Enhancements Are Blackbox Non-Trivial: Impossibility of Enhanced Trapdoor Permutations from Standard Trapdoor Permutations

Mohammad Hajiabadi

University of California Berkeley and University of Virginia.
mdhajiabadi@berkeley.edu

Abstract. Trapdoor permutations (TDP) are a fundamental primitive in cryptography. Several variants of this notion have emerged as a result of different applications. However, it is not clear whether these variants can be based on the standard notion of TDPs.

We study the question of whether enhanced trapdoor permutations can be based on classical trapdoor permutations. The main motivation of our work is in the context of existing TDP-based constructions of oblivious transfer and non-interactive zero knowledge protocols, which require enhancements to the classical TDP notion. We prove that these enhancements are non-trivial, in the sense that there does not exist fully blackbox constructions of enhanced TDPs from classical TDPs.

On the technical side, we show that the enhanced TDP security of any construction in the random TDP oracle world can be broken via a polynomial number of queries to the TDP oracle as well as a weakening oracle, which provides inversion with respect to randomness. We also show that the standard one-wayness of the random TDP oracle stays intact in the presence of this weakening oracle.

1 Introduction

Trapdoor permutations (TDPs) [RSA78,Rab79] are a family of permutations, where each permutation in the family is easy to compute given the underlying index key, and also easy to invert given a corresponding trapdoor key. The classical notion of one-wayness for TDPs states that it is hard to invert a randomly chosen permutation from the family on a random image. While classical TDPs suffice for many applications, such as public-key encryption (PKE) [Yao82], parallel constructions of pseudorandom synthesizers [NR99], etc, for certain applications we need to strengthen this basic one-wayness notion. The main reason is that in protocols in which TDPs are used, the adversary may sometimes have some side information about the underlying image element, which may give her some advantage.

Technically, TDPs come with a sampling algorithm S, which, on input an index key IK and random coins R, outputs an element from the domain $\mathsf{Dom}_{\mathrm{ik}}$

of the permutation $\mathsf{E}(\mathrm{IK}, \cdot)$. We call a TDP *enhanced* if it is hard to find the pre-image of a random image element $\mathrm{Y} := \mathsf{S}(\mathrm{IK}; \mathrm{R})$ even if the inverter is given the randomness R (along with IK). Intuitively, enhanced TDPs allow a sampler, given only the underlying index key, to sample an image point obliviously to its pre-image: if we sample $\mathrm{Y} = \mathsf{S}(\mathrm{IK}; \mathrm{R})$ for a random R, then even given R, we are still oblivious to the corresponding pre-image of Y.

To see when this need of enhancement arises, consider the classical construction of hones-but-curious oblivious transfer (OT) protocols [EGL82,GMW87]. In this setting, a receiver $\mathsf{Alice}(\mathrm{b}, \cdot)$ with input bit b wishes to secretly learn the message $\mathrm{m_b}$ of Bob's two messages $(\mathrm{m}_0, \mathrm{m}_1)$. She does so by sending two image elements Y_1 and Y_2 of a TDP $\mathsf{E}(\mathrm{IK}, \cdot)$, where IK's trapdoor key is only known to Bob, in such a way that Alice knows the pre-image of $\mathrm{Y_b}$ but not of $\mathrm{Y_{1-b}}$. She does so by sampling $\mathrm{Y_{1-b}}$ *obliviously* and by sampling $\mathrm{Y_b}$ by applying $\mathsf{E}(\mathrm{IK}, \cdot)$ on a random domain element X. Bob sends to Alice encryptions c_1 and c_2 of the two bits m_0 and m_1 under the standard TDP-based PKE construction, using Y_0 and Y_1 as the 'encoded randomness.' Alice can open $\mathrm{c_b}$ to recover $\mathrm{Y_b}$. In order to ensure privacy for Bob, we need to assume that the underlying TDP is enhanced one-way.

The need for strengthening the notion of TDPs was first discovered by Bellare and Yung [BY93], noting that the previous TDP-based non-interactive zero knowledge (NIZK) construction in [FLS90] requires the set of valid permutations to be *certifiable*. Goldreich [Gol04] was the first to realize the need for enhanced TDPs in the context of OT constructions. It was also later discovered that for the TDP-based non-interactive zero knowledge (NIZK) protocol [FLS90] the zero-knowledge property relies on the TDP being *doubly enhanced* [Gol11], in addition to the certifiability property. Informally, doubly-enhanced TDPs are enhanced TDPs that provide the feature that given an index key IK it is possible to sample random coins R_y together with the pre-image of $\mathsf{S}(\mathrm{IK}, \mathrm{R}_y)$. As noted in [Gol11,GR13] the main reason these requirements were not noticed earlier is because TDPs had implicitly been assumed to be permutations over $\{0,1\}^\kappa$ (or over domains which enable trivial sampling algorithms). While these idealized TDPs are doubly enhanced, we do not have any candidate constructions for them.

Faced with this difficulty, Haitner [Hai04] gives a more complicated OT protocol which works with respect to any classical TDP with *dense domains*. It is not however clear whether such TDPs can be built from classical one-way TDPs.

In summary, the possibility of basing OT or NIZK on classical TDPs remains unknown. One way to address these is to investigate whether enhanced TDPs can be constructed from standard TDPs.

1.1 Our Result and Discussion

We take a first step toward understanding the relationships between various notions of TDPs. Our main result shows that enhanced TDPs cannot be constructed from classical TDPs in a fully blackbox way (in the taxonomy of [RTV04]).

We give an overview of our result and techniques in Section 1.2. In what follows, we discuss the significance of our work.

TDPs are rather coarse as a primitive, since the set of assumptions from which TDPs can be built is relatively small, being limited to factoring-related assumptions [RSA78,Rab79] and obfuscation-based assumptions [BPW16]. Also, variants of the popular RSA and Rabin TDPs (see e.g., [KKM12]) as well as variants of iO-based TDPs are already doubly enhanced [GR13,BPW16].[1] Given this state of affairs, one may ask about the motivations of this work. We provide the following motivations.

- In a similar vein, Hsiao and Reyzin [HR04] draw attention to the distinction between secret-coin collision resistant hash functions (CRHF) and public-coin CRHF by showing that the latter cannot be constructed from the former in a blackbox way. Prior to their work, these two notions had been deemed to be equivalent. In some sense, our result shows that a similar situation relating to public-versus-secret coins holds in the TDP setting as well, emphasizing the need of rigorously showing which version is required in each application and achieved by a future construction.
- Goldreich and Rothblum [Gol11] show that the TDP-based PKE construction, when instantiated with enhanced TDPs, offer properties, such as *oblivious ciphertext samplability*, that have useful applications. This gives applications beyond the OT and NIZK settings, and serves as another motivation for studying the possibility of basing enhanced TDPs on standard TDPs.
- TDPs turn out to be tricky objects to define, because after several decades of research, still new aspects of this primitive are revealed, which turn out to be required by some applications, but which were overlooked before. (See for example the recent work of [CL17]). Faced with this landscape of TDP with various properties, from a theoretical point of view, one would like to understand to what extent these notions relate to each other, elucidating and simplifying the landscape.

Open problems. Our work leads to the following open problem: is it possible to prove that OT cannot be based on standard TDPs in a blackbox way? Since our work removes one path toward this goal, our techniques may be useful in an eventual separation (if at all possible).

Other related work. There is a rich body of research on understanding the limitations of TDPs. In particular, we know that TDPs cannot be used in a blackbox way to construct two-message statistically-hiding commitments [Fis02], identity-based encryption [BPR$^+$08], correlated-secure trapdoor functions [Vah10] and verifiable random functions [FS12]. To the best of our knowledge, all these separations still hold even if the base TDP is doubly enhanced. Haitner et al. [HHRS07] give lower-bounds on the round complexity of statistically-hiding commitments making blackbox use of TDPs. There is a positive construction of

[1] The TDP construction in [BPW16] does not satisfy doubly-enhanced one-wayness, but a relaxed version of it, which nevertheless suffices for their respective application.

TDPs from indistinguishability obfuscations (IO) and one-way functions [BPW16], which is not so-called *domain invariant*. The result of Asharov and Segev [AS16] justifies this, showing that current non-blackbox iO-based techniques are not sufficient to give us domain-invariant TDPs.

Gertner et al [GKM$^+$00] show that TDPs cannot be built from trapdoor functions (TDFs) in a blackbox way. Their result is incomparable to ours (and their techniques are also different), because their base primitive is TDFs, and in their proof they make essential of the fact that the domain of a TDF can be different from the range. Our result in contrast is about a separation between two notions of TDPs.

1.2 Technical Overview

As common in blackbox impossibility results, we will prove our impossibility by giving an oracle relative to which the base primitive exists, but not the target primitive. Consider a random TDP oracle $\mathbf{O} = (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$ with the following sub-oracles. The key-generation oracle $\mathbf{g} : \{0,1\}^\kappa \mapsto \{0,1\}^\kappa$ is a random injective function mapping a trapdoor key tk to an index key ik. The evaluation oracle $\mathbf{e}(\mathrm{ik}, \cdot) : \{0,1\}^{5\kappa} \mapsto \{0,1\}^{5\kappa}$ on an index key ik is defined over all elements in $\{0,1\}^{5\kappa}$; however, $\mathbf{e}(\mathrm{ik}, \cdot)$ is a permutation only over a *sparse* subset $\mathsf{Dom}_{\mathrm{ik}}$ of $\{0,1\}^{5\kappa}$, where $|\mathsf{Dom}_{\mathrm{ik}}| = 2^\kappa$ (hence the name sparseness). That is, we have $\mathbf{e}(\mathrm{ik}, \mathsf{Dom}_{\mathrm{ik}}) = \mathsf{Dom}_{\mathrm{ik}}$.

The sampling oracle $\mathbf{s}(\mathrm{ik}, \cdot)$ is a random injective function which allows us to sample from $\mathsf{Dom}_{\mathrm{ik}}$: given a string $\mathrm{r} \in \{0,1\}^\kappa$, $\mathbf{s}(\mathrm{ik}, \mathrm{r})$ returns an element in $\mathsf{Dom}_{\mathrm{ik}}$. Finally, the inversion oracle $\mathbf{d}$ is defined in a manner consistent with the other oracles.

The oracle $\mathbf{O}$ *by itself is too strong.* Such a randomly chosen oracle $\mathbf{O}$ is overly strong, satisfying already all enhanced forms of one-wayness. Thus, it cannot be taken as is for deriving an impossibility. To address this problem, we will add a weakening oracle $\mathbf{u}$, which does not harm the standard one-wayness of $\mathbf{O}$, but which helps us break the enhanced one-wayness of any blackbox construction $(\mathsf{G}^{\mathbf{O}}, \mathsf{S}^{\mathbf{O}}, \mathsf{E}^{\mathbf{O}}, \mathsf{D}^{\mathbf{O}})$. Our blackbox separation will then follow from this.

Intuition behind the weakening oracle $\mathbf{u}$. As a starter, suppose we are content with $\mathbf{u}$ only breaking the enhanced one-wayness of $\mathbf{O}$ (as opposed to any TDP construction from $\mathbf{O}$). Thus, $\mathbf{u}$ should provide help for an inverter who has the randomness of the challenge image. A natural choice for $\mathbf{u}$ would be the following: on input $\mathbf{u}(\mathrm{ik}, \mathrm{r})$, let $\mathrm{y} := \mathbf{s}(\mathrm{ik}, \mathrm{r})$ and return $\mathrm{x} \in \mathsf{Dom}_{\mathrm{ik}}$ for which we have $\mathbf{e}(\mathrm{ik}, \mathrm{x}) = \mathrm{y}$.

