) unless a building block primitive already achieves both "one-way" property and "permutation" property simultaneously.

Moreover, we extend our above result to show some restricted type of black-box separations among injective OWFs. More precisely, we show in Section 4 that for any integer pair (ℓ, m) satisfying $m > \ell \geq 0$, there is no *range-invariant* fully black-box construction of an ℓ-bit-increasing injective OWF from an m-bit-increasing injective OWF, where a construction of an injective OWF from other primitives is said to be range-invariant if the range of the constructed injective OWF depends only on the construction and is independent of the building block primitives. Note that a OWP is a 0-bit-increasing injective OWF, and any construction of a OWP from other primitives is inherently range-invariant, and thus our first separation result is the special case of the latter one in which $\ell = 0$. Although this range-invariance condition seems a bit heavy when $\ell > 0$, this result shows a hierarchy among injective OWFs (including a OWP), and we think this result is interesting. So far, we are not sure whether we can remove the range-invariance condition from this separation result, and thus we would like to leave it as an open problem.

In order to make our black-box separation results stronger, for both of our separation results, we consider a stronger type of one-wayness, *adaptive one-wayness*, for building block OWFs. Roughly, an injective function is adaptively one-way if it is one-way against adversaries that have access to a "magical" inversion oracle which takes a string (other than a challenge instance that the adversary has to invert) as input and returns the inverse of the value. Our definition of adaptive one-wayness is different from the one introduced by Pandey et al. [11] who considered it in a "tag-based" setting while ours does not consider it. See Section 2 for a formal definition.

---

[3] This is the most restrictive type of black-box constructions formalized in [13]. However, it should be noticed that most cryptographic constructions of a primitive from another primitive is fully black-box.

[4] Roughly, a function is said to be regular if it is length-preserving and each image of the function has the same-sized set of preimages, and the regularity is the size of the set of preimages which map to a same image.

1.2 Technical Overview of Our Separation Results

The combination the results by Rudich [14] and Kahn et al. [10] shows that there is no (so-called $\forall\exists$ semi-)black-box construction of a OWP from a OWF. However, it is known that if we only need to exclude a more restrictive form of black-box constructions, *fully black-box* constructions, of a OWP from a OWF, proving the following statement is sufficient (see also [12]): "*For any oracle probabilistic polynomial time algorithm (PPTA)* P*, if* $\mathsf{P}^{\mathcal{O}}$ *implements a permutation for all random oracles*[5] $\mathcal{O}$*, then there is an oracle PPTA that has access to* $\mathcal{O}$ *and a* PSPACE*-complete oracle and inverts* $\mathsf{P}^{\mathcal{O}}$*.*"[6] Let us call this statement "(A)".

Since we will use the basic proof strategy for the statement (A) in our black-box separation results, we briefly outline the proof, and then explain the problems that arise when proving our results.

Basic Proof Strategy for Separation of OWP and (Ordinary) OWF. To prove the statement (A), we construct a computationally unbounded adversary $\mathcal{A}$ that makes only polynomially many queries to its given random oracle $\mathcal{O}$ and successfully inverts $\mathsf{P}^{\mathcal{O}}$: Given a string $y^* \in \{0,1\}^k$ which $\mathcal{A}$ has to find the preimege under $\mathsf{P}^{\mathcal{O}}$, $\mathcal{A}$ first generates an empty list L that will be used to maintain the "known" query/answer pair of $\mathcal{O}$, and then repeats the following steps 1 to 3 for polynomially many times:

Step 1: find a string x' and an oracle $\widetilde{\mathcal{O}}$ under the condition: $\widetilde{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha)$ for all $\alpha \in L$ and $\mathsf{P}^{\widetilde{\mathcal{O}}}(x') = y^*$.

Step 2: check if $\mathsf{P}^{\mathcal{O}}(x') = y^*$ (note that here we use $\mathcal{O}$, not $\widetilde{\mathcal{O}}$), and terminate with output this x' if this is the case.

Step 3: ask $\mathcal{O}$ all the queries made by $\mathsf{P}^{\widetilde{\mathcal{O}}}(x')$ and update the known query/answer pair list L.

The key observation is that *in each iteration, either (a)* $\mathcal{A}$ *finds a preimage* x^* *such that* $\mathsf{P}^{\mathcal{O}}(x^*) = y^*$ *and terminates at Step 2, or (b) at Step 3* $\mathcal{A}$ *finds (and stores in* L*) at least one query that is also made by* P *during the computation of* $y^* = \mathsf{P}^{\mathcal{O}}(x^*)$ *but has not been contained in the known queries* L*.* Very roughly, this is because if (a) and (b) are simultaneously false, then we can construct a "hybrid" random oracle $\widehat{\mathcal{O}}$ that behaves like $\mathcal{O}$ on the queries made by $\mathsf{P}^{\mathcal{O}}(x^*)$ and like $\widetilde{\mathcal{O}}$ on those made by $\mathsf{P}^{\widetilde{\mathcal{O}}}(x')$. This hybrid oracle $\widehat{\mathcal{O}}$ has the property that $\mathsf{P}^{\widehat{\mathcal{O}}}(x^*) = \mathsf{P}^{\widehat{\mathcal{O}}}(x') = y^*$ while $x^* \neq x'$, which is a contradiction because $\mathsf{P}^{\widehat{\mathcal{O}}}$ implements a permutation which cannot have a collision (recall that $\mathsf{P}^{\mathcal{O}'}$ implements a permutation for all random oracles $\mathcal{O}'$). Since P makes only polynomially many queries to $\mathcal{O}$, by repeating the above procedure polynomially many times $\mathcal{A}$ eventually finds the preimage x^* and terminates, or we reach the situation where L contains all the queries made by $\mathsf{P}^{\mathcal{O}}(x^*)$. Note that if L contains all

[5] Here, "all random oracles" should be interpreted as "all the possible instances of random oracles".

[6] According to [14, Sect. 9.1], this statement can be shown as a corollary of the result independently discovered in [1, 6, 16]. For more details, see [14].

the queries made by $\mathsf{P}^{\mathcal{O}}(x^*)$, then the preimage of y^* under the permutation $\mathsf{P}^{\widetilde{\mathcal{O}}}$ (where $\widetilde{\mathcal{O}}$ is the oracle chosen at Step 1) must be x^*, because $\mathcal{O}(\alpha) = \widetilde{\mathcal{O}}(\alpha)$ for all the queries α made by $\mathsf{P}^{\mathcal{O}}(x^*)$, which means that $\mathsf{P}^{\mathcal{O}}(x^*) = \mathsf{P}^{\widetilde{\mathcal{O}}}(x^*) = y^*$. Since Step 1 in each iteration can be simulated by a PPTA with oracle access to a PSPACE-complete oracle[7], $\mathcal{A}$ can actually be a PPTA.

Problems for Separations of OWP and Length-Increasing Injective OWF. For our purpose of (fully) black-box separation of a OWP from a length-increasing injective OWF (say, m-bit-increasing with $m > 0$), we would like to replace the random oracle in the statement (A) with a random instance of oracles $\mathcal{O}$ which is m-bit-increasing and injective (we call it m-bit-increasing injective oracle). We are given an oracle PPTA P such that $\mathsf{P}^{\mathcal{O}'}$ implements a permutation for all m-bit-increasing injective oracles $\mathcal{O}'$. Then, consider the same proof strategy as above, i.e. constructing a PPTA adversary $\mathcal{A}$ that has access to an m-bit-increasing injective oracle $\mathcal{O}$ and PSPACE-complete oracle, and tries to invert a given instance $y^* = \mathsf{P}^{\mathcal{O}}(x^*)$ by the above procedure. However, if we naively do the sampling process of x' and $\widetilde{\mathcal{O}}$ at Step 1, we will meet some problem when arguing the above key observation. More concretely, even though we have $\mathsf{P}^{\widetilde{\mathcal{O}}}(x') = y^*$ and $x^* \neq x'$, it might be the case that the range of $\mathcal{O}$ and $\widetilde{\mathcal{O}}$ have an overlap outside the range corresponding to L, which could prevent the hybrid oracle $\widehat{\mathcal{O}}$ from being injective. If $\widehat{\mathcal{O}}$ is not injective, then it is no longer guaranteed that $\mathsf{P}^{\widehat{\mathcal{O}}}$ is a permutation, and thus having a collision does not cause a contradiction.