Indeed, the above oracle $\mathbf{u}$ breaks the enhanced one-wayness of $\mathbf{O}$. We can also see that the oracle $\mathbf{u}$ does not harm the standard one-wayness of $\mathbf{O}$. This is because of the sparse and random nature of the outputs of the oracles, making the oracle $\mathbf{u}$ effectively useless for standard one-wayness. However, this oracle $\mathbf{u}$ is not much useful beyond this simple scenario. In particular, imaging a self-composing TDP construction, whose evaluation algorithm $\mathsf{E}^{\mathbf{e}}$ is the self-composition of

$\mathbf{e}(\mathrm{ik},\cdot)$; i.e., $\mathsf{E}^{\mathbf{e}}(\mathrm{ik},\mathrm{x}) = \mathbf{e}(\mathrm{ik},\mathbf{e}(\mathrm{ik},\mathrm{x}))$. An adversary $\mathcal{A}$ against enhanced one-wayness is given $(\mathrm{ik},\mathrm{r},\mathrm{y})$, and should find x such that $\mathrm{y} = \mathbf{e}(\mathrm{ik},\mathbf{e}(\mathrm{ik},\mathrm{x}))$. Given the randomness r, the adversary $\mathcal{A}$ can find x_0 such that $\mathbf{e}(\mathrm{ik},\mathrm{x}_0) = \mathrm{y}$ by calling $\mathbf{u}(\mathrm{ik},\mathrm{r})$, but $\mathcal{A}$ cannot continue to get to x, because $\mathcal{A}$ does not have the randomness of x_0.

Description of the oracle $\mathbf{u}$. The above discussion directs us toward a natural choice of $\mathbf{u}$: on input (ik,r), letting $\mathrm{y} := \mathbf{s}(\mathrm{ik},\mathrm{r})$, the oracle $\mathbf{u}(\mathrm{ik},\mathrm{r})$ returns the randomness of the pre-image of y, not the pre-image itself. That is, letting $\mathrm{x} \in \mathsf{Dom}_{\mathrm{ik}}$ be such that $\mathbf{e}(\mathrm{ik},\mathrm{x}) = \mathrm{y}$, the oracle $\mathbf{u}(\mathrm{ik},\mathrm{r})$ returns r_0, where $\mathbf{s}(\mathrm{ik},\mathrm{r}_0) = \mathrm{x}$.

Returning to the construction example above, it is not hard to see that this new oracle $\mathbf{u}$ not only breaks the enhanced one-wayness of the self-composition construction, but that of more general k-composition constructions, in which we compose $\mathbf{e}(\mathrm{ik},\cdot)$ k times. One would just need to sequentially call $\mathbf{u}$ k times to get down to the base pre-image.

The construction does not call $\mathbf{u}$ *itself.* We will assume that the construction $(\mathsf{G}^{\mathbf{O}},\mathsf{S}^{\mathbf{O}},\mathsf{E}^{\mathbf{O}},\mathsf{D}^{\mathbf{O}})$, which we want to show that can be broken by a polynomial number of queries to $(\mathbf{O},\mathbf{u})$, does not call $\mathbf{u}$ itself. This is sufficient for deriving a fully blackbox separation because the base oracle $\mathbf{O}$ *by itself* is a one-way TDP against all poly-query adversaries with access to $(\mathbf{O},\mathbf{u})$. Our separation model is close to those of [GMR01,HR04], which only rule out fully-blackbox constructions, as opposed to the earlier models of [IR89,Sim98,GKM$^+$00], which also rule out relativizing reductions.

Main techniques. We now give a high-level sketch of how to attack a general construction $(\mathsf{G}^{\mathbf{O}},\mathsf{S}^{\mathbf{O}},\mathsf{E}^{\mathbf{O}},\mathsf{D}^{\mathbf{O}})$. Let (IK,R) be the challenge input to the adversary: if $\mathrm{Y} := \mathsf{S}^{\mathbf{O}}(\mathrm{IK};\mathrm{R})$, the adversary should invert Y w.r.t. IK. The main difficult part in inverting Y is to reply to queries for which we need to invert some image y w.r.t. the oracle $\mathbf{e}(\mathrm{ik},\cdot)$. We denote such queries as $\mathbf{e}^{-1}(\mathrm{ik},\mathrm{y})$: namely, if $\mathbf{e}(\mathrm{ik},\mathrm{x}) = \mathrm{y}$, then $\mathbf{e}^{-1}(\mathrm{ik},\mathrm{y}) = \mathrm{x}$.

As in the above k-composition construction example, suppose (informally) one can start the decryption execution of Y without having the underlying inversion key; namely, it is just a matter of answering a few oracle queries of the form $\mathbf{e}^{-1}(\mathrm{ik},\mathrm{y})$ for various (ik,y). Roughly, for any meaningful query $\mathrm{qu} := \mathbf{e}^{-1}(\mathrm{ik},\mathrm{y})$ during this execution we will have two cases: (I) y was generated during the process which produced $(\mathrm{IK},*) \xleftarrow{\$} \mathsf{G}^{\mathbf{O}}(1^{\kappa})$: namely, during this process there was a query/response $((\mathrm{ik},\mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ or $((\mathrm{ik},\mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$ for some x and r, and (II) y was generated during the execution of $\mathrm{Y} := \mathsf{S}^{\mathbf{O}}(\mathrm{IK};\mathrm{R})$.

We will show that cases (I) and (II) are the only likely cases; this is roughly because otherwise one can forge such a valid (ik,y) without making a corresponding query: This is very unlikely because of the sparseness of the oracle outputs.

Let Q_s be the set of all queries/responses during $\mathsf{S}^{\mathbf{O}}(\mathrm{IK};\mathrm{R})$. If during the inversion of Y Case (II) holds, then either $((\mathrm{ik},\mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ in Q_s, in which case the answer to the query qu is clear, or $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$ is in Q_s, which can be used along with the oracle $\mathbf{u}$ to reply to the query qu.

The main difficult part of our analysis involves handling Case (I): in this case the adversary does not have enough information to reply to qu correctly. At a high-level, our solution is as follows. We will distinguish between two types of such qu queries: *important* and *immaterial*. We say qu is important if a query/response $((\mathrm{ik}, *) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$ or $((\mathrm{ik}, *) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ happens with 'good' probability during a random execution of $\mathrm{X}' \xleftarrow{\$} \mathsf{S}^{\mathbf{O}}(\mathrm{IK})$ followed by $\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X}')$. If qu is important, then $\mathbf{e}^{-1}(\mathrm{ik}, \mathrm{y})$ is likely to be determined by performing these two preceding executions many times. If qu is immaterial (namely, it will not be picked up during these many sample executions), then we will show that during the inversion of Y one may reply to qu with a random answer without making the result of the overall inversion of Y significantly skewed. The intuition is: in this case neither of $((\mathrm{ik}, *) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$ and $((\mathrm{ik}, *) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ are likely to happen during the sampling algorithm that produced the challenge pre-image X and during $\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X})$ which results in Y. We will use this intuition to build hybrid oracles, denoted $\mathbf{O}\Diamond\widetilde{\mathbf{O}}$, which provide random answers to such immaterial queries but relative to which all of IK, X and Y are valid.

In Section 4 we will give a more concrete overview of our techniques and approach by showing how to break the enhanced one-wayness of any construction whose oracle access is of the form $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$. We will then give the general attack against all constructions in Section 5.

2 Preliminaries

If $\mathcal{D}$ is a distribution, we use $\mathrm{x} \xleftarrow{\$} \mathcal{D}$ to indicate x is sampled according to $\mathcal{D}$ and we use $\mathrm{x}' \in \mathcal{D}$ to indicate $\mathrm{x}' \in \mathsf{support}(\mathcal{D})$. If $\mathsf{R}(\mathrm{x}_1, \ldots, \mathrm{x}_n)$ is a randomized algorithm, then $\mathsf{R}(\mathrm{a}_1, \ldots, \mathrm{a}_n)$ denotes the random variable $\mathsf{R}(\mathrm{a}_1, \ldots, \mathrm{a}_n; \mathrm{r})$, where $\mathrm{r} \xleftarrow{\$} \{0,1\}^*$.

If f is a function and Dom is a set, then $f(\mathsf{Dom}) \triangleq \{f(x) \mid x \in \mathsf{Dom}\}$.

We start with the definition of a family of trapdoor permutations. Each function $\mathsf{E}(\mathrm{IK}, \cdot)$ in the family acts as a permutation over a domain $\mathsf{Dom}_{\mathrm{IK}} \subseteq \{0,1\}^w$ (for some fixed polynomial w specified by the permutation family), where the domain $\mathsf{Dom}_{\mathrm{IK}}$ may possibly depend on IK. Moreover, this induced permutation can be inverted using any matching trapdoor key for IK. Finally, there is a sampling algorithm S, where $\mathsf{S}(\mathrm{IK})$ allows one to sample from $\mathsf{Dom}_{\mathrm{IK}}$.

Definition 1 (Trapdoor Permutations). *Let $w = w(\kappa)$ be an arbitrary polynomial. A family of trapdoor permutations* TDP *consists of four PPT algorithms* G, S, E *and* D *defined as follows.*

- $\mathsf{G}(1^\kappa)$*: The key generation algorithm* G *takes as input a security parameter* 1^κ *and outputs a pair* $(\mathrm{IK}, \mathrm{TK})$ *of index/trapdoor keys.*

- $\mathsf{S}(\mathrm{IK};\mathrm{R})$*: The sampling algorithm* S *takes as input an index key* IK *and randomness* $\mathrm{R} \in \{0,1\}^\kappa$ *and outputs an element* $\mathrm{X} \in \{0,1\}^w$*. We use* $\mathsf{Dom}_{\mathrm{IK}}$ *to denote the set of values* X *which are outputted by* $\mathsf{S}(\mathrm{IK};\cdot)$*.*
- $\mathsf{E}(\mathrm{IK},\mathrm{X})$*: The evaluation algorithm* E *takes as input an index key* IK *and an element* $\mathrm{X} \in \{0,1\}^w$ *and outputs* $\mathrm{Y} \in \{0,1\}^w \cup \{\perp\}$*.*
- $\mathsf{D}(\mathrm{TK},\mathrm{Y})$*: The inversion algorithm* D *takes as input a trapdoor key* TK*, and an element* $\mathrm{Y} \in \{0,1\}^w$ *and outputs* $\mathrm{X} \in \{0,1\}^w \cup \{\perp\}$*.*

We will now define the notion of correctness, as well as two one-wayness notions. As terminology, we say that an index key IK *is valid if* $(\mathrm{IK},*) = \mathsf{G}(1^\kappa;\mathrm{R})$ *for some randomness* R*.*

- ***Correctness.*** *For any valid index key* IK*, the function* $\mathsf{E}(\mathrm{IK},\cdot)$ *induces a permutation over* $\mathsf{Dom}_{\mathrm{IK}}$*. Moreover, for any security parameter* κ *we have* $\Pr[\mathsf{D}(\mathrm{TK},\mathsf{E}(\mathrm{IK},\mathrm{X})) = \mathrm{X}] = 1$*, where* $(\mathrm{IK},\mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}(1^\kappa)$*,* $\mathrm{R} \overset{\$}{\leftarrow} \{0,1\}^\kappa$ *and* $\mathrm{X} := \mathsf{S}(\mathrm{IK};\mathrm{R})$*.*
- ***Standard one-wayness.*** *For any PPT adversary we have* $\mathcal{A} \Pr[\mathcal{A}(\mathrm{IK},\mathrm{Y}) = \mathsf{D}(\mathrm{TK},\mathrm{Y})] = \mathrm{negl}(\kappa)$*, where* $(\mathrm{IK},\mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}(1^\kappa)$*,* $\mathrm{R} \overset{\$}{\leftarrow} \{0,1\}^\kappa$ *and* $\mathrm{Y} := \mathsf{S}(\mathrm{IK};\mathrm{R})$*.*
- ***Enhanced one-wayness.*** *For any PPT adversary* $\mathcal{A}$

$$\Pr[\mathcal{A}(\mathrm{IK},\mathrm{Y},\mathrm{R}) = \mathsf{D}(\mathrm{TK},\mathrm{Y})] = \mathrm{negl}(\kappa),$$

where $(\mathrm{IK},\mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}(1^\kappa)$*,* $\mathrm{R} \overset{\$}{\leftarrow} \{0,1\}^\kappa$*,* $\mathrm{Y} := \mathsf{S}(\mathrm{IK};\mathrm{R})$*. Note that* Y *can be computed from* IK *and* R*, but we include it separately just for notational convenience.*

We now define the notion of fully-blackbox constructions, tailored to our setting. See [RTV04,BBF13] for more general notions.