Therefore, in order to avoid such a situation, we have to choose the m-bit-increasing injective oracle $\widetilde{\mathcal{O}}$ in Step 1 so that we can always construct an "injective" hybrid oracle $\widehat{\mathcal{O}}$ from $\mathcal{O}$ and $\widetilde{\mathcal{O}}$. Our solution is to choose $\widetilde{\mathcal{O}}$ so that (i) $\widetilde{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha)$ for all $\alpha \in L$ and (ii) for all other inputs from $\{0,1\}^* \backslash L$, the range of $\widetilde{\mathcal{O}}$ and that of $\mathcal{O}$ are disjoint. It will be shown later that picking such $\widetilde{\mathcal{O}}$ is always possible as long as $\widetilde{\mathcal{O}}$ is length-increasing.

Another problem that arises here is that it might not be possible to execute Step 1 modified as above by only using a PSPACE-complete oracle and making only polynomially many queries to $\mathcal{O}$, because it seems that we need the entire knowledge about the range of $\mathcal{O}$ in order to pick such $\widetilde{\mathcal{O}}$. However, since $\mathcal{O}$ is a random m-bit-increasing injective oracle, it seems hard to know the range of $\mathcal{O}$ entirely by making only polynomially many queries to $\mathcal{O}$. To solve it, we adopt the so-called "two oracle" separation paradigm introduced by Hsiao and Reyzin [8], which is sufficient for showing the non-existence of fully black-box constructions. More concretely, we introduce another oracle $\mathcal{B}$ (which we call "breaking oracle") that helps $\mathcal{A}$ to do the above procedure of picking x' such that $\mathsf{P}^{\widetilde{\mathcal{O}}}(x') = y^*$ where $\widetilde{\mathcal{O}}$ is chosen as above. To make Step 3 possible, the oracle $\mathcal{B}$ also outputs a query set that is made by $\mathsf{P}^{\widetilde{\mathcal{O}}}(x')$ to $\widetilde{\mathcal{O}}$.

[7] This is because what we need is not the entire oracle $\widetilde{\mathcal{O}}$, but the queries made by $\mathsf{P}^{\widetilde{\mathcal{O}}}(x')$, which is a witness for a certain NP language statement, which can be picked by using a PSPACE-complete oracle.

Now, since we have introduced a new oracle $\mathcal{B}$, we also have to show that an m-bit-increasing injective oracle $\mathcal{O}$ is one-way in the presence of $\mathcal{B}$. We will show that (adaptive) one-wayness of $\mathcal{O}$ in the presence of $\mathcal{B}$ can be reduced to (adaptive) one-wayness of a *random permutation oracle* π against computationally unbounded adversary $\mathcal{S}$ that makes only polynomially many queries.

1.3 Paper Organization

The rest of this paper is organized as follows. In Section 2 we review some basic definitions and facts used for describing our results. In Section 3, we show our main result on a black-box separation of a OWP from a length-increasing injective OWF, and we extend the result for a restricted type of black-box separations among injective OWFs in Section 4.

2 Preliminaries

In this section, we review basic definitions and some fact necessary used for describing our results.

Basic Notations. Throughout this paper, we use the following notations: "$\mathbb{N}$" denotes the set of natural numbers. "$x||y$" denotes a concatenation of x and y. If x is a string, then "$|x|$" denotes the bit length of x. "$x \leftarrow y$" 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. If Ψ is a probability distribution, then "$x \leftarrow_{\mathtt{R}} \Psi$" denotes that x is chosen according to Ψ, and "$[\Psi]$" denotes the "support" of Ψ, that is, $[\Psi] = \{x | \Pr_{x' \leftarrow_{\mathtt{R}} \Psi}[x' = x] > 0\}$. "PPTA" denotes *probabilistic polynomial time algorithm*. If $\mathcal{A}$ is a (probabilistic) algorithm, then "$z \leftarrow_{\mathtt{R}} \mathcal{A}(x, y, \dots)$" denotes that $\mathcal{A}$ takes x, y, $\dots$ as input and outputs z, and "$\mathcal{A}^{\mathcal{O}}$" denotes that $\mathcal{A}$ has oracle access to an oracle $\mathcal{O}$. If f is a function/algorithm/oracle and S is a (sub)domain of f, then we define $f(S) = \{f(x) | x \in S\}$. We say that an oracle algorithm has query complexity q if the algorithm makes at most q queries to its given oracle. "Perm_n" denotes the set of all permutations over $\{0,1\}^n$. A function $f : \mathbb{N} \to [0,1]$ is said to be *negligible* in k if $f(k) < 1/p(k)$ for any positive polynomial $p(k)$ and all sufficiently large $k \in \mathbb{N}$.

2.1 One-Way Functions

We say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way function (OWF) if there exists a PPTA that for any x computes $f(x)$, and for any PPTA $\mathcal{A}$, the following advantage function $\mathsf{Adv}^{\mathtt{OW}}_{f,\mathcal{A}}(k)$ is negligible in k:

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

If the function f is injective, then we call it an injective OWF. If f is injective and length-preserving (i.e. permutation), we call it a one-way permutation (OWP). If $|f^{-1}(f(x))| = |f^{-1}(f(y))|$ for any $x, y \in \{0,1\}^n$ and any $n \in \mathbb{N}$, we call it

a regular OWF, and in particular, if $\alpha(n) = |f^{-1}(f(x))|$, then we call it an α-regular OWF.

Adaptive One-wayness for Injective Functions. In this paper, we will use a stronger type of one-wayness for strengthening our black-box separation results.

We say that an injective function $f : \{0,1\}^* \to \{0,1\}^*$ is adaptive one-way, if there exists a PPTA that for any x computes $f(x)$, and for any PPTA $\mathcal{A}$, the following advantage function $\mathsf{Adv}^{\mathtt{AOW}}_{f,\mathcal{A}}(k)$ is negligible in k.

$$\mathsf{Adv}^{\mathtt{AOW}}_{f,\mathcal{A}}(k) = \Pr[x^* \leftarrow_{\mathtt{R}} \{0,1\}^k; y^* \leftarrow f(x^*); x' \leftarrow_{\mathtt{R}} \mathcal{A}^{f^{-1}_{\neq y^*}(\cdot)}(1^k, y^*) : x' = x^*],$$

where $f^{-1}_{\neq y^*}(\cdot)$ is the *inversion* oracle which takes a string y as input and outputs x such that $f(x) = y$ if such x exists and $y \neq y^*$, or outputs $\perp$ otherwise.

We also say that f is an adaptive one-way function (AOWF). (Whenever we say f is an AOWF, we always mean that f is injective.)

2.2 Basic Fact about Random Permutations

In this paper, we will use a simple fact that a random permutation is adaptively one-way even against a computationally unbounded adversary who is given oracle access to the permutation and its inversion only polynomially many times.

Let $\mathcal{A}$ be an oracle adversary. Consider the following experiment $\mathsf{Expt}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{A}}(k)$:

$$\mathsf{Expt}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{A}}(k) : [\pi \leftarrow_{\mathtt{R}} \mathsf{Perm}_k; \alpha^* \leftarrow_{\mathtt{R}} \{0,1\}^k; \beta^* \leftarrow \pi(\alpha^*);$$
$$\mathtt{Return}\ 1\ \mathtt{iff}\ \mathcal{A}^{\pi,\pi^{-1}_{\neq \beta^*}}(1^k, \beta^*)\ \mathtt{returns}\ \alpha^*]$$

(Here, "RP" stands for "random permutation", and note that the experiment includes the choice of the permutation π.) We define the advantage function of an oracle adversary $\mathcal{A}$ by $\mathsf{Adv}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{A}}(k) = \Pr[\mathsf{Expt}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{A}}(k) = 1]$. Regarding the above experiment, the following is easy to prove (the proof is omitted due to lack of space).

Lemma 1. *For any (even computationally unbounded) adversary $\mathcal{A}$ with polynomial query complexity, $\mathsf{Adv}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{A}}(k)$ is negligible in k. Specifically, if $\mathcal{A}$ has query complexity q, then $\mathsf{Adv}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{A}}(k) \leq \frac{(q+1)}{2^k - q}$.*

3 Black-box Separation of OWP from Length-Increasing Injective AOWF

In this section, we show that there is no fully black-box construction of a OWP from a length-increasing injective OWF, even if the latter is just 1-bit-increasing and achieves adaptive one-wayness.

We first recall the formal definition of a fully black-box construction [13] of a OWP from an m-bit-increasing injective AOWF.

Definition 1. *Let $m > 0$ be an integer. We say that there exists a fully black-box construction of a OWP from an m-bit-increasing injective AOWF, if there exist oracle PPTAs P and $\mathcal{R}$ such that for all functions f that implement m-bit-increasing injective functions and all algorithms $\mathcal{A}$ (where f and $\mathcal{A}$ are of arbitrary complexity):*

Correctness: P^f *is a permutation.*
Security: *If* $\mathsf{Adv}^{\mathtt{OW}}_{\mathsf{P}^f,\mathcal{A}}(k)$ *is non-negligible, so is* $\mathsf{Adv}^{\mathtt{AOW}}_{f,\mathcal{R}^{f,\mathcal{A}}}(k)$.

Now, we show the following separation between a OWP and a length-increasing injective OWF:

Theorem 1. *For any integer $m > 0$, there is no fully black-box construction of a OWP from an m-bit-increasing injective AOWF.*

To prove Theorem 1, We use a variant of the two oracle separation paradigm [8]:

Lemma 2. *Let $m > 0$ be an integer. Assume that there exists a distribution Ψ of an oracle pair $(\mathcal{O}, \mathcal{B})$ that satisfies the following three conditions:*

(1): $\mathcal{O}$ implements an m-bit-increasing injective function for all $(\mathcal{O}, \mathcal{B}) \in [\Psi]$.
(2): For any oracle PPTA $\mathcal{A}$, $\mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_R\Psi}[\mathsf{Adv}^{\mathtt{AOW}}_{\mathcal{O},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)]$ is negligible.
(3): For any oracle PPTA P, if $\mathsf{P}^{\mathcal{O}'}$ implements a permutation for all $(\mathcal{O}', \mathcal{B}') \in [\Psi]$, then there is an oracle PPTA $\mathcal{A}$ s. t. $\mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_R\Psi}[\mathsf{Adv}^{\mathtt{OW}}_{\mathsf{P}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)] = 1$.

Then, there is no fully black-box construction of a OWP from an m-bit-increasing injective AOWF.

In order to use Lemma 2, we define the distribution Ψ of an oracle pair $(\mathcal{O}, \mathcal{B})$ in Section 3.1. Then we show that Ψ satisfies the conditions (1) and (2) in Section 3.2, and Ψ satisfies the condition (3) in Section 3.3. Finally in Section 3.4 we show the formal proof of Theorem 1.

3.1 Definitions of Oracles and Their Distribution

m-Bit-Increasing Injective Oracle $\mathcal{O}$. Let $m > 0$ be an integer. We say that an oracle $\mathcal{O}$ is an *m-bit-increasing injective oracle* if (1) for every $n \in \mathbb{N}$, $\mathcal{O}$ is of the form $\mathcal{O} : \{0,1\}^n \to \{0,1\}^{n+m}$, and (2) $\mathcal{O}$ is injective. Let $\mathbb{O}_m$ be the set of all m-bit-increasing injective oracles.

"Breaking" Oracle $\mathcal{B}$. Before describing the definition of our breaking oracle, we introduce the following notation.

Definition 2. *Let P be an oracle algorithm, $\mathcal{O} \in \mathbb{O}_m$ be an m-bit-increasing injective oracle, and x be a string. A* query set *with respect to $(\mathsf{P}, \mathcal{O}, x)$, denoted by $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)}$, is a set of all the queries to $\mathcal{O}$ made by $\mathsf{P}^{\mathcal{O}}(x)$, i.e., $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)} = \{\ \alpha \mid \mathsf{P}^{\mathcal{O}}(x)$ makes a query α to $\mathcal{O}\ \}$.*

By definition, if the running time of P is bounded by τ_{P}, then $|\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)}| \leq \tau_{\mathsf{P}}$.

Now, we describe the definition of the breaking oracle $\mathcal{B}$ formally. Our breaking oracle $\mathcal{B}$ is associated with an m-bit-increasing injective oracle $\mathcal{O} \in \mathbb{O}_m$. Since $\mathcal{B}$ generates a slightly different m-bit-increasing injective oracle $\widetilde{\mathcal{O}}$ based on $\mathcal{O}$ during its procedure, in order not to mix up them we refer to the oracle $\mathcal{O}$ with which $\mathcal{B}$ is associated as the "original oracle", and the oracle $\widetilde{\mathcal{O}}$ that is generated in the procedure of $\mathcal{B}$ as a "modified oracle".

$\mathcal{B}$ takes the following three objects as input:

1. a description of an oracle algorithm P that is a candidate of a OWP.
2. a set of strings $L \subseteq \{0,1\}^*$
3. a string y

$\mathcal{B}$ then does the following:

Step 1 (Correctness check): Check if $\mathsf{P}^{\mathcal{O}'}$ implements a permutation over $\{0,1\}^{|y|}$ for all the possible instances of m-bit-increasing injective oracles $\mathcal{O}' \in \mathbb{O}_m$. If the check fails, output $\perp$ and stop.

Step 2 (Generating a modified oracle $\widetilde{\mathcal{O}}$): For each $n \in \mathbb{N}$, pick uniformly an m-bit-increasing injective function $g'_n : \{0,1\}^n \to (\{0,1\}^{n+m} \backslash \mathcal{O}(\{0,1\}^n))$. Here, note that the size of the set $\{0,1\}^{n+m} \backslash \mathcal{O}(\{0,1\}^n)$ is $2^{n+m} - 2^n \geq 2^n$ (because $m > 0$), and thus picking such an injective function g'_n is always possible. Then, a "modified" oracle $\widetilde{\mathcal{O}}$ is defined as:

$$\widetilde{\mathcal{O}}(\alpha) = \begin{cases} \mathcal{O}(\alpha) & \text{if } \alpha \in L \\ g'_{|\alpha|}(\alpha) & \text{otherwise} \end{cases}$$

Note that $\widetilde{\mathcal{O}} \in \mathbb{O}_m$: clearly $\widetilde{\mathcal{O}}$ is m-bit-increasing, and is injective for each of the subdomains L and $\{0,1\}^* \backslash L$; the set $\widetilde{\mathcal{O}}(L) = \mathcal{O}(L)$ and the set $\widetilde{\mathcal{O}}(\{0,1\}^* \backslash L) = \{g'_{|\alpha|}(\alpha) | \alpha \in \{0,1\}^* \backslash L\}$ are always disjoint, and thus there is no pair $(\alpha, \alpha') \in L \times (\{0,1\}^* \backslash L)$ such that $\widetilde{\mathcal{O}}(\alpha) = \widetilde{\mathcal{O}}(\alpha')$.

Step 3 (Finding a preimage and a query set wrt. $\widetilde{\mathcal{O}}$): Find x such that $\mathsf{P}^{\widetilde{\mathcal{O}}}(x) = y$. (Note that $x \in \{0,1\}^{|y|}$ is unique since it is guaranteed by Correctness check that $\mathsf{P}^{\widetilde{\mathcal{O}}}$ implements a permutation over $\{0,1\}^{|y|}$.) Output x and the corresponding query set $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}}(x)}$.

The above completes the description of $\mathcal{B}$. Although $\mathcal{B}$ is probabilistic (see Step 2), in order to make $\mathcal{B}$ behave as a deterministic function, we assume that the randomness $\mathcal{B}$ uses to pick functions $\{g'_n(\cdot)\}_{n \in \mathbb{N}}$ is fixed for each input to $\mathcal{B}$, and make $\mathcal{B}$ choose the same functions $\{g'_n(\cdot)\}_{n \in \mathbb{N}}$ for the same input (P, L, y).)

Let τ_{P} be the maximum running time of P, where the maximum is over all oracles $\mathcal{O} \in \mathbb{O}_m$ and all inputs of length $|y|$. Similarly to [15] and [17], we count each $\mathcal{B}$-query as $|L| + \tau_{\mathsf{P}}$ queries, rather than naively counting it as a single query, and make $\mathcal{B}$ output the response after these steps have passed from the point $\mathcal{B}$ receives the input. This is to prevent an adversary from making a very "large" $\mathcal{B}$-query that may give too much information about $\mathcal{O}$ to the adversary.