Definition 2 (Fully blackbox constructions). *A fully-blackbox (shortly, a blackbox) construction of an enhanced TDP from a standard TDP consists of a PPT oracle-aided construction* $(\mathsf{G},\mathsf{S},\mathsf{E},\mathsf{D})$ *and a PPT oracle-aided reduction algorithm* Red *satisfying the following. For any correct TDP oracle* $\mathbf{O} = (\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d})$ *(where correctness is defined in Definition 1) we have*

1. ***Correctness:*** $(\mathsf{G}^{\mathbf{O}},\mathsf{S}^{\mathbf{O}},\mathsf{E}^{\mathbf{O}},\mathsf{D}^{\mathbf{O}})$ *is a correct TDP;*
2. ***Security:*** *for any adversary* $\mathcal{A}$ *breaking the enhanced one-wayness of the oracle-aided scheme* $(\mathsf{G}^{\mathbf{O}},\mathsf{S}^{\mathbf{O}},\mathsf{E}^{\mathbf{O}},\mathsf{D}^{\mathbf{O}})$*, the oracle algorithm* $\mathsf{Red}^{\mathbf{O},\mathcal{A}}$ *breaks the standard one-wayness of* $\mathbf{O}$*.*

3 Main Theorem and Proofs Roadmap

In this section we describe our main theorem and the roadmap of the proofs.

As common in impossibility results, we prove our main theorem by showing the existence of an oracle relative to which the base primitive exists (namely, standard TDPs), but not the target primitive (namely, enhanced TDPs). Technically, our separation model is closest to that of [HR04], which only results in fully-blackbox separations, as opposed to the more general *relativizing* separations, considered in most previous work, e.g., [IR89,Sim98,GKM$^+$00].

Theorem 1 (Impossibility of Enhanced TDPs from Standard TDPs). *There exists oracles $(\mathbf{O}, \mathbf{u}, \mathbf{v})$, where $\mathbf{O} := (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$, such that both the following conditions hold.*

1. *$\mathbf{O}$ is a standard TDP against every polynomial-query adversary $\mathcal{A}^{\mathbf{O},\mathbf{u},\mathbf{v}}$: That is, the probability that $\mathcal{A}^{\mathbf{O},\mathbf{u},\mathbf{v}}(\mathrm{ik}, \mathrm{y}) = \mathrm{x}$ is at most negligible, where $(\mathrm{ik}, \mathrm{tk}) \xleftarrow{\$} \mathbf{g}(1^\kappa)$, $\mathrm{x} \xleftarrow{\$} \mathbf{s}(\mathrm{ik})$ and $\mathrm{y} := \mathbf{e}(\mathrm{ik}, \mathrm{x})$.*
2. *The enhanced one-wayness of any construction $(\mathsf{G}^{\mathbf{O}}, \mathsf{S}^{\mathbf{O}}, \mathsf{E}^{\mathbf{O}}, \mathsf{D}^{\mathbf{O}})$ can be broken by a poly-query adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$. That is, the probability that $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(\mathrm{IK}, \mathrm{R}, \mathrm{Y}) = \mathsf{D}^{\mathbf{O}}(\mathrm{TK}, \mathrm{Y})$ is non-negligible, where $(\mathrm{IK}, \mathrm{TK}) \xleftarrow{\$} \mathsf{G}^{\mathbf{O}}(1^\kappa)$, $\mathrm{R} \xleftarrow{\$} \{0,1\}^*$ and $\mathrm{Y} := \mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R})$.*

As a result, there exists no fully-blackbox construction of enhanced TDPs from standard TDPs.

Roadmap: Proof of Theorem 1. The “as a result” part follows immediately from Parts 1 and 2 of the theorem, and thus we focus on proving these two parts. (For completeness, we show how to derive the “as a result” part below.) As common in impossibility results, we show the existence of the oracles $(\mathbf{O}, \mathbf{u}, \mathbf{v})$, required by Theorem 1, by first describing a distribution of oracles, and then proving results for oracles randomly chosen from this distribution. We will first start by describing a distribution Ψ of oracles $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$. A randomly chosen $\mathbf{O} = (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$ from this distribution will allow one to implement an ideal version of a TDP, which not only satisfies standard one-wayness, but also enhanced-one-wayness. We then introduce two weakening oracles $\mathbf{u}$ and $\mathbf{v}$, so that the oracle $\mathbf{O}$ still provides standard one-wayness in the presence of $\mathbf{u}$ and $\mathbf{v}$, but the enhanced one-wayness of any TDP construction instantiated with $\mathbf{O}$ can be broken by making a polynomial number of queries to $(\mathbf{O}, \mathbf{u}, \mathbf{v})$.

In the following definition, whenever we say a function $f\colon \mathsf{Dom} \to \mathsf{Ran}$ with property P (e.g., injectivity) is a randomly chosen function we mean f is chosen uniformly at random from the space of all functions from Dom to Ran having property P.

Definition 3. *We define an oracle distribution Ψ that produces an ensemble of oracles $(\mathbf{O}_\kappa, \mathbf{u}_\kappa, \mathbf{v}_\kappa)_\kappa$. For all κ and all $\mathrm{ik} \in \{0,1\}^\kappa$, choose a set D_{ik} uniformly at random under the conditions that $\mathsf{D}_{\mathrm{ik}} \subseteq \{0,1\}^{5\kappa}$ and that $|\mathsf{D}_{\mathrm{ik}}| = 2^\kappa$.*

- *$\mathbf{g}_\kappa\colon \{0,1\}^\kappa \to \{0,1\}^\kappa$ is a random injective function, mapping a trapdoor key to an index key.*
- *$\mathbf{s}_\kappa\colon \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^{5\kappa}$ is a random function, where for all $\mathrm{ik} \in \{0,1\}^\kappa$: $\mathbf{s}_\kappa(\mathrm{ik}, \cdot)$ is 1-1 and for all $\mathrm{r} \in \{0,1\}^\kappa$: $\mathbf{s}_\kappa(\mathrm{ik}, r) \in \mathsf{D}_{\mathrm{ik}}$*

- $\mathbf{e}_\kappa\colon \{0,1\}^\kappa \times \{0,1\}^{5\kappa} \to \{0,1\}^{5\kappa} \cup \{\bot\}$ *is a random function, satisfying the following two conditions: for all* $\mathrm{ik} \in \{0,1\}^\kappa$*:* $\mathbf{e}_\kappa(\mathrm{ik}, \mathsf{D}_{\mathrm{ik}}) = \mathsf{D}_{\mathrm{ik}}$ *and for all* $\mathrm{x} \notin \mathsf{D}_{\mathrm{ik}}$*:* $\mathbf{e}_\kappa(\mathrm{ik}, \mathrm{x}) = \bot$*.*
- $\mathbf{d}_\kappa\colon \{0,1\}^\kappa \times \{0,1\}^{5\kappa} \to \{0,1\}^{5\kappa} \cup \{\bot\}$ *is a function, where* $\mathbf{d}_\kappa(\mathrm{tk}, \mathrm{y})$ *is defined as follows. Letting* $\mathrm{ik} := \mathbf{g}_\kappa(\mathrm{tk})$*, if* $\mathrm{y} \in \mathsf{D}_{\mathrm{ik}}$*, then letting* x *be the unique string satisfying* $\mathbf{e}_\kappa(\mathrm{ik}, \mathrm{x}) = \mathrm{y}$*, set* $\mathbf{d}_\kappa(\mathrm{tk}, \mathrm{y}) := \mathrm{x}$*. Otherwise (i.e., if* $\mathrm{y} \notin \mathsf{D}_{\mathrm{ik}}$*), set* $\mathbf{d}_\kappa(\mathrm{tk}, \mathrm{y}) := \bot$*.*
- $\mathbf{u}_\kappa\colon \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ *is defined as follows. For* $\mathrm{ik} \in \{0,1\}^\kappa$ *and* $\mathrm{r} \in \{0,1\}^\kappa$*, letting* $\mathrm{y} := \mathbf{s}_\kappa(\mathrm{ik}, \mathrm{r})$ *and* r_0 *be such that* $\mathrm{y} = \mathbf{e}_\kappa(\mathrm{ik}, \mathbf{s}_\kappa(\mathrm{ik}, \mathrm{r}_0))$*, set* $\mathbf{u}_\kappa(\mathrm{ik}, \mathrm{r}) := \mathrm{r}_0$*.*
- $\mathbf{v}_\kappa\colon \{0,1\}^\kappa \times \{0,1\}^{5\kappa} \to \{\bot, \top\}$ *is defined as follows:* $\mathbf{v}_\kappa(\mathrm{ik}, \mathrm{x})$ *checks whether the given input* x *is in* D_{ik} *or not: set* $\mathbf{v}_\kappa(\mathrm{ik}, \mathrm{x}) := \top$ *if* $\mathrm{x} \in \mathsf{D}_{\mathrm{ik}}$*, and* $\mathbf{v}_\kappa(\mathrm{ik}, \mathrm{x}) := \bot$*, otherwise.*

Redundancy of the oracle $\mathbf{v}_\kappa$*.* Note that the oracle $\mathbf{v}_\kappa$ can be simulated by $\mathbf{e}_\kappa$. We only include this oracle as it will simplicity notation.

Convention and notation. We will often drop the security parameter κ as a sub-index to the oracles whenever the underlying security parameter is clear from the context. For an oracle algorithm $A^{\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d}}$ we use notation such as $(\mathrm{qu} \underset{\mathbf{g}}{\to} \mathrm{an})$ to indicate that $\mathcal{A}$ queries $\mathbf{g}$ on qu and receives an as the answer. We also use $(\mathrm{qu} \underset{\mathbf{g}}{\to} ?)$ to indicate that the query qu is asked.

We will now give a simple-information theoretic lemma showing that a randomly chosen TDP $\mathbf{O}$ is standard one-way even in the presence of the oracle $\mathbf{u}$. The proof of the following theorem is based on simple information theoretic arguments and so is omitted.

Lemma 1 (O is one-way relative to $(\mathbf{O}, \mathbf{u}, \mathbf{v})$). *For any polynomial query adversary* $\mathcal{A}$ *we have*

$$\Pr[\mathcal{A}^{\mathbf{O},\mathbf{u},\mathbf{v}}(\mathrm{ik}, \mathrm{y}) = \mathrm{x} \text{ and } \mathbf{e}(\mathrm{ik}, \mathrm{x}) = \mathrm{y}] \leq \tfrac{1}{2^{\kappa/3}},$$

where $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v}) \leftarrow \Psi$*,* $\mathbf{O} := (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$*,* $\mathrm{tk} \overset{\$}{\leftarrow} \{0,1\}^\kappa$ *and* $\mathrm{ik} = \mathbf{g}(\mathrm{tk})$*. This bound holds so long as* $\mathcal{A}$ *is poly-query bounded (and unbounded otherwise).*

The following lemma shows how to break the enhanced one-wayness of any candidate construction.

Lemma 2 (Breaking enhanced one-wayness of any construction). *Let* $(\mathsf{G}, \mathsf{S}, \mathsf{E}, \mathsf{D})$ *be a candidate blackbox construction of a TDP. There exists a polynomial query adversary* Break *such that*

$$\Pr[\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R}, \mathrm{Y}) = \mathrm{X}] \geq 1 - \frac{1}{\kappa^2},$$

where $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v}) \overset{\$}{\leftarrow} \Psi$*,* $\mathbf{O} := (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$*,* $(\mathrm{IK}, \mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}^{\mathbf{O}}(1^\kappa)$*,* $\mathrm{R} \overset{\$}{\leftarrow} \{0,1\}^*$*,* $\mathrm{Y} := \mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R})$ *and* $\mathrm{X} := \mathsf{D}^{\mathbf{O}}(\mathrm{TK}, \mathrm{Y})$*.*

Completing the Proof of Theorem 1. The proof of Theorem 1 follows easily by combining Lemmas 1 and 2, as given below.