We call a $\mathcal{B}$-query *valid* if the correctness check passes, and *invalid* otherwise.

What should be noticed about the modified m-bit-increasing injective oracle $\widetilde{\mathcal{O}}$ is: For any $L \subseteq \{0,1\}^*$,

- $\widetilde{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha)$ for all $\alpha \in L$
- the set $\mathcal{O}(\{0,1\}^*)$ and the set $\widetilde{\mathcal{O}}(\{0,1\}^* \backslash L)$ are always disjoint

Moreover, the following property of $\mathcal{B}$ will be used later for breaking any candidate of OWP that is constructed from $\mathcal{O} \in \mathbb{O}_m$.

Lemma 3. *Let* P *be a PPTA such that* $\mathsf{P}^{\mathcal{O}'}$ *implements a permutation for all* $\mathcal{O}' \in \mathbb{O}_m$*. For any string* x*, any* $L \subseteq \{0,1\}^*$*, and any* $\mathcal{O} \in \mathbb{O}_m$*, if* $\mathcal{B}$ *is associated with* $\mathcal{O}$ *and* $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)} \subseteq L$*, then* $\mathcal{B}(\mathsf{P}, L, \mathsf{P}^{\mathcal{O}}(x))$ *returns* x *and* $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)}$*.*

Proof. Fix a string x, a set $L \subseteq \{0,1\}^*$ such that $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)} \subseteq L$, and $\mathcal{O} \in \mathbb{O}_m$. Let $y = \mathsf{P}^{\mathcal{O}}(x)$. By the given condition, P will pass the correctness check. Let $\widetilde{\mathcal{O}}$ be the modified m-bit-increasing injective oracle generated in the step 2 of $\mathcal{B}$. By the definition of $\mathcal{B}$ and the given condition $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)} \subseteq L$, it holds that $\widetilde{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha)$ for all $\alpha \in \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)}$. This means that $\mathsf{P}^{\widetilde{\mathcal{O}}}(x) = \mathsf{P}^{\mathcal{O}}(x) = y$ and $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}}(x)} = \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)}$. Since $\mathsf{P}^{\widetilde{\mathcal{O}}}(\cdot)$ is a permutation, the preimage of y under $\mathsf{P}^{\widetilde{\mathcal{O}}}(\cdot)$ is unique and must be x, which is found and output in Step 3 of $\mathcal{B}$ together with the query set $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}}(x)} = \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x)}$. This completes the proof of Lemma 3. □

Distribution Ψ of Oracles $(\mathcal{O}, \mathcal{B})$. We define "how to pick oracles $(\mathcal{O}, \mathcal{B})$ according to the distribution Ψ" as follows: Pick an m-bit-increasing injective oracle $\mathcal{O} \in \mathbb{O}_m$ uniformly at random, and then pick the breaking oracle $\mathcal{B}$ associated with $\mathcal{O}$ ($\mathcal{B}$'s internal randomness is fixed in this step).

3.2 Adaptive One-Wayness of $\mathcal{O}$ in the Presence of $\mathcal{B}$

From the definition of Ψ in the previous subsection, it is clear that for any $(\mathcal{O}, \mathcal{B}) \in [\Psi]$, $\mathcal{O}$ implements an m-bit-increasing injective function. The rest of this subsection is devoted to proving the following.

Lemma 4. *For any oracle PPTA* $\mathcal{A}$*,* $\mathbf{E}_{(\mathcal{O},\mathcal{B}) \leftarrow_R \Psi}[\mathsf{Adv}^{\mathtt{AOW}}_{\mathcal{O},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)]$ *is negligible.*

Proof. Fix an arbitrary PPTA adversary $\mathcal{A}$, and let $\tau_{\mathcal{A}} = \tau_{\mathcal{A}}(k)$ be $\mathcal{A}$'s maximum running time (when run with input 1^k). Since $\mathcal{A}$ is a PPTA, $\tau_{\mathcal{A}}$ is polynomial.

The expectation (over the choice of $(\mathcal{O}, \mathcal{B})$) of the advantage of the adversary $\mathcal{A}$ attacking adaptive one-wayness of $\mathcal{O}$ can be written as:

$$\mathop{\mathbf{E}}_{(\mathcal{O},\mathcal{B}) \leftarrow_R \Psi} \left[\mathsf{Adv}^{\mathtt{AOW}}_{\mathcal{O},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k) \right]$$

$$= \Pr[(\mathcal{O}, \mathcal{B}) \leftarrow_R \Psi; x^* \leftarrow_R \{0,1\}^k; y^* \leftarrow \mathcal{O}(x^*); x' \leftarrow_R \mathcal{A}^{\mathcal{O}, \mathcal{O}^{-1}_{\neq y^*}, \mathcal{B}}(y^*) : x' = x^*]$$

For notational convenience, we denote by $\widetilde{\mathsf{Expt}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k)$ the experiment

$$\widetilde{\mathsf{Expt}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k) : [(\mathcal{O}, \mathcal{B}) \leftarrow_R \Psi; x^* \leftarrow_R \{0,1\}^k; y^* \leftarrow \mathcal{O}(x^*); x' \leftarrow_R \mathcal{A}^{\mathcal{O}, \mathcal{O}^{-1}_{\neq y^*}, \mathcal{B}}(y^*)],$$

and we write $\widetilde{\mathsf{Adv}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k) = \mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_{\mathrm{R}}\Psi}\left[\ \mathsf{Adv}^{\mathtt{AOW}}_{\mathcal{O},\mathcal{A}^{\mathcal{O}},\mathcal{B}}(k)\ \right]$.

Assume towards a contradiction that $\widetilde{\mathsf{Adv}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k)$ is not negligible. Then, we show that we can construct another *computationally unbounded* adversary $\mathcal{S}$ that has query complexity at most $\tau_{\mathcal{A}}$ (and thus has polynomial query complexity) and has non-negligible advantage in the experiment $\mathsf{Expt}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{S}}(k)$, which will contradict Lemma 1. $\mathcal{S}$ is given 1^k and an image $\beta^* = \pi(\alpha^*)$ for randomly chosen $\alpha^* \in \{0,1\}^k$ and $\pi \in \mathsf{Perm}_k$, is given oracle access to π and $\pi^{-1}_{\neq\beta^*}$, and has to find α^*. The description of $\mathcal{S}^{\pi,\pi^{-1}_{\neq\beta^*}}(1^k,\beta^*)$ is as follows.

Description of $\mathcal{S}^{\pi,\pi^{-1}_{\neq\beta^*}}(1^k,\beta^* = \pi(\alpha^*))$*:* (Below, recall that $\mathcal{S}$ is computationally unbounded and thus can do very powerful things such as picking an injective function uniformly, etc.) Firstly, $\mathcal{S}$ picks its own m-bit-increasing injective oracle $\mathcal{O} \in \mathbb{O}_m$ uniformly at random[8]. Next, $\mathcal{S}$ defines a slightly modified m-bit-increasing injective oracle $\mathcal{O}_\pi$ into which $\mathcal{S}$'s oracle π is "embedded" as follows:

$$\mathcal{O}_\pi(\alpha) = \begin{cases} \mathcal{O}(\pi(\alpha)) & \text{if } |\alpha| = k \\ \mathcal{O}(\alpha) & \text{otherwise} \end{cases}$$

Note that this modification does not change the range of the m-bit-increasing injective oracle, i.e. $\mathcal{O}_\pi(\{0,1\}^*) = \mathcal{O}(\{0,1\}^*)$.

Then, $\mathcal{S}$ sets $y^* \leftarrow \mathcal{O}(\beta^*) = \mathcal{O}(\pi(\alpha^*)) = \mathcal{O}_\pi(\alpha^*)$, and runs $\mathcal{A}$ with input $(1^k, y^*)$. Hereafter, $\mathcal{S}$ starts simulating $\widetilde{\mathsf{Expt}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}$ for $\mathcal{A}$ in which the m-bit-increasing injective oracle is $\mathcal{O}_\pi$. We note that since π is not entirely known to $\mathcal{S}$, $\mathcal{O}_\pi$ for input length k (and the corresponding inversion) is not entirely known to $\mathcal{S}$. However, $\mathcal{S}$ knows all the knowledge about $\mathcal{O}$ (recall that $\mathcal{O}$ is picked by $\mathcal{S}$) and $\mathcal{S}$ can access to π and $\pi^{-1}_{\neq\beta^*}$, and thus $\mathcal{S}$ can perfectly simulate the responses of $\mathcal{O}$-queries and the inversion (i.e. $\mathcal{O}^{-1}_{\neq y^*}$-)queries from $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathcal{B}$-query (P, L, y), if (P, L, y) has been queried before, the same answer is used as a response. Otherwise, $\mathcal{S}$ responds as follows.

1. Firstly, $\mathcal{S}$ does Correctness check of P as is done in Step 1 in $\mathcal{B}$, by its computationally unbounded power (e.g. exhaustively checking if $\mathsf{P}^{\mathcal{O}'}$ implements a permutation over $\{0,1\}^{|y|}$ for all $\mathcal{O}' \in \mathbb{O}_m$)[9]. If the check fails, $\mathcal{S}$ returns $\perp$ to $\mathcal{A}$ after $|L| + \tau_{\mathsf{P}}$ steps from the point $\mathcal{A}$ made the $\mathcal{B}$-query, where τ_{P} is the maximum running time of P.
   Next, for each $\alpha \in L \cap \{0,1\}^k$, $\mathcal{S}$ issues α to π and obtains the corresponding value $\beta = \pi(\alpha)$. Then, for each response $\beta \in \pi(L \cap \{0,1\}^k)$ obtained in the above procedure, $\mathcal{S}$ computes $\gamma \leftarrow \mathcal{O}(\beta)$.
2. $\mathcal{S}$ generates the "modified" m-bit-increasing injective oracle $\widetilde{\mathcal{O}}_\pi$ (corresponding to $\mathcal{O}_\pi$) as is done in Step 2 of $\mathcal{B}$[10]. As mentioned earlier, $\mathcal{S}$ does not have

[8] Here, actually it is enough to pick $\mathcal{O}$ for input length up to $\tau_{\mathcal{A}}$, because $\mathcal{A}$ cannot issue an $\mathcal{O}$-query (and an $\mathcal{O}^{-1}_{\neq y^*}$-query) of length more than $\tau_{\mathcal{A}}$.

[9] Here, it is enough to check the correctness of P with oracles $\mathcal{O}$ that is defined up to input length τ_{P}, because P cannot issue an $\mathcal{O}$-query of length longer than τ_{P}.

[10] With similar reasons to what we mentioned in the previous footnotes, the new oracle $\widetilde{\mathcal{O}}_\pi$ is only needed to be defined for input length up to $\tau_{\mathcal{A}}$.

all the knowledge about the m-bit-increasing injective oracle $\mathcal{O}_\pi$ that $\mathcal{S}$ is using for simulating $\widetilde{\mathsf{Expt}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}$ for $\mathcal{A}$, because π is not entirely known to $\mathcal{S}$. However, for the strings $\alpha \in L \cap \{0,1\}^k$, the corresponding evaluations $\gamma = \mathcal{O}_\pi(\alpha) = \mathcal{O}(\pi(\alpha))$ are already calculated in the above step, and to generate $\widetilde{\mathcal{O}}_\pi$, no further knowledge about $\pi(\cdot)$ is needed. Note also that $\mathcal{O}$ itself was generated by $\mathcal{S}$, and the range of $\mathcal{O}_\pi$ is identical to that of $\mathcal{O}$ (i.e. $\mathcal{O}_\pi(\{0,1\}^*) = \mathcal{O}(\{0,1\}^*)$), which means that $\mathcal{S}$ can appropriately pick an injective function $g'_n : \{0,1\}^n \to (\{0,1\}^{n+m} \backslash \mathcal{O}_\pi(\{0,1\}^n))$ for each input length $n \in \mathbb{N}$.

3. $\mathcal{S}$ finds x such that $\mathsf{P}^{\widetilde{\mathcal{O}}_\pi}(x) = y$ and the corresponding query set $\mathsf{QS}_{\mathsf{P}^{\tilde{\mathcal{O}}_\pi}(x)}$ by using $\mathcal{S}$'s computationally unbounded power and the entire knowledge about $\widetilde{\mathcal{O}}_\pi$. Finally, $\mathcal{S}$ returns x and $\mathsf{QS}_{\mathsf{P}^{\tilde{\mathcal{O}}}(x)}$ to $\mathcal{A}$ after $|L| + \tau_{\mathsf{P}}$ steps from the point $\mathcal{A}$ made this $\mathcal{B}$-query.

When $\mathcal{A}$ terminates with an output, $\mathcal{S}$ outputs what $\mathcal{A}$ outputs as a candidate of the preimage of β^* under π and terminates.

The above completes the description of $\mathcal{S}$. Let us confirm the query complexity of $\mathcal{S}$. $\mathcal{S}$ issues at most one query to π (resp. $\pi^{-1}_{\neq\beta^*}$) for simulating a response to an $\mathcal{O}$-query (resp. $\mathcal{O}^{-1}_{\neq y^*}$-query), and at most $|L|$ queries to π for simulating a response to a $\mathcal{B}$-query. Recall that in the original experiment $\widetilde{\mathsf{Expt}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}$, each $\mathcal{B}$-query (P, L, y) is responded after $|L| + \tau_{\mathsf{P}}$ steps from the point $\mathcal{B}$ is invoked, where τ_{P} is the maximum running time of P (for $|y|$-bit input). These imply that the number of $\mathcal{S}$'s queries never exceeds $\mathcal{A}$'s running time. Since the running time of $\mathcal{A}$ is at most $\tau_{\mathcal{A}}$, $\mathcal{S}$'s query complexity is at most $\tau_{\mathcal{A}}$.

Moreover, as we have explained in the description of $\mathcal{S}$, $\mathcal{S}$ perfectly simulates the experiment $\widetilde{\mathsf{Expt}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k)$ for $\mathcal{A}$ so that the m-bit-increasing injective oracle is $\mathcal{O}_\pi$. Under this situation, if $\mathcal{A}$ succeeds in outputting a preimage $\mathcal{O}_\pi^{-1}(y^*) = \pi^{-1}(\mathcal{O}^{-1}(y^*))$, since $\beta^* = \mathcal{O}^{-1}(y^*)$, $\mathcal{S}$ also succeeds in outputting the preimage $\alpha^* = \pi^{-1}(\beta^*)$. Therefore, we have $\mathsf{Adv}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{S}}(k) = \widetilde{\mathsf{Adv}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k)$, which means that if $\widetilde{\mathsf{Adv}}^{\mathtt{AOW}}_{\mathbb{O}_m,\mathcal{A}}(k) = \mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_{\mathrm{R}}\Psi}\left[\ \mathsf{Adv}^{\mathtt{AOW}}_{\mathcal{O},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)\ \right]$ is non-negligible, so is $\mathsf{Adv}^{\mathtt{AOW}}_{\mathrm{RP},\mathcal{S}}(k)$. Since $\mathcal{S}$ has only polynomial query complexity (at most polynomial $\tau_{\mathcal{A}}$), this contradicts Lemma 1, and thus $\mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_{\mathrm{R}}\Psi}\left[\ \mathsf{Adv}^{\mathtt{AOW}}_{\mathcal{O},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)\ \right]$ must be negligible. This completes the proof of Lemma 4. □

3.3 Breaking Any Candidate of One-Way Permutation with $\mathcal{B}$

In this subsection, we show that for any candidate of a OWP that is constructed using an m-bit-increasing injective oracle $\mathcal{O} \in \mathbb{O}_m$, there is a *perfect* inverter that has access to $\mathcal{O}$ and $\mathcal{B}$, where $(\mathcal{O}, \mathcal{B})$ are chosen according to Ψ defined in Section 3.1.

Lemma 5. *For any oracle PPTA* P*, if* $\mathsf{P}^{\mathcal{O}'}$ *implements a permutation for all* $(\mathcal{O}', \mathcal{B}') \in [\Psi]$*, then there is an oracle PPTA* $\mathcal{A}$ *such that* $\mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_R\Psi}[\mathsf{Adv}^{\mathtt{OW}}_{\mathsf{P}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)] = 1$.