Proof (of Theorem 1). We will first prove the "as a result" part of the theorem. Suppose to the contrary that there exists an enhanced TDP construction $(\mathsf{G}, \mathsf{S}, \mathsf{E}, \mathsf{D})$, and let Red be the PPT security reduction algorithm guaranteed to exist by Definition 2. Let $(\mathbf{O}, \mathbf{u}, \mathbf{v})$ be the oracle shown to exist by Parts 1 and 2 of the theorem. By Part 2 of the theorem we know that there exists a polynomial query adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$ which breaks the enhanced one-wayness of $(\mathsf{G}^{\mathbf{O}}, \mathsf{S}^{\mathbf{O}}, \mathsf{E}^{\mathbf{O}}, \mathsf{D}^{\mathbf{O}})$. Thus, by definition of blackbox constructions, $\mathsf{Red}^{\mathsf{Break},\mathbf{O}}$ should break the standard one-wayness of $\mathbf{O}$. This however is a contradiction to Part 1, because $\mathsf{Red}^{\mathsf{Break},\mathbf{O}}$ can be simulated by a polynomial query adversary $\mathcal{A}^{\mathbf{O},\mathbf{u},\mathbf{v}}$.

We now prove Parts 1 and 2. To show the existence of the oracles $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ required by the theorem, we show

1. For a measure-one of oracles $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$, the oracle $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$ is standard oneway against all polynomial-query adversaries with oracle access to $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$.
2. For a measure-one of oracles $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$, the adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$ breaks the enhanced one-wayness of $(\mathsf{G}^{\mathbf{O}}, \mathsf{S}^{\mathbf{O}}, \mathsf{E}^{\mathbf{O}}, \mathsf{D}^{\mathbf{O}})$.

The above two statements implies the existence of a specific oracle $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$, meeting the requirement of the theorem.

We show how to derive Condition 2 from Lemma 2. The proof of Condition 1 follows similarly from Lemma 1.

By Lemma 2 we have

$$\Pr_{(\mathbf{O},\mathbf{u},\mathbf{v}),\mathrm{IK},\mathrm{R}}[\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R}) = \mathrm{X}] \geq 1 - \frac{1}{\kappa^2}. \tag{1}$$

Using a simple averaging argument we may obtain

$$\Pr_{(\mathbf{O},\mathbf{u},\mathbf{v})}\left[\Pr_{\mathrm{IK},\mathrm{R}}[\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R}) = \mathrm{X}] \geq \frac{1}{\kappa^3}\right] \geq 1 - \frac{1}{\kappa^{1.5}}. \tag{2}$$

Thus, for at most a $\frac{1}{\kappa^{1.5}}$ fraction of all oracles $(\mathbf{O}, \mathbf{u}, \mathbf{v})$, the adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$, on security parameter 1^κ, recovers the pre-image corectly with probability less than $\frac{1}{\kappa^3}$. Since $\sum \frac{1}{\kappa^{1.5}}$ converges, by the Borel-Cantelli Lemma we have that for a measure-one of oracles $(\mathbf{O}, \mathbf{u}, \mathbf{v})$, the adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$ breaks the enhanced-onewayness of $(\mathsf{G}^{\mathbf{O}}, \mathsf{S}^{\mathbf{O}}, \mathsf{E}^{\mathbf{O}}, \mathsf{D}^{\mathbf{O}})$: for all sufficiently large κ, the adversary recovers X from $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R}, \mathrm{Y})$ with probability at least $\frac{1}{\kappa^3}$. □

Roadmap for the Proof of Lemma 2. We are left with proving Lemma 2, which constitutes the main technical bulk of our work. As a warp up, first in Section 4 we will prove and give an overview of our techniques for a special case of Lemma 2: that in which the oracle access of the construction is of the form $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$. Then, we will give the proof for the general case in Section 5.

4 Proof of Lemma 2: Special Case $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$

In this section we show how to break the enhanced one-wayness of a simple class of TDP constructions, those in which the oracle access is of the form $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$. We call such constructions *type-1*. We first start with a general overview.

Setup. The input to the adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$ is $(\mathrm{IK}, \mathrm{R}, \mathrm{Y})$, where $(\mathrm{IK}, \mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}^{\mathbf{g}}(1^{\kappa})$, $\mathrm{R} \overset{\$}{\leftarrow} \{0,1\}^*$ and $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK}; \mathrm{R})$. The goal of Break is to find X such that $\mathrm{X} := \mathsf{D}^{\mathbf{d}}(\mathrm{TK}, \mathrm{Y})$.

High-level idea of Break***'s strategy.*** Consider a partial fake oracle $\mathbf{g}'$ and randomness R' under which we have $\mathsf{G}^{\mathbf{g}'}(\mathrm{R}') = (\mathrm{IK}, \widetilde{\mathrm{TK}})$ for some $\widetilde{\mathrm{TK}}$. By a partial oracle we mean an oracle that is defined only on a small set of all queries, those that occur exactly during the execution of $\mathsf{G}^{\mathbf{g}'}(\mathrm{R}')$. Such a fake oracle $\mathbf{g}'$ and corresponding matching randomness R' can be found by doing expensive offline computation and without interacting at all with the real oracles $(\mathbf{O}, \mathbf{u}, \mathbf{v})$.

Now consider the effect of super-imposing $\mathbf{g}'$ on the real oracle $\mathbf{g}$ to get an oracle $\widetilde{\mathbf{g}}$. This oracle $\widetilde{\mathbf{g}}$ is defined according to $\mathbf{g}'$ on all queries defined in $\mathbf{g}'$, and otherwise is defined as in $\mathbf{g}$.

For this perturbed oracle $\widetilde{\mathbf{g}}$, we will define a correspondingly perturbed oracle $\widetilde{\mathbf{d}}$ so that $(\widetilde{\mathbf{g}}, \mathbf{s}, \mathbf{e}, \widetilde{\mathbf{d}})$ is a valid TDP. Now since we know $\mathsf{G}^{\widetilde{\mathbf{g}}}(\mathrm{R}') = (\mathrm{IK}, \widetilde{\mathrm{TK}})$, we must have $\mathrm{X} = \mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$, and thus recovering the challenge pre-image X amounts to one's ability to perform the execution of $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ by only making a polynomial number queries to $(\mathbf{O}, \mathbf{u}, \mathbf{v})$. As we will see, the naive way of performing this execution will result in an exponential number of queries to $(\mathbf{O}, \mathbf{u}, \mathbf{v})$. Our main technique will allow us to get around this problem by making use of the oracle $\mathbf{u}$ and knowledge of R (which is the randomness underlying the image point Y).

Organization of Section 4. In Section 4.1 we will give a more detailed (but still informal) overview of the above approach for the case in which each of the algorithms $(\mathsf{G}, \mathsf{S}, \mathsf{E}, \mathsf{D})$ makes only one query. We will then formally describe an attack against any candidate many-query construction $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$ in the next two subsections.

4.1 General Overview: One Query Case

We will now give a concrete overview of the above abstract approach for the following type of construction: We assume each of the algorithms $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$ makes only one query. The input to the adversary $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$ is $(\mathrm{IK}, \mathrm{R}, \mathrm{Y})$, where $(\mathrm{IK}, \mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}^{\mathbf{g}}(1^{\kappa})$, $\mathrm{R} \overset{\$}{\leftarrow} \{0,1\}^*$ and $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK}; \mathrm{R})$. Let X denote Break's challenge image point; namely, we have $\mathsf{E}^{\mathbf{e}}(\mathrm{IK}, \mathrm{X}) = \mathrm{Y}$.

We sketch the main steps taken by Break, and will explain about each of them.

Sampling a fake oracle and a trapdoor key. Sample an oracle $\mathbf{g}'$ and a randomness value R' uniformly at random in such a way that

$$\mathsf{G}^{\mathbf{g}'}(1^\kappa;\mathrm{R}') = (\mathrm{IK}, \widetilde{\mathrm{TK}}), \tag{3}$$

for some $\widetilde{\mathrm{TK}}$. Since G makes only one query, we may think of $\mathbf{g}'$ as only one query/response pair $\mathsf{qa} := (\mathrm{tk} \underset{\mathbf{g}}{\rightarrow} \mathrm{ik})$. Thus, we may write Equation 3 as $\mathsf{G}^{\mathsf{qa}}(1^\kappa;\mathrm{R}') = (\mathrm{IK}, \widetilde{\mathrm{TK}})$.

Defining the oracle $\widetilde{\mathbf{g}}$. Consider an oracle $\widetilde{\mathbf{g}} := \mathsf{qa}\Diamond^*\mathbf{g}$, where the *composed* oracle $\mathsf{qa}\Diamond^*\mathbf{g}$ is defined as follows: $(\mathsf{qa}\Diamond^*\mathbf{g})(\mathrm{tk}') = \mathrm{ik}$ if $\mathrm{tk}' = \mathrm{tk}$; otherwise, $(\mathsf{qa}\Diamond^*\mathbf{g})(\mathrm{tk}') = \mathbf{g}(\mathrm{tk}')$. Briefly, the oracle $\mathsf{qa}\Diamond^*\mathbf{g}$ first forwards a given query to qa, and if the query is not defined there, the query will be forwarded to $\mathbf{g}$.

Defining the oracle $\widetilde{\mathbf{d}}$. We now define $\widetilde{\mathbf{d}}$ in such a way that $(\widetilde{\mathbf{g}}, \mathbf{s}, \mathbf{e}, \widetilde{\mathbf{d}})$ forms a valid TDP oracle. For any tk' and y', the value of $\widetilde{\mathbf{d}}(\mathrm{tk}', \mathrm{y}')$ is formed as follows. Letting $\mathrm{ik}' = \widetilde{\mathbf{g}}(\mathrm{tk}')$:

- If $\mathbf{v}(\mathrm{ik}', \mathrm{y}') = \perp$, then set $\widetilde{\mathbf{d}}(\mathrm{tk}', \mathrm{y}') = \perp$;
- Otherwise, letting x' be the unique string for which we have $\mathbf{e}(\mathrm{ik}', \mathrm{x}') = \mathrm{y}'$, set $\widetilde{\mathbf{d}}(\mathrm{tk}', \mathrm{y}') = \mathrm{x}'$. Note that since we know $\mathbf{v}(\mathrm{ik}', \mathrm{y}') = \top$ (because otherwise the previous check would hold), by definition of $\mathbf{e}$ (Definition 3) such x' does exist and it is unique.

Performing the execution $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ is enough. It is straightforward to verify that $(\widetilde{\mathbf{g}}, \mathbf{s}, \mathbf{e}, \widetilde{\mathbf{d}})$ forms a valid TDP oracle. Moreover, by definition of $\widetilde{\mathbf{g}}$ and R', we have $\mathsf{G}^{\widetilde{\mathbf{g}}}(\mathrm{R}') = (\mathrm{IK}, \widetilde{\mathrm{TK}})$. Now since $\mathsf{E}^{\mathbf{e}}(\mathrm{IK}, \mathrm{X}) = \mathrm{Y}$, by completeness of the construction, we will have $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y}) = \mathrm{X}$, where X is Break's challenge image point.

Executing $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ *efficiently?* Can we execute $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ by making only a polynomial number of queries to $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u})$? Let us look at all the possibilities for a possible encountered query $((\mathrm{tk}', \mathrm{y}') \underset{\widetilde{\mathbf{d}}}{\rightarrow} ?)$ below. Let $\mathrm{ik}' := \widetilde{\mathbf{g}}(\mathrm{tk}')$, which can be computed by making at most one query to $\mathbf{g}$.