Proof. Fix an oracle PPTA P such that $\mathsf{P}^{\mathcal{O}'}$ implements a permutation for all $\mathcal{O}' \in \mathbb{O}_m$. Let $\tau_{\mathsf{P}} = \tau_{\mathsf{P}}(k)$ be the maximum running time of P on input k-bit strings. Since P is a PPTA, τ_{P} is a polynomial.

The expectation (over the choice of $(\mathcal{O}, \mathcal{B})$) of the advantage of an oracle PPTA adversary $\mathcal{A}$ attacking the one-wayness of $\mathsf{P}^{\mathcal{O}}$ can be written as:

$$\begin{aligned}&\mathop{\mathbf{E}}_{(\mathcal{O},\mathcal{B})\leftarrow_{\mathtt{R}}\Psi}\left[\ \mathsf{Adv}^{\mathtt{OW}}_{\mathsf{P}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)\ \right]\\ &= \Pr\left[(\mathcal{O},\mathcal{B}) \leftarrow_{\mathtt{R}} \Psi; x^* \leftarrow_{\mathtt{R}} \{0,1\}^k; y^* \leftarrow \mathsf{P}^{\mathcal{O}}(x^*); x' \leftarrow_{\mathtt{R}} \mathcal{A}^{\mathcal{O},\mathcal{B}}(y^*) : x' = x^*\right]\end{aligned}$$

For notational convenience, we denote by $\widetilde{\mathsf{Expt}}^{\mathtt{OW}}_{\mathsf{P},\mathcal{A}}(k)$ the experiment

$$\widetilde{\mathsf{Expt}}^{\mathtt{OW}}_{\mathsf{P},\mathcal{A}}(k) : [(\mathcal{O},\mathcal{B}) \leftarrow_{\mathtt{R}} \Psi; x^* \leftarrow_{\mathtt{R}} \{0,1\}^k; y^* \leftarrow \mathsf{P}^{\mathcal{O}}(x^*); x' \leftarrow_{\mathtt{R}} \mathcal{A}^{\mathcal{O},\mathcal{B}}(y^*)]$$

($\widetilde{\mathsf{Expt}}^{\mathtt{OW}}_{\mathsf{P},\mathcal{A}}(k)$ includes the sampling of oracles $(\mathcal{O}, \mathcal{B})$ according to Ψ), and we write $\widetilde{\mathsf{Adv}}^{\mathtt{OW}}_{\mathsf{P},\mathcal{A}}(k) = \mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_{\mathtt{R}}\Psi}\left[\ \mathsf{Adv}^{\mathtt{OW}}_{\mathsf{P}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)\ \right]$.

We show that there is an oracle PPTA adversary $\mathcal{A}$ such that $\widetilde{\mathsf{Adv}}^{\mathtt{OW}}_{\mathsf{P},\mathcal{A}}(k) = 1$. That is, $\mathcal{A}$ is given 1^k and $y^* \in \{0,1\}^k$, has access to two oracles $(\mathcal{O}, \mathcal{B})$, and will always find the preimage $x^* \in \{0,1\}^k$ of y^* under the permutation $\mathsf{P}^{\mathcal{O}}$. The description of $\mathcal{A}^{\mathcal{O},\mathcal{B}}(1^k, y^*)$ is as follows:

Description of $\mathcal{A}^{\mathcal{O},\mathcal{B}}(1^k, y^*)$: Firstly, $\mathcal{A}$ generates an empty list $L_1 = \emptyset$. Then for $1 \le i \le \tau_{\mathsf{P}} + 1$, $\mathcal{A}$ does the following.

Iterations for $1 \le i \le \tau_{\mathsf{P}} + 1$: $\mathcal{A}$ issues a $\mathcal{B}$-query (P, L_i, y^*). Let x_i and $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i)}$ be the response from $\mathcal{B}$, where $\widetilde{\mathcal{O}}_i$ is the modified m-bit-increasing injective oracle generated in the step 2 of $\mathcal{B}$ in the i-th iteration, x_i is a string such that $\mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i) = y^*$, and $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i)}$ is the corresponding query set. (Since $\mathsf{P}^{\mathcal{O}'}$ implements a permutation for all oracles $\mathcal{O}' \in \mathbb{O}_m$, $\mathcal{A}$'s $\mathcal{B}$-query is always valid.) Then, $\mathcal{A}$ computes $y_i \leftarrow \mathsf{P}^{\mathcal{O}}(x_i)$. If $y_i = y^*$, $\mathcal{A}$ terminates with output x_i as the preimage of y^* under the permutation $\mathsf{P}^{\mathcal{O}}$. Otherwise (i.e. $y_i \neq y^*$), $\mathcal{A}$ updates the list by $L_{i+1} \leftarrow L_i \cup \mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i)}$, and goes to the next iteration.

If $\mathcal{A}$ does not terminate after $\tau_{\mathsf{P}} + 1$ iterations, $\mathcal{A}$ simply gives up and aborts.

The above completes the description of $\mathcal{A}$. We first confirm the query complexity of $\mathcal{A}$. Clearly, the query complexity becomes maximum if $\mathcal{A}$ performs all $\tau_{\mathsf{P}} + 1$ iterations without discovering the preimage. Thus we consider this case. In the i-th iteration, $\mathcal{A}$ makes one $\mathcal{B}$-query (P, L_i, y^*) which is counted as $|L_i| + \tau_{\mathsf{P}}$ queries, and executes $\mathsf{P}^{\mathcal{O}}$ once during which at most τ_{P} queries are made to $\mathcal{O}$. Therefore, in the i-th iteration, the query complexity can increase by at most $|L_i| + 2\tau_{\mathsf{P}}$. Moreover, recall that in each iteration the list L_i is updated to $L_{i+1} \leftarrow L_i \cup \mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i)}$. Since $|L_1| = 0$ and it is always the case that $|\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i)}| \le \tau_{\mathsf{P}}$, we have $|L_i| \le (i-1)\tau_{\mathsf{P}} \le \tau_{\mathsf{P}}^2$ for $1 \le i \le \tau_{\mathsf{P}} + 1$. This implies that in the i-th iteration, the query complexity can increase by at most

$|L_i| + 2\tau_{\mathsf{P}} \leq \tau_{\mathsf{P}}^2 + 2\tau_{\mathsf{P}} = \tau_{\mathsf{P}}(\tau_{\mathsf{P}} + 2)$. Thus, after $\tau_{\mathsf{P}} + 1$ iterations, $\mathcal{A}$'s total query complexity is at most $\tau_{\mathsf{P}}(\tau_{\mathsf{P}} + 1)(\tau_{\mathsf{P}} + 2)$, which is polynomial in k. Therefore, the number of steps incurred by the use of oracles $(\mathcal{O}, \mathcal{B})$ is at most polynomial, which means $\mathcal{A}$ works in polynomial time.

Next, we show that $\mathcal{A}$ can always find the preimage x^* of y^* under $\mathsf{P}^{\mathcal{O}}$. We show the following key claim, which states that in each iteration $\mathcal{A}$ either finds the preimage x^* or finds at least one "undiscovered" $\mathcal{O}$-query $\alpha \in \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$, i.e. a query to $\mathcal{O}$ that is made by P during a computation of $\mathsf{P}^{\mathcal{O}}(x^*)$ but is not contained in L_i.

Claim 1. For every $i \in \{1, \ldots, \tau_{\mathsf{P}}\}$, at least one of the following two is true:

(a) $x_i = x^*$
(b) the set $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i$ contains at least one $\alpha \in \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$

Proof of Claim 1. Assume towards a contradiction that we have both $\overline{\text{(a)}}$ $x_i \neq x^*$ and $\overline{\text{(b)}}$ no $\alpha \in \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$ is contained in the set $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i$. The latter means that the set $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$ and the set $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i$ are disjoint.

Consider the following "hybrid" oracle $\widehat{\mathcal{O}}$ defined by:

$$\widehat{\mathcal{O}}(\alpha) = \begin{cases} \widetilde{\mathcal{O}}_i(\alpha) & \text{if } \alpha \in (\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i) \\ \mathcal{O}(\alpha) & \text{otherwise} \end{cases}$$

We argue $\widehat{\mathcal{O}} \in \mathbb{O}_m$, i.e., $\widehat{\mathcal{O}}$ is also a possible instance of an m-bit-increasing injective oracle. For notational convenience, we write $A = (\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i)$. Firstly, it is clear that $\widehat{\mathcal{O}}$ is m-bit-increasing, and is injective for each of the subdomains A and $\{0,1\}^* \backslash A$, because so are $\mathcal{O}$ and $\widetilde{\mathcal{O}}_i$. Secondly, recall that the sets $\widetilde{\mathcal{O}}_i(\{0,1\}^* \backslash L_i)$ and $\mathcal{O}(\{0,1\}^*)$ are always disjoint (see the explanation after the description of $\mathcal{B}$ in Section 3.1). Since we trivially have $A \subseteq (\{0,1\}^* \backslash L_i)$ and $(\{0,1\}^* \backslash A) \subseteq \{0,1\}^*$, the set $\widehat{\mathcal{O}}(A) = \widetilde{\mathcal{O}}_i(A)$ and the set $\widehat{\mathcal{O}}(\{0,1\}^* \backslash A) = \mathcal{O}(\{0,1\}^* \backslash A)$ are also disjoint, which means that there is no pair $(\alpha, \alpha') \in A \times (\{0,1\}^* \backslash A)$ such that $\widehat{\mathcal{O}}(\alpha) = \widehat{\mathcal{O}}(\alpha')$. These imply $\widehat{\mathcal{O}} \in \mathbb{O}_m$.

Therefore, $\mathsf{P}^{\widehat{\mathcal{O}}}$ also implements a permutation.

Next, we confirm the property of this hybrid oracle $\widehat{\mathcal{O}}$. The condition $\overline{\text{(b)}}$ and the definitions of $\mathcal{O}$, $\widetilde{\mathcal{O}}_i$, and $\widehat{\mathcal{O}}$ imply the following relations among these oracles:

(1): $\widehat{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha) = \widetilde{\mathcal{O}}_i(\alpha)$ for all $\alpha \in L_i$
(2): $\widehat{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha)$ for all $\alpha \in \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$
(3): $\widehat{\mathcal{O}}(\alpha) = \widetilde{\mathcal{O}}_i(\alpha)$ for all $\alpha \in \mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i$

where (2) is due to the condition $\overline{(b)}$, i.e. the set $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$ and the set $\mathsf{QS}_{\mathsf{P}^{\widetilde{\mathcal{O}}_i(x_i)}} \backslash L_i$ are disjoint. On the one hand, (1) and (2) imply $\widehat{\mathcal{O}}(\alpha) = \mathcal{O}(\alpha)$ for all $\alpha \in \mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)}$, which in turn implies $\mathsf{P}^{\widehat{\mathcal{O}}}(x^*) = \mathsf{P}^{\mathcal{O}}(x^*) = y^*$. On the

other hand, (1) and (3) imply $\widehat{\mathcal{O}}(\alpha) = \widetilde{\mathcal{O}}_i(\alpha)$ for all $\alpha \in \mathsf{QS}_{\mathsf{P}^{\bar{\mathcal{O}}_i(x_i)}}$, which in turn implies $\mathsf{P}^{\widehat{\mathcal{O}}}(x_i) = \mathsf{P}^{\widetilde{\mathcal{O}}_i}(x_i) = y^*$.

In summary, we have $\mathsf{P}^{\widehat{\mathcal{O}}}(x^*) = \mathsf{P}^{\widehat{\mathcal{O}}}(x_i) = y^*$, while we also have $x^* \neq x_i$ by the condition $\overline{(\mathrm{a})}$. This is a contradiction because $\mathsf{P}^{\widehat{\mathcal{O}}}$ is a permutation, which cannot have a collision. Therefore, our hypothesis must be false, and (a) or (b) must be true. This completes the proof of Claim 1. □

Due to Claim 1, in the i-th iteration (for $1 \leq i \leq \tau_{\mathsf{P}}$) $\mathcal{A}$ finds the preimage x^* and stops, or the set $\mathsf{QS}_{\mathsf{P}^{\bar{\mathcal{O}}_i(x_i)}} \backslash L_i$ contains at least one α which is contained in $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)}$ but is not contained in L_i. In the latter case, since the list L_i is updated as $L_{i+1} \leftarrow L_i \cup \mathsf{QS}_{\mathsf{P}^{\bar{\mathcal{O}}_i(x_i)}}$, the size of the set of "undiscovered queries" $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \backslash L_i$ decreases at least by one in each iteration. Recall that $|\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)}| \leq \tau_{\mathsf{P}}$. Therefore, when $\mathcal{A}$ is executed, $\mathcal{A}$ finds the preimage x^* and stops within τ_{P} iterations, or after τ_{P} iterations we have $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)} \subseteq L_{\tau_{\mathsf{P}}+1}$. In the latter case, by Lemma 3, the $(\tau_{\mathsf{P}}+1)$-th $\mathcal{B}$-query $(\mathsf{P}, L_{\tau_{\mathsf{P}}+1}, y^*)$ results in x^* (and $\mathsf{QS}_{\mathsf{P}^{\mathcal{O}}(x^*)}$).

These imply that $\mathcal{A}$ always outputs x^* within $\tau_{\mathsf{P}}+1$ iterations, which means that we have $\widetilde{\mathsf{Adv}}^{\mathtt{OW}}_{\mathsf{P},\mathcal{A}} = \mathbf{E}_{(\mathcal{O},\mathcal{B}) \leftarrow_{\mathrm{R}} \Psi}[\mathsf{Adv}^{\mathtt{OW}}_{\mathsf{P}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)] = 1$. This completes the proof of Lemma 5. □

3.4 Putting Everything Together

Here, we show the formal proof of Theorem 1.

Proof of Theorem 1. Let $m > 0$ be an integer. We use the distribution Ψ of oracles $(\mathcal{O}, \mathcal{B})$ defined in Section 3.1 for Lemma 2. Then, by definition any $\mathcal{O}$ where $(\mathcal{O}, \mathcal{B}) \in [\Psi]$ is m-bit-increasing and injective, and thus the assumption (1) in Lemma 2 is satisfied. Moreover, the assumptions (2) and (3) are satisfied due to Lemma 4 in Section 3.2 and Lemma 5 in Section 3.3, respectively. Since the distribution Ψ satisfies all the assumptions in Lemma 2, it follows that there is no fully black-box construction of a OWP from an m-bit-increasing injective AOWF. This statement holds for any integer $m > 0$. This completes the proof of Theorem 1. □

An immediate corollary of Theorem 1 is the following:

Corollary 1. *For any integer $m > 0$, there is no fully black-box construction of a OWP from a 2^m-regular OWF.*

Proof. Let $m > 0$. Suppose $f : \{0,1\}^n \to \{0,1\}^{n+m}$ is an m-bit-increasing injective OWF. From f, construct a new function $g : \{0,1\}^{n+m} \to \{0,1\}^{n+m}$ by $g(x||x') = f(x)$ where $|x| = n$ and $|x'| = m$ (i.e, g ignores the last m-bits of the input). Then, this function g is a 2^m-regular OWF (for security parameter 1^n) as long as g is an m-bit-increasing injective OWF (for security parameter 1^n). Trivially, the construction of g from f and the security proof are black-box, and thus for any integer $m > 0$, there is a fully black-box construction of a 2^m-regular OWF from an m-bit-increasing injective OWF. Since a fully-black-box

construction of a primitive from another primitive is a transitive relation, we have the claimed result. □

Remark. In Theorem 1, the "stretch" m of the building block (injective) AOWFs has been treated as a constant. However, our negative results can be extended to the case in which $m = m(k)$ is a function of the input-length k (i.e. security parameter), as long as $m(k) > 0$ for all $k > 0$. m in Corollary 1 can be a function of the security parameter as well.

4 Restricted Type of Black-box Separations among Injective OWFs

In this section, we show that Theorem 1 can be extended to show the impossibility of a restricted type of black-box constructions, which we call *range-invariant* fully black-box constructions, of an injective OWF from another injective OWF.

Definition 3. *Let $\ell, m \geq 0$ be integers. We say there exists a* range-invariant *fully black-box construction of an ℓ-bit-increasing injective OWF from an m-bit-increasing injective AOWF, if there exist oracle PPTAs G and $\mathcal{R}$ such that:*

(Correctness) *For all m-bit-increasing injective functions f (of arbitrary complexity), G^f is an ℓ-bit-increasing injective function and* has the same range[11].