1. **Simple case**: $\mathrm{ik}' \neq \mathrm{ik}$ (recall that ik is defined in the query/response set qa, which in turn forms $\widetilde{\mathbf{g}}$): in this case by inspection we can see that we indeed have $\mathbf{d}(\mathrm{tk}', \mathrm{y}') = \widetilde{\mathbf{d}}(\mathrm{tk}', \mathrm{y}')$, and so the answer can be determined by calling $\mathbf{d}$ directly.
2. **Simple case**: $\mathrm{ik}' = \mathrm{ik}$ and $\mathbf{v}(\mathrm{ik}, \mathrm{y}') = \perp$: in this case we can again easily see that $\mathbf{d}(\mathrm{tk}', \mathrm{y}') = \perp$.
3. **Problematic case**: $\mathrm{ik}' = \mathrm{ik}$ and $\mathbf{v}(\mathrm{ik}, \mathrm{y}') \neq \perp$: in this case Break cannot right away compute the value of $\widetilde{\mathbf{d}}(\mathrm{tk}', \mathrm{y}')$ because in order to do so, Break must find an x' such that $\mathbf{e}(\mathrm{ik}, \mathrm{x}') = \mathrm{y}'$.

The oracle* $\mathbf{u}$ *and randomness* R *to the rescue. From the above discussion, the attacker Break only needs to handle Case 3. That is, from the pair $(\mathrm{ik},\mathrm{y}')$, upon which Line 3 is hit, and without knowledge of ik's trapdoor key $\mathbf{g}^{-1}(\mathrm{ik})$, the attacker Break should find an x' such that $\mathbf{e}(\mathrm{ik},\mathrm{x}') = \mathrm{y}'$. Recall that D makes only one query, and so if Break gets past this "one-time" problematic case, it will be done.

Recall that the input to Break is $(\mathrm{IK},\mathrm{R},\mathrm{Y})$, where R is the randomness underlying the image point Y. We claim that with all but negligible probability the following must hold: letting $(\mathrm{ik},\mathrm{y}')$ be the pair upon which Line 3 was hit, during the execution of $\mathsf{S}^{\mathbf{s}}(\mathrm{IK};\mathrm{R})$ we must have a query/response pair $((\mathrm{ik},\mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}')$ for some r. Assuming that this claim holds, Break may then simply call $((\mathrm{ik},\mathrm{r}) \underset{\mathbf{u}}{\rightarrow} ?)$ to get r', and then call $((\mathrm{ik},\mathrm{r}') \underset{\mathbf{s}}{\rightarrow} ?)$ to get x', completing its attack.

It remains to prove the above claim. We show that if the claim does not hold, then one may efficiently produce a pair $(\mathrm{ik}',\mathrm{y}')$, where y' is a valid image of $\mathbf{s}(\mathrm{ik}',*)$, *without* ever calling $\mathbf{s}(\mathrm{ik}',\cdot)$ on the corresponding pre-image of y', and without ever calling $\mathbf{e}$ and $\mathbf{d}$ at all. Due to the sparse and random nature of the oracle $\mathbf{s}$, the probability of this event is at most negligible. To produce $(\mathrm{ik}',\mathrm{y}')$, do the following.

- Sample $(\mathrm{IK},\mathrm{TK}) \xleftarrow{\$} \mathsf{G}^{\mathbf{g}}(1^{\kappa})$, $\mathrm{R} \xleftarrow{\$} \{0,1\}^*$ and set $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK};\mathrm{R})$.
- Form $\widetilde{\mathrm{TK}}$ and $\widetilde{\mathbf{d}}$ as above. (This step is done offline, without interacting with the real oracles.)
- Run $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}},\mathrm{Y})$ and as soon as as query $((\mathrm{tk}',\mathrm{y}') \underset{\widetilde{\mathbf{d}}}{\rightarrow} ?)$ is made, return $(\mathrm{ik}',\mathrm{y}')$, where $\mathrm{ik}' := \mathbf{g}(\mathrm{tk}')$.

Our claim about the pair $(\mathrm{ik}',\mathrm{y})$ now follows.

4.2 Definitions and Simple Lemmas

In this section we will give some definitions and simple lemmas, which will then be used in Section 4.3. Some of these were informally reviewed in Section 4.1.

TDP-Valid and Ψ-Valid Oracles. Recall the distribution Ψ on oracles $(\mathbf{O},\mathbf{u},\mathbf{v})$ given in Definition 3. We say that an oracle $\mathbf{O}_1 := (\mathbf{g}_1,\mathbf{s}_1,\mathbf{e}_1,\mathbf{d}_1)$ is *Ψ-valid* if $\mathbf{O}_1$ is a possible output of Ψ. This means in particular that the input and output sizes of the sub-routines of $\mathbf{O}_1$ match those specified in Definition 3. We say that an oracle $\mathbf{O}_2 := (\mathbf{g}_2,\mathbf{s}_2,\mathbf{e}_2,\mathbf{d}_2)$ is *TDP valid* if $\mathbf{O}_2$ satisfies the completeness condition of Definition 1. Note that if an oracle is Ψ-valid then it is also TDP-valid, but the converse is not true.

Similarly, we say that a partial oracle $\mathbf{O}'$ (which is not defined on all points) is Ψ-valid (resp., TDP-valid) if there exists a full Ψ-valid (resp., a full TDP-valid) oracle $\mathbf{O}$ such that $\mathbf{O}' \subseteq \mathbf{O}$. Here, $\mathbf{O}' \subseteq \mathbf{O}$ means that $\mathbf{O}$ agree with $\mathbf{O}'$.

Definition 4 (Composed Oracles $\Diamond^*$). *Let* $\mathbf{O} := (\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d})$ *be a Ψ-valid oracle and let*

$$\mathbf{g}' := \{(\mathrm{tk}_1 \underset{\mathbf{g}}{\rightarrow} \mathrm{ik}_1),\ldots,(\mathrm{tk}_w \underset{\mathbf{g}}{\rightarrow} \mathrm{ik}_w)\}$$

be a partial Ψ-valid oracle consisting of only $\mathbf{g}$*-type queries. We define the composed oracle* $\mathbf{g}'\Diamond^*\mathbf{O} := (\widetilde{\mathbf{g}}, \mathbf{s}, \mathbf{e}, \widetilde{\mathbf{d}})$*, which has perturbed key-generation and inversion oracles, as follows.*

- $\widetilde{\mathbf{g}}(\cdot)$*: for a given* tk*, let* $\widetilde{\mathbf{g}}(\mathrm{tk}) \triangleq \mathrm{ik}_i$ *if* $\mathrm{tk} = \mathrm{tk}_i$ *for* $i \in [w]$*; otherwise,* $\widetilde{\mathbf{g}}(\mathrm{tk}) \triangleq \mathbf{g}(\mathrm{tk})$*.*
- $\widetilde{\mathbf{d}}(\cdot,\cdot)$*: for a given pair* $(\mathrm{tk}, \mathrm{y})$*, define* $\widetilde{\mathbf{d}}(\mathrm{tk}, \mathrm{y})$ *as follows. Assuming* $\mathrm{ik} = \widetilde{\mathbf{g}}(\mathrm{tk})$*, let* $\widetilde{\mathbf{d}}(\mathrm{tk}, \mathrm{y}) \triangleq \mathbf{e}^{-1}(\mathrm{ik}, \mathrm{y})$*. Here,* $\mathbf{e}^{-1}(\mathrm{ik}, \cdot)$ *is the inverse function of* $\mathbf{e}(\mathrm{ik}, \cdot)$ *— i.e.,* $\mathbf{e}^{-1}(\mathrm{ik}, \mathrm{y}) = \mathrm{x}$ *if for some* x*,* $\mathbf{e}(\mathrm{ik}, \mathrm{x}) = \mathrm{y}$*; otherwise,* $\mathbf{e}^{-1}(\mathrm{ik}, \mathrm{y}) = \perp$*. Note that by definition of* Ψ*, the function* $\mathbf{e}^{-1}(\mathrm{ik}, \cdot)$ *is indeed well-defined.*

It is straightforward to verify that the operation $\Diamond^*$ preserves completeness.

Lemma 3. *Let* $\mathbf{O}$ *and* $\mathbf{g}'$ *be as in Definition 4. Then, the composed oracle* $\mathbf{g}'\Diamond^*\mathbf{O}$ *is TDP-valid.*

Proof. The proof is straightforward and so is omitted. □

Consider a random Ψ-valid oracle $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$. Imagine an adversary that wants to come up with a pair $(\mathrm{ik}, \mathrm{y}) \in \{0,1\}^\kappa \times \{0,1\}^{5\kappa}$ of an index-key/image such that y lies in the support of $\mathbf{s}(\mathrm{ik})$. The following lemma shows that the probability that an adversary can do this in non-trivial way is exponentially small.

Lemma 4. *For any polynomial query oracle adversary* $\mathcal{B}$ *with access only to the oracles* $(\mathbf{g}, \mathbf{s}, \mathbf{u}, \mathbf{v})$ *we have*

$$\Pr\left[(\mathrm{ik}, \mathrm{y}) \xleftarrow{\$} \mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(1^\kappa) \text{ s.t. } \Big(((\mathrm{ik}, *) \underset{\mathbf{s}}{\to} \mathrm{y}) \notin \mathsf{Que}\Big) \wedge (\mathbf{v}(\mathrm{ik}, \mathrm{y}) = \top) \wedge (|\mathrm{ik}| = \kappa))\right] \leq \frac{1}{2^{3\kappa}}, \tag{4}$$

where $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v}) \xleftarrow{\$} \Psi$ *and* Que *is the set of all query/response pairs that* $\mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}$ *makes. We stress that* $\mathcal{B}$ *is not allowed to make* $\mathbf{e}$ *or* $\mathbf{d}$ *queries.*[2]

Proof. The proof is based on a simple information-theoretic argument and so we sketch the main idea. Assume w.l.o.g. that $\mathcal{B}$ before returning its guess $(\mathrm{ik}, \mathrm{y})$, it calls the oracle $\mathbf{v}$ on $(\mathrm{ik}, \mathrm{y})$. This only increases the number of queries by one.

At any point of execution, say the next query of $\mathcal{B}$ is a *hit* if the next query is a $\mathbf{v}$ query, say $((\mathrm{ik}', \mathrm{y}') \underset{\mathbf{v}}{\to} ?)$, which is a valid forgery; namely, (a) $((\mathrm{ik}', *) \underset{\mathbf{s}}{\to} \mathrm{y}') \notin \mathsf{Que}$, (b) $|\mathrm{ik}'| = \kappa$ and (c) $\mathbf{v}(\mathrm{ik}', \mathrm{y}') = \top$.

At any point, the probability that the next query is a hit given we had no hits before is at most $\dfrac{2^\kappa}{2^{5\kappa} - 2^\kappa}$. The proof now follows by a union bound.

[2] We may define and prove a version of this lemma which allows the adversary $\mathcal{B}$ to also make $\mathbf{e}$ and $\mathbf{d}$ queries. This current version however suffices for what we need for the simple separation we show in this section.

4.3 Many-Query Case

Fix the candidate type-1 construction $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$. We will build an adversary $\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}$ which breaks the enhanced one-wayness of $(\mathsf{G}^{\mathbf{g}}, \mathsf{S}^{\mathbf{s}}, \mathsf{E}^{\mathbf{e}}, \mathsf{D}^{\mathbf{d}})$ by making a polynomial number of queries to its oracles. The attacker Break does not need to call the oracles $\mathbf{e}$ and $\mathbf{d}$ during its attack, so we did not put them as superscripts to Break.

For simplicity we assume the following for all constructions $(\mathsf{G}, \mathsf{S}, \mathsf{E}, \mathsf{D})$ discussed in this paper. This assumption is made only for simplicity and all our results can be proved without it.

Assumption 2. Each of the algorithms $\mathsf{G}^{\mathbf{O}}$, $\mathsf{S}^{\mathbf{O}}$, $\mathsf{E}^{\mathbf{O}}$ and $\mathsf{D}^{\mathbf{O}}$ on a security parameter 1^κ call their oracle $\mathbf{O}$ always on the same security parameter 1^κ.

We will now describe the attacker Break. We will use notation and concepts from Definition 4.

Attacker $\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(\mathrm{IK}, \mathrm{R}, \mathrm{Y})$:

Oracles: $(\mathbf{g}, \mathbf{s}, \mathbf{u}, \mathbf{v})$, where $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v}) \xleftarrow{\$} \Psi$. Set $\mathbf{O} := (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$.

Input: $(\mathrm{IK}, \mathrm{R}, \mathrm{Y})$, where $(\mathrm{IK}, *) \xleftarrow{\$} \mathsf{G}^{\mathbf{g}}(1^\kappa)$, $\mathrm{R} \xleftarrow{\$} \{0,1\}^\kappa$ and $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK}; \mathrm{R})$.

Operations:

1. Sample (in an offline manner) a pair $(\mathbf{g}', \mathrm{R}')$ uniformly at random, where $\mathbf{g}'$ is a partial Ψ-valid oracle and $\mathrm{R}' \in \{0,1\}^\kappa$, under the condition that $(\mathrm{IK}, \widetilde{\mathrm{TK}}) = \mathsf{G}^{\mathbf{g}'}(1^\kappa; \mathrm{R}')$, for some $\widetilde{\mathrm{TK}}$. Let $\mathbf{g}' \Diamond^* \mathbf{O} := (\widetilde{\mathbf{g}}, \mathbf{s}, \mathbf{e}, \widetilde{\mathbf{d}})$ be formed as in Definition 4.
2. Let $\mathsf{L} := \emptyset$. Run $\mathsf{S}^{\mathbf{s}}(\mathrm{IK}; \mathrm{R})$ and for any query/response pair $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$ made, add $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$ to L.
3. Simulate the execution of $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ using the oracles $\mathbf{g}, \mathbf{s}, \mathbf{u}, \mathbf{v}$ as follows. For any encountered query $\mathsf{qu} := ((\mathrm{tk}, \mathrm{y}) \underset{\widetilde{\mathbf{d}}}{\rightarrow} ?)$, first compute $\widetilde{\mathbf{g}}(\mathrm{tk})$ to get ik; this can be done by making at most one query to $\mathbf{g}$. Then,
   (a) if $\mathbf{v}(\mathrm{ik}, \mathrm{y}) = \perp$, then reply to qu with $\perp$ and continue the execution;
   (b) else if $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}) \in \mathsf{L}$ for some r, then call $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{u}}{\rightarrow} ?)$ to receive r_0 and call $((\mathrm{ik}, \mathrm{r}_0) \underset{\mathbf{s}}{\rightarrow} ?)$ to get x. Return x as the response to the query qu, add $((\mathrm{ik}, \mathrm{r}_0) \underset{\mathbf{s}}{\rightarrow} \mathrm{x})$ to L and continue the execution.
   (c) else (i.e., if $\mathbf{v}(\mathrm{ik}, \mathrm{y}) = \top$ and $((\mathrm{ik}, *) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}) \notin \mathsf{L}$), then halt the execution and return Fail.
4. If the simulation has not halted yet, return $\widetilde{\mathrm{X}}$, the output of $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$.

Theorem 3. *The attacker* Break *is successful with probability at least* $1 - \frac{1}{2^{3\kappa}}$. *Namely,*

$$\Pr[\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(\mathrm{IK}, \mathrm{R}, \mathrm{Y}) = \mathrm{X}] \geq 1 - \frac{1}{2^{3\kappa}},$$

where the probability is taken over $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v}) \leftarrow \Psi$*,* $(\mathrm{IK}, \mathrm{TK}) \leftarrow \mathsf{G}^{\mathbf{g}}(1^\kappa)$*,* $\mathrm{R} \leftarrow \{0,1\}^\kappa$*,* $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK}, \mathrm{R})$ *and* $\mathrm{X} := \mathsf{D}^{\mathbf{d}}(\mathrm{TK}, \mathrm{Y})$.

Proof roadmap. We show that if the execution of Break never halts due to Line 3c, then the retrieved string $\widetilde{\mathrm{X}}$ is indeed the correct pre-image of Y. We will then show that the probability that Line 3c is ever hit (which we call the event Bad) is at most $\frac{1}{2^{3\kappa}}$, by "reudcing" it to Lemma 4. These two will complete the proof.

Lemma 5. *Let* Bad *be the event that line (3c) is hit during the execution of* $\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(\mathrm{IK},\mathrm{R},\mathrm{Y})$*. Then*

$$\Pr[\mathrm{Bad}] \leq \frac{1}{2^{3\kappa}},$$

where the probability is taken over $(\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{v}) \leftarrow \Psi$*,* $(\mathrm{IK},*) \leftarrow \mathsf{G}^{\mathbf{g}}(1^\kappa)$*,* $\mathrm{R} \leftarrow \{0,1\}^\kappa$*,* $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK},R)$ *and over* Break*'s random coins.*

We first show how to derive Theorem 3 from Lemma 5 and we will then prove Lemma 5.

Proof of Theorem 3. All probabilities that appear below are taken over the variables sampled in the theorem. We claim

$$\alpha \triangleq \Pr[\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(\mathrm{IK},\mathrm{R},\mathrm{Y}) = X \mid \overline{\mathrm{Bad}}] = 1.$$

Assuming the claim is true, we may combine it with Lemma 5 to get

$$\Pr[\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(\mathrm{IK},\mathrm{R},\mathrm{Y}) = X] \geq (1 - \frac{1}{2^{3\kappa}})\alpha = 1 - \frac{1}{2^{3\kappa}},$$

as desired. To prove the above claim first note that by Lemma 3 we have $\mathbf{g}'\lozenge^*\mathbf{O} := (\widetilde{\mathbf{g}},\mathbf{s},\mathbf{e},\widetilde{\mathbf{d}})$ is a valid TDP-oracle, where $\mathbf{g}'$ is formed in Step 1 of Break's execution. Moreover, recall that $\mathrm{Y} = \mathsf{E}^{\mathbf{e}}(\mathrm{IK},X)$ and that $(\mathrm{IK},\widetilde{\mathrm{TK}}) \in \mathsf{G}^{\widetilde{\mathbf{g}}}(1^\kappa)$. Thus, by the correctness condition of the blackbox construction $(\mathsf{G},\mathsf{S},\mathsf{E},\mathsf{D})$ (Definition 2) we have $X = \mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}},\mathrm{Y})$. The claim now follows by noting that if the event Bad does not hold, then the simulated execution of $\mathsf{D}^{\widetilde{\mathbf{d}}}(\widetilde{\mathrm{TK}},\mathrm{Y})$ performed by Break proceeds identically to the real decryption. The proof is now complete. □

Proof of Lemma 5. Let $\beta := \Pr[\mathrm{Bad}]$. We show how to construct an adversary $\mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}$ with oracle access to $(\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v})$ which makes a poly number of queries and with probability at least β forges some $(\mathrm{ik},\mathrm{y}) \in \{0,1\}^\kappa \times \{0,1\}^{5\kappa}$ in the sense of Lemma 4. Applying the lemma we will then obtain $\beta \leq \frac{1}{2^{3\kappa}}$, as desired.

The adversary $\mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(1^\kappa)$ first samples a random input $(\mathrm{IK},\mathrm{R},\mathrm{Y})$ for Break: namely, $(\mathrm{IK},\mathrm{TK}) \xleftarrow{\$} \mathsf{G}^{\mathbf{g}}(1^\kappa)$, $\mathrm{R} \xleftarrow{\$} \{0,1\}^*$ and $\mathrm{Y} := \mathsf{S}^{\mathbf{s}}(\mathrm{IK};\mathrm{R})$. Then, $\mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}$ simulates the execution of $\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}(\mathrm{IK},\mathrm{R},\mathrm{Y})$ with the only deviation that whenever Break's execution hits Line (3c) with the underlying strings ik and y, then $\mathcal{B}$ halts and returns (ik,y). If Break's execution is successfully completed without ever hitting Line (3c), then $\mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}$ gives up and returns $\perp$. Let Que

be the set of all query/response pairs that $\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{u},\mathbf{v}}$ makes to its oracles, and note $|\mathsf{Que}|$ is polynomial.

Validity of $\mathcal{B}$'s forgery output. As per Lemma 4, we need to show three things: that, (a) $((\mathrm{ik},*) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}) \notin \mathsf{Que}$, (b) $\mathbf{v}(\mathrm{ik},\mathrm{y}) = \top$ and (c) $|\mathrm{ik}| = \kappa$.

Condition (a) holds because for any $\mathbf{s}$ query $((\mathrm{ik}',\mathrm{r}') \underset{\mathbf{s}}{\rightarrow} \mathrm{y}')$ ever made by $\mathcal{B}$, this query/response is added to the set L. By the underlying if-condition of Line (3c), we have $((\mathrm{ik},*) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}) \notin \mathsf{L}$ and hence $((\mathrm{ik},*) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}) \notin \mathsf{Que}$. Condition (b) also holds immediately by the underlying if-condition of Line (3c). Finally, by Assumption 2 $|\mathrm{ik}| = \kappa$. To see this, recall from the description of Break that $\mathrm{ik} = \widetilde{\mathbf{g}}(\mathrm{tk})$ and that $|\mathrm{tk}| = \kappa$. Thus, by definition of $\widetilde{\mathbf{g}}$ we have $|\mathrm{ik}| = \kappa$, as desired. The proof is now complete. □

5 Proof of Lemma 2: general case

Sketch of the Attack. Let $(\mathrm{IK},\mathrm{R},\mathrm{Y})$ be the inputs to $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}$, where $(\mathrm{IK},\mathrm{TK}) \overset{\$}{\leftarrow} \mathsf{G}^{\mathbf{O}}(1^\kappa)$ and $\mathrm{Y} := \mathsf{S}^{\mathbf{O}}(\mathrm{IK};\mathrm{R})$. Let Q be the set of all query/response pairs during $\mathsf{S}^{\mathbf{O}}(\mathrm{IK};\mathrm{R}) = \mathrm{Y}$. Let $\mathrm{X} := \mathsf{D}^{\mathbf{O}}(\mathrm{TK},\mathrm{Y})$. Let us first try to proceed as before: sample $(\mathbf{O}',\widetilde{\mathrm{TK}})$ such that $(\mathrm{IK},\widetilde{\mathrm{TK}}) \overset{\$}{\leftarrow} \mathsf{G}^{\mathbf{O}'}(1^\kappa)$ and attempt to perform $\mathsf{D}^{\mathbf{O}'\diamond^*\mathbf{O}}$. However, things are not as simple as before. Previously, we were able to show that for any meaningful query which asks for the value of $\mathbf{e}^{-1}(\mathrm{ik},\mathrm{y})$, we must have $((\mathrm{ik},*) \underset{\mathbf{s}}{\rightarrow} \mathrm{y})$, and so Break can simulate the answer using $\mathbf{u}$. However, this does not hold here, because y may be coming from the queries made by $\mathsf{G}^{\mathbf{O}}$, to which Break does not have access.

Our solution at a high level is as follows. We work with a partial oracle $\widetilde{\mathbf{O}}$ for which initially we have $(\mathrm{IK},\widetilde{\mathrm{TK}}) \overset{\$}{\leftarrow} \mathsf{G}^{\widetilde{\mathbf{O}}}(1^\kappa)$. This oracle will then be used to invert Y (using $\widetilde{\mathrm{TK}}$) as the secret key, but since $\widetilde{\mathbf{O}}$ is not necessarily defined on all encountered queries (since it is a partial oracle) we need to "make up" answers as we go on in a *consistent* manner. Ideally, we would like to produce answers by directly resorting to $\mathbf{O}$, so to make the whole execution as close to the real execution as possible. However, this is not always possible, and so at times we need to fake some answers. Whenever, a new answer is generated (either by directly calling $\mathbf{O}$ or by faking it) we add the new query/answer pair to $\widetilde{\mathbf{O}}$ and will continue. Let us elaborate more.