(Security) *For all m-bit-increasing injective functions f and all algorithms $\mathcal{A}$ (where f and $\mathcal{A}$ are of arbitrary complexity), if $\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{G}^f,\mathcal{A}}(k)$ is non-negligible, so is $\mathsf{Adv}^{\mathsf{AOW}}_{f,\mathcal{R}^f,\mathcal{A}}(k)$.*

In other words, the range of the constructed ℓ-bit-increasing injective function G^f depends solely on G, and independent of the building block f.

Now, we state our black-box separation result among injective OWFs.

Theorem 2. *For any integer pair (ℓ, m) satisfying $m > \ell \geq 0$, there is no range-invariant fully black-box construction of an ℓ-bit-increasing injective OWF from an m-bit-increasing injective AOWF.*

Note that a permutation is a 0-bit-increasing injective function. Moreover, any construction of a permutation from other primitives has the same range, namely, a set of all strings, and thus is inherently range-invariant. Therefore, Theorem 1 is the special case of Theorem 2 in which $\ell = 0$.

Since Theorem 2 can be proved very similarly to Theorem 1, below we only show the outline of the proof, highlighting the different points from the proof of Theorem 1 we need to care.

As in the case of Theorem 1, in order to show Theorem 2, we can use the generalized version of Lemma 2 (which can be proved similarly to Lemma 2):

[11] That is, G^f and $\mathsf{G}^{f'}$ have the same range for any f, f' implementing m-bit-increasing injective functions.

Lemma 6. *Let ℓ and m be integers satisfying $m > \ell \geq 0$. Assume that there exists a distribution Ψ of an oracle pair $(\mathcal{O}, \mathcal{B})$ that satisfies the following three conditions:*

(1) and (2): Same as Lemma 2.
(3): For any oracle PPTA G, if $\mathsf{G}^{\mathcal{O}'}$ implements an ℓ-bit-increasing injective function and has the same range *for all $(\mathcal{O}', \mathcal{B}') \in [\Psi]$, then there exists an oracle PPTA $\mathcal{A}$ such that $\mathbf{E}_{(\mathcal{O},\mathcal{B})\leftarrow_R\Psi}[\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{G}^{\mathcal{O}},\mathcal{A}^{\mathcal{O},\mathcal{B}}}(k)] = 1$.*

Then, there is no range-invariant fully black-box construction of an ℓ-bit-increasing injective OWF from an m-bit-increasing injective AOWF.

We remark that the range-invariance condition in Theorem 2 is due to the additional condition on the range of $\mathsf{G}^{\mathcal{O}}$ in (3) in the above lemma. The reason why we need this additional condition on the range of G is to ensure that if a string y is in the range of $\mathsf{G}^{\mathcal{O}}$ for some $\mathcal{O} \in \mathbb{O}_m$, then y is also in the range of $\mathsf{G}^{\mathcal{O}'}$ for any $\mathcal{O}' \in \mathbb{O}_m$, and thus the preimage of y under $\mathsf{G}^{\mathcal{O}'}$ always exists.

To use Lemma 6 to prove Theorem 2, it remains to show the definition of an oracle pair $(\mathcal{O}, \mathcal{B})$ and their distibution Ψ, and the conditions (1) to (3) of Lemma 6 (i.e. the statements corresponding to Lemmas 4 and 5). For $\mathcal{O}$, we again use an m-bit-increasing injective oracle. The breaking oracle $\mathcal{B}$ needs to be modified slightly: in Step 1, $\mathcal{B}$ checks if a given description of an algorithm G implements an ℓ-bit-increasing injective function and has the same range for all m-bit-increasing injective oracles, and also checks if a given string y belongs to the range of $\mathsf{G}^{\mathcal{O}}$. If G and y pass the check, then it is guaranteed that there always exist x satisfying $y^* = \mathsf{G}^{\mathcal{O}}(x^*) = \mathsf{G}^{\widetilde{\mathcal{O}}}(x)$ where $\widetilde{\mathcal{O}}$ is the modified oracle generated in the step 2 of $\mathcal{B}$ and x is a string found in the step 3 of $\mathcal{B}$. (Without the checks, it is possible that such x does not exist, which we want to avoid.) The distribution Ψ is also naturally defined.

The statement corresponding to Lemma 4, which roughly states that a random m-bit-increasing injective function is adaptively one-way even against adversaries with access to $\mathcal{B}$ that is modified as above, can be similarly proved as Lemma 4.

To show the statement corresponding to Lemma 5, which roughly states that no ℓ-bit-increasing injective function $\mathsf{G}^{\mathcal{O}}$ with the "range-invariance" can be one-way if $\mathcal{B}$ is available, we need to rely on the additional assumption on the range of G, especially when showing the statement analogous to Claim 1. Recall that in order to prove Claim 1 by contradiction we need a property that under several different oracles, namely the original oracle $\mathcal{O}$, the modified oracle $\widetilde{\mathcal{O}}_i$, and the "hybrid oracle" $\widehat{\mathcal{O}}$, P always implements a permutation and causes a situation in which $y^* = \mathsf{P}^{\widehat{\mathcal{O}}}(x^*) = \mathsf{P}^{\widehat{\mathcal{O}}}(x_i)$ and $x^* \neq x_i$. In order for the same strategy to work, the challenge instance y^* needs to belong to the range of the ℓ-bit-increasing injective functions $\mathsf{G}^{\mathcal{O}}$, $\mathsf{G}^{\widetilde{\mathcal{O}}_i}$, and $\mathsf{G}^{\widehat{\mathcal{O}}}$. Due to the assumption we have made, however, it is guaranteed that y^* always belong to the range of these ℓ-bit-increasing injective functions, and we can cause a situation in which $y^* = \mathsf{G}^{\widehat{\mathcal{O}}}(x^*) = \mathsf{G}^{\widehat{\mathcal{O}}}(x_i)$ and $x^* \neq x_i$.

Acknowledgement

The authors would like to thank Jacob Schuldt for his helpful discussion. The authors also would like to thank anonymous reviewers of TCC 2011 for their invaluable comments.

References

1. M. Blum and R. Impabliazzo. Generic oracles and oracle classes (extended abstract). *FOCS 1987*, pp. 118–126, 1987.
2. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Computing*, 13(4):850–864, 1984.
3. 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.
4. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
5. O. Goldreich, L.A. Levin, and N. Nisan. On constructing 1-1 one-way functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 2(29), 1995.
6. J. Hartmanis and L.A. Hemachandra. One-way functions and the nonisomorphism of NP-complete sets. *Theor. Comput. Sci.*, 81(1):155–183, 1991.
7. 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.
8. C.-Y. Hsiao and L. Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? *CRYPTO 2004*, LNCS 3152, pp. 92–105, 2004.
9. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. *STOC 1989*, pp. 44–61, 1989.
10. J. Kahn, M. Saks, and C. Smyth. A dual version of Reimer's inequality and a proof of Rudich's conjecture. *CoCo 2000*, pp. 98–103, 2000.
11. O. Pandey, R. Pass, and V. Vaikuntanathan. Adaptive one-way functions and applications. *CRYPTO 2008*, LNCS 5157, pp. 57–74, 2008.
12. O. Reingold. On black-box separations in cryptography, 2006. One of Tutorials in TCC 2006. A slide file available at http://research.ihost.com/tcc06/slides/Reingold-tutorial.ppt.
13. O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. *TCC 2004*, LNCS 2951, pp. 1–20, 2004.
14. S. Rudich. Limits on the provable consequences of one-way functions, 1988. PhD thesis, University of California at Berkeley.
15. D.R. Simon. Finding collision on one-way street: Can secure hash functions be based on general assumptions? *EUROCRYPT 1998*, LNCS 1403, pp. 334–345, 1998.
16. G. Tardos. Query complexity, or why is it difficult to separate $NP^A \cup coNP^A$ from P^A by random oracles A? *Combinatorica*, 9(4):385–392, 1989.
17. Y. Vahlis. Two is a crowd? a black-box separation of one-wayness and security under correlated inputs. *TCC 2010*, LNCS 5978, pp. 165–182, 2010.
18. A.C. Yao. Theory and application of trapdoor functions. *FOCS 1982*, pp. 80–91, 1982.