Consider the execution of $\mathsf{D}^{\widetilde{\mathbf{O}}}(\widetilde{\mathrm{TK}},\mathrm{Y})$: Suppose we encounter a query qu that is not defined in $\widetilde{\mathbf{O}}$ yet. We have two cases. If qu is of type $\mathbf{g}$, $\mathbf{s}$ or $\mathbf{e}$ — namely, a query which does not require any "trapdoor" information to reply to — we will use the oracle $\mathbf{O}$ directly to answer to this query but with some case to make sure we do not introduce inconsistencies. (Remember that $\widetilde{\mathbf{O}}$ fakes some answers, so "blind" use of $\mathbf{O}$ may potentially creat inconsistencies.) If, however, qu is of $\mathbf{d}$-type, we will make use of our trapdoor-based accumulated knowledge of the oracle $\mathbf{O}$ along with the oracle $\mathbf{u}$ if we happened to have the required information. Let us give a more detailed explanation.

1. Suppose $\mathrm{qu} := ((\mathrm{ik}', \mathrm{r}') \underset{\mathbf{s}}{\rightarrow} ?)$, but $\widetilde{\mathbf{O}}(\mathrm{qu})$ is not defined yet. Suppose $\mathrm{x}' = \mathbf{s}(\mathrm{ik}', \mathrm{r}')$. We may think we can simply reply to qu with x' and add the query/response pair $\mathrm{qua} := ((\mathrm{ik}', \mathrm{r}') \underset{\mathbf{s}}{\rightarrow} \mathrm{x}')$ to $\widetilde{\mathbf{O}}$. However, we may get the following problem: There may already be a (fake) query/response $\mathrm{qua}_1 := ((\mathrm{ik}', \mathrm{x}') \underset{\mathbf{e}}{\rightarrow} \perp) \in \widetilde{\mathbf{O}}$, which would be inconsistent with qua. Thus, $\widetilde{\mathbf{O}} \cup \{\mathrm{qua}\}$ will not be TDP-consistent, and so we cannot guarantee correct inversion w.r.t. this oracle. We handle this as follows: In case of such inconsistencies, we will reply to qu with a random answer (which is unlikely to create inconsistencies) and will add the result to $\widetilde{\mathbf{O}}$.
   A same situation may hold for an $\mathbf{e}$ query and we will handle such inconsistencies in a similar manner. For $\mathbf{g}$ queries, however, we will preempt the possibility of inconsistencies by putting Break in "normal form"; see Assumption 5.
2. Suppose $\mathrm{qu} := ((\mathrm{tk}', \mathrm{y}') \underset{\mathbf{d}}{\rightarrow} ?)$, and $\mathrm{qua} := (\mathrm{tk}' \underset{\mathbf{g}}{\rightarrow} \mathrm{ik}) \in \widetilde{\mathbf{O}}$. (We will force qua to already be in $\widetilde{\mathbf{O}}$ by putting Break in normal form.) We have two cases: (a) trapdoor-available: $\mathbf{g}(\mathrm{tk}') = \mathrm{ik}$ (i.e., tk' is the real trapdoor key); or (b) trapdoor-absent: $\mathbf{g}(\mathrm{tk}') \neq \mathrm{ik}$: That is, the trapdoor key tk' has been "faked" before.
   If case (a) holds, we call the real oracle $\mathbf{O}$ on qu and will use the result as is if it leads to no inconsistencies — we, however, now have many more cases of inconsistencies, as compared to Part 1; if an inconsistency occurs, we will fake the answer.
   For case (b) we need to resort to our side trapdoor-information about $\mathbf{O}$ (e.g., set Q above: the set of all query/response pairs during $\mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R}) = \mathrm{Y}$). Also, to handle case (b), we will also need to collect all frequent trapdoor information that happen during random executions of $\mathsf{S}^{\mathbf{O}}$ and $\mathsf{E}^{\mathbf{O}}$. This collection of information is done in Step 1 of the algorithm Break.

For our analysis, we will show w.h.p. the union of $\widetilde{\mathbf{O}}_{\mathrm{uni}} \triangleq \widetilde{\mathbf{O}} \cup \mathsf{W}_1 \cup \mathsf{W}_2$ is TDP-valid, where W_1 is the (hidden) set of all queries/responses made to sample the challenge pre-image X and W_2 is the (hidden) set of all query/response pairs in $\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X})$. Note that W_1 and W_2 are not available to Break (which is the reason we called them hidden). Proving this will show that w.h.p. the decrypted result, $\widetilde{\mathrm{X}}$, by Break will be equal to X. This is because relative to $\widetilde{\mathbf{O}}_{\mathrm{uni}}$, $(\mathrm{IK}, \widetilde{\mathrm{TK}})$ is valid, X is valid (i.e., outputted by $\mathsf{S}^{\widetilde{\mathbf{O}}_{\mathrm{uni}}}(\mathrm{IK})$), $\mathrm{Y} = \mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X})$ and $\widetilde{\mathrm{X}} = \mathsf{D}^{\mathbf{O}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$.

We now proceed to describe the attack formally. We start with the following assumption.

Assumption 4. We assume that $\mathsf{G}^{\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d}}$ never calls the oracle $\mathbf{d}$. (It can predict the answer with high probability.) For notational convenience we keep $\mathbf{d}$ as a superscript to G.

We first start by describing two procedures that will be used by Break. The first procedure samples many executions of S and E in order to collect frequent trapdoors. The second procedure allows one to sample a fake secret key w.r.t. a priori information about the real oracle $\mathbf{O}$.

Definition 5 (Sampling frequent queries). *We define a probabilistic oracle procedure* $\mathsf{SFreq}^{\mathbf{O}}$*:*

- ***Input:*** $(1^\kappa, p, \mathrm{IK})$*, where* p *is an integer.*
- ***Output:*** *A set of query/response pairs* $\mathsf{Freq} \leftarrow \mathsf{SFreq}^{\mathbf{O}}(1^\kappa, p, \mathrm{IK})$ *sampled as follows. Let* $\mathsf{Freq} = \emptyset$*. Do the following* p *times:*
  - *Sample* $\mathrm{X} \leftarrow \mathsf{S}^{\mathbf{O}}(\mathrm{IK})$ *and execute* $\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X})$ *and record all query/response pairs to* Freq.

Definition 6. *We define the procedure* SOrc.

- ***Input:*** $(\mathsf{Freq}, \mathrm{IK})$*: A set of query/answer pairs* Freq *and an index key* IK.
- ***Output:*** $(\mathrm{TK}', \mathsf{Q}_g, \mathsf{Q}_s, \mathsf{Q}_e)$*, produced as follows.* Q_e *sampled as follows. Sample a* Ψ*-generated* $\mathbf{O}' = (\mathbf{g}', \mathbf{s}', \mathbf{e}', \mathbf{d}')$ *and* TK' *uniformly at random subject to the conditions that (a)* $\mathbf{O}'$ *is consistent with* Freq *(i.e.,* $\mathbf{O}' \cup \mathsf{Freq}$ *is a valid TDP) and (b)* $\mathsf{G}^{\mathbf{O}'} = (\mathrm{IK}, \mathrm{TK}')$*. Let* Q_g*,* Q_s *and* Q_e *contain, respectively, the* $\mathbf{g}$*,* $\mathbf{s}$ *and* $\mathbf{e}$ *query/response pairs made during the execution of* $\mathsf{G}^{\mathbf{O}'}$*. (Recall that by Assumption 4 no* $\mathbf{d}$ *queries are made.)*

We need the following normal-form condition for our attack algorithm.

Assumption 5. We assume the following for any oracle algorithm A with oracle access to $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$: Any query $((\mathrm{tk}, \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} ?)$ is preceded by a query $(\mathrm{tk} \underset{\mathbf{g}}{\rightarrow} ?)$. Moreover, if $\mathbf{d}(\mathrm{tk}, \mathrm{y}) = \mathrm{x} \neq \perp$, then A will make the query $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} ?)$ after making the query $((\mathrm{tk}, \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} ?)$.

Partial oracles. In the algorithm Break below we will work with partial oracles, defined only on a subset of their input queries. Specifically, for a partial oracle $\widetilde{\mathbf{O}}$ we define the following notation: We write $\widetilde{\mathbf{O}}(\mathrm{qu}) = \mathrm{null}$ to indicate $\widetilde{\mathbf{O}}$ is not defined on the query qu. This should not be confused with $\widetilde{\mathbf{O}}(\mathrm{qu}) = \perp$ as we use $\widetilde{\mathbf{O}}(\mathrm{qu}) = \perp$ to indicate that the output of $\widetilde{\mathbf{O}}(\mathrm{qu})$ is a fixed invalid symbol. We say $\widetilde{\mathbf{O}}$ is TDP consistent, if there exists a full TDP oracle $\widetilde{\mathbf{O}}_{\mathrm{full}}$ such that $\widetilde{\mathbf{O}} \subseteq \widetilde{\mathbf{O}}_{\mathrm{full}}$.

Parameter γ*.* For any Ψ-valid oracle $\mathbf{O}$ we assume that each of the algorithms $\mathsf{G}^{\mathbf{O}}$, $\mathsf{S}^{\mathbf{O}}$, $\mathsf{E}^{\mathbf{O}}$ and $\mathsf{D}^{\mathbf{O}}$ on inputs corresponding to the security parameter 1^κ make exactly κ^γ oracle queries.

The Attack Algorithm $\mathsf{Break}^{\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{v}}$*:* We describe all components of the attack algorithm.

Oracles. $(\mathbf{O}, \mathbf{u}, \mathbf{v})$. Parse $\mathbf{O} := (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$.

Input. $(1^\kappa, \mathrm{IK}, \mathrm{R})$

Output. $(\widetilde{\mathrm{X}}, \mathsf{Freq}, \widetilde{\mathbf{O}})$.[3]

1. Sample $\mathsf{Freq} \leftarrow \mathsf{SFreq}^{\mathbf{O}}(1^{\kappa}, \kappa^{2\gamma+8}, \mathrm{IK})$. Let $\widetilde{\mathbf{O}}$ and Real be two partial oracles, both initially empty.
2. Sample $(\widetilde{\mathrm{TK}}, \mathsf{Q_g}, \mathsf{Q_s}, \mathsf{Q_e}) \leftarrow \mathsf{SOrc}(\mathsf{Freq}, \mathrm{IK})$. Add $\mathsf{Q_g} \cup \mathsf{Q_s} \cup \mathsf{Q_e} \cup \mathsf{Freq}$ to $\widetilde{\mathbf{O}}$.
3. Run $\mathsf{S}^{\mathbf{O}}(\mathrm{R})$ — which gives us the challenge image Y — and add all the underlying query/response pairs to Real. Also, add all elements of Freq to Real. From this point on, all the queries made to the real oracles $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ will be recorded in Real.
4. Simulate the execution of $\mathsf{D}^{\cdot}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ and answer an encountered query qu as follows:

4.1. **Already answered in $\widetilde{\mathbf{O}}$**: if for some ans, $(\mathrm{qu}, \mathrm{ans}) \in \widetilde{\mathbf{O}}$, then reply to qu with ans;

4.2. **g-type query**: if qu is of $\mathbf{g}$-type, then reply to qu by calling the real oracle $\mathbf{g}$ and add the query/response pair to $\widetilde{\mathbf{O}}$;

4.3. **s-type query**: if $\mathrm{qu} := ((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} ?)$, then call $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{x})$. If $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \perp) \notin \widetilde{\mathbf{O}}$, then reply to qu with x and add $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{x})$ to $\widetilde{\mathbf{O}}$. Otherwise, reply to qu with $\mathrm{x}' \leftarrow \{0,1\}^{5\kappa}$ and add $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} \mathrm{x}')$ to $\widetilde{\mathbf{O}}$.

4.4. **e-type query**: if $\mathrm{qu} := ((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} ?)$ for some ik and x: Call the real oracle $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} ?)$ to get y;

4.4.1. if $\mathrm{y} = \perp$ or $((*, *) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}) \notin \widetilde{\mathbf{O}}$, then reply to qu with y add $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ to $\widetilde{\mathbf{O}}$;

4.4.2. Otherwise, reply to qu with a random $\mathrm{y}' \leftarrow \{0,1\}^{5\kappa}$ and add $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}')$ to $\widetilde{\mathbf{O}}$.

4.5. **d-type query**: if $\mathrm{qu} := ((\mathrm{tk}, \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} ?)$ for some tk and y: letting ik be such that $(\mathrm{tk} \underset{\mathbf{g}}{\rightarrow} \mathrm{ik}) \in \widetilde{\mathbf{O}}$

4.5.1. if $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}) \in \widetilde{\mathbf{O}}$, then reply to qu with x and add $((\mathrm{tk}, \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} \mathrm{x})$ to $\widetilde{\mathbf{O}}$.

4.5.2. else if $((\mathrm{ik}, \mathrm{y}) \underset{\mathbf{e}}{\rightarrow} \perp) \in \widetilde{\mathbf{O}}$ then reply to qu with $\perp$ and add $((\mathrm{tk}, \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} \perp)$ to $\widetilde{\mathbf{O}}$.

4.5.3. otherwise,

4.5.3.1. if for some tk': $(\mathrm{tk}' \underset{\mathbf{g}}{\rightarrow} \mathrm{ik}) \in \mathsf{Real}$ then call $((\mathrm{tk}', \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} ?)$ to get x:

(A) if $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} *) \notin \widetilde{\mathbf{O}}$, then reply to qu with x and add $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

(B) if $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} *) \in \widetilde{\mathbf{O}}$ then reply to qu with a random $\mathrm{x}' \leftarrow \{0,1\}^{5\kappa}$ and add $((\mathrm{ik}, \mathrm{x}') \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

[3] $\widetilde{\mathrm{X}}$ is the final result of inversion. The other two outputs, namely $\mathsf{Freq}, \widetilde{\mathbf{O}}$, are partial oracles, which are included in the output so to help us later state our security statements easier.

4.5.3.2. else if $((\mathrm{ik},\mathrm{x}) \xrightarrow[\mathbf{e}]{} \mathrm{y}) \in \mathsf{Real}$, then

(A) if $((\mathrm{ik},\mathrm{x}) \xrightarrow[\mathbf{e}]{} *) \notin \widetilde{\mathbf{O}}$ then reply to qu with x and add $((\mathrm{ik},\mathrm{x}) \xrightarrow[\mathbf{e}]{} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

(B) if $((\mathrm{ik},\mathrm{x}) \xrightarrow[\mathbf{e}]{} *) \in \widetilde{\mathbf{O}}$ then reply to qu with a random $\mathrm{x}' \leftarrow \{0,1\}^{5\kappa}$ and add $((\mathrm{ik},\mathrm{x}') \xrightarrow[\mathbf{e}]{} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

4.5.3.3. else if for some r: $((\mathrm{ik},\mathrm{r}) \xrightarrow[\mathbf{s}]{} \mathrm{y}) \in \mathsf{Real}$, then call $((\mathrm{ik},\mathrm{r}) \xrightarrow[\mathbf{u}]{} ?)$ to get r_0 and call $((\mathrm{ik},\mathrm{r}_0) \xrightarrow[\mathbf{s}]{} ?)$ to get x_0:

(A) if $((\mathrm{ik},\mathrm{x}_0) \xrightarrow[\mathbf{e}]{} *) \notin \widetilde{\mathbf{O}}$, then reply to qu with x_0 and add $((\mathrm{ik},\mathrm{x}_0) \xrightarrow[\mathbf{e}]{} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

(B) if $((\mathrm{ik},\mathrm{x}_0) \xrightarrow[\mathbf{e}]{} *) \in \widetilde{\mathbf{O}}$, then reply to qu with a random $\mathrm{x}' \leftarrow \{0,1\}^{5\kappa}$ and add $((\mathrm{ik},\mathrm{x}') \xrightarrow[\mathbf{e}]{} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

4.5.3.4. else if $\mathbf{v}(\mathrm{ik},\mathrm{y}) = \bot$ then reply to qu with $\bot$;

4.5.3.5. otherwise, reply to qu with a random $\mathrm{x}' \leftarrow \{0,1\}^{5\kappa}$ and add both of $((\mathrm{tk},\mathrm{y}) \xrightarrow[\mathbf{d}]{} \mathrm{x}')$ and $((\mathrm{ik},\mathrm{x}') \xrightarrow[\mathbf{e}]{} \mathrm{y})$ to $\widetilde{\mathbf{O}}$.

5. Letting $\widetilde{\mathrm{X}}$ be the result of the simulated execution of $\mathsf{D}^{\cdot}(\widetilde{\mathrm{TK}}, Y)$, return $(\widetilde{\mathrm{X}}, \mathsf{Freq}, \widetilde{\mathbf{O}})$.

5.1 Proof of Attack Effectiveness

We now focus on proving Lemma 2. We first start with a simple information theoretic lemma, which generalizes Lemma 4 to the case in which the "forger" may call all the underlying oracles. For that lemma, we need the following definition.

Definition 7. *Let* Q *be a set of query/response pairs obtained from oracles* $(\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{v})$. *We say that* (ik,x) *is embedded in* Q *if*

- $((\mathrm{ik},*) \xrightarrow[\mathbf{s}]{} \mathrm{x}) \in \mathsf{Q}$, *or*
- $((\mathrm{ik},*) \xrightarrow[\mathbf{e}]{} \mathrm{x}) \in \mathsf{Q}$ *or*
- *for some* tk*:* $(\mathrm{tk} \xrightarrow[\mathbf{g}]{} \mathrm{ik}) \in \mathsf{Q}$ *and* $((\mathrm{tk},*) \xrightarrow[\mathbf{d}]{} \mathrm{x}) \in \mathsf{Q}$.

The following lemma generalizes Lemma 4. The proof again follows using standard information-theoretic arguments and so is omitted.

Lemma 6. *Let* $\mathcal{B}$ *be a a polynomial-query oracle adversary. We have*

$$\Pr_{(\mathbf{O},\mathbf{u},\mathbf{v})\leftarrow\Psi}[(\mathrm{ik},\mathrm{y}) \xleftarrow{\$} \mathcal{B}^{\mathbf{g},\mathbf{s},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{v}}(1^\kappa) \text{ s.t. } |\mathrm{ik}| = \kappa$$

$$\text{and } \mathbf{v}(\mathrm{ik},\mathrm{y}) = \top \text{ and } (\mathrm{ik},\mathrm{y}) \text{ is not embedded in } \mathsf{Que}] \leq \frac{1}{2^{3\kappa}}, \quad (5)$$

where Que *is the set of all query/answer pairs of* $\mathcal{B}$.

We define the following environment that specifies a random choice of the oracles $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v})$ as well as random variables used to form a random input to an adversary against enhanced one-wayness of the construction.

Environment. The environment $\mathsf{Env}(\kappa)$ specifies the following random variables: $(\mathrm{IK}, \mathsf{Query}, \mathrm{R}_y, \mathrm{Y}, \mathrm{R}_x, \mathrm{X}, \mathbf{O})$:

- $(\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{v}) \leftarrow \Psi$. Let $\mathbf{O} := (\mathbf{g}, \mathbf{s}, \mathbf{e}, \mathbf{d})$;
- $(\mathrm{IK}, \mathrm{TK}) \leftarrow \mathsf{G}^{\mathbf{O}}(1^\kappa)$ and let Query be the set of all query/response pairs asked during the execution;
- $\mathrm{R}_y \leftarrow \{0,1\}^*$;
- $\mathrm{Y} := \mathsf{S}^{\mathbf{O}}(\mathrm{IK}, \mathrm{R}_y)$;
- $\mathrm{X} := \mathsf{D}^{\mathbf{O}}(\mathrm{TK}, \mathrm{Y})$
- $\mathrm{R}_x \leftarrow \mathsf{S}$, where
$$\mathsf{S} := \{\mathrm{R} \mid \mathsf{S}^{\mathbf{O}}(\mathrm{IK}, \mathrm{R}) = \mathrm{X}\}.$$

Notation HitQ. For an oracle algorithm $\mathsf{A}^{\mathbf{O}}$ we let $\mathsf{HitQ}(\mathsf{A}^{\mathbf{O}}(\mathrm{X}))$ denote the set of all query response pairs made during the execution of $\mathsf{A}^{\mathbf{O}}(\mathrm{X})$. If A is a randomized algorithm, then $\mathsf{HitQ}(\mathsf{A}^{\mathbf{O}}(\mathrm{X}))$ will be a random variable.

Notation $\Diamond$. For a partial oracle $\widetilde{\mathbf{O}}$ and full oracle $\mathbf{O}$ we let $\widetilde{\mathbf{O}}\Diamond\mathbf{O}$ denote the oracle that responds to a query qu as follows: if $\widetilde{\mathbf{O}}(\mathrm{qu}) \neq \mathrm{null}$ then $\widetilde{\mathbf{O}}\Diamond\mathbf{O}(\mathrm{qu}) = \widetilde{\mathbf{O}}(\mathrm{qu})$; otherwise, $\widetilde{\mathbf{O}}\Diamond\mathbf{O}(\mathrm{qu}) = \mathbf{O}(\mathrm{qu})$. Note that even if both $\widetilde{\mathbf{O}}$ and $\mathbf{O}$ are TDP consistent, $\widetilde{\mathbf{O}}\Diamond\mathbf{O}$ is not necessarily so.

Lemma 7. *Let* $(\mathrm{IK}, \mathsf{Query}, \mathrm{R}_y, \mathrm{Y}, \mathrm{R}_x, \mathrm{X}, \mathbf{O}, \mathbf{u}, \mathbf{v}) \leftarrow \mathsf{Env}(\kappa)$ *and* $(\widetilde{\mathrm{X}}, \mathsf{Freq}, \widetilde{\mathbf{O}}) \leftarrow \mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R})$.

1.
$$\Pr[\mathrm{X} = \mathsf{S}^{\widetilde{\mathbf{O}}\Diamond\mathbf{O}}(\mathrm{IK}, \mathrm{R_x}) \textit{ and } \mathsf{E}^{\widetilde{\mathbf{O}}\Diamond\mathbf{O}}(\mathrm{IK}, \mathrm{X}) = \mathrm{Y}] \geq 1 - \frac{1}{4\kappa^2} \tag{6}$$

2. Letting
$$\mathsf{ALLQ} := \widetilde{\mathbf{O}} \cup \mathsf{HitQ}(\mathsf{S}^{\widetilde{\mathbf{O}}\Diamond\mathbf{O}}(\mathrm{IK}, \mathrm{R_x})) \cup \mathsf{HitQ}(\mathsf{E}^{\widetilde{\mathbf{O}}\Diamond\mathbf{O}}(\mathrm{IK}, \mathrm{X}))$$
we have $\Pr[\mathsf{ALLQ}$ *is TDP consistent* $] \geq 1 - \frac{1}{2\kappa^2}$.

Let us first show how to use Lemma 7 to prove Lemma 2. We will then prove Lemma 7.

*Proof (**Proof of Lemma 2**).* Let all the variables be sampled as in Lemma 2. Let $\mathrm{R}_x \leftarrow \mathsf{S}$, where
$$\mathsf{S} := \{\mathrm{R} \mid \mathsf{S}^{\mathbf{O}}(\mathrm{IK}, \mathrm{R}) = \mathrm{X}\}.$$

Let Evnt_1 and Evnt_2 denote the events of Parts 1 and 2 of Lemma 7. That is,

- $\mathrm{Evnt}_1 : \mathrm{X} = \mathsf{S}^{\widetilde{\mathbf{O}}\Diamond\mathbf{O}}(\mathrm{IK}, \mathrm{R_x})$ and $\mathsf{E}^{\widetilde{\mathbf{O}}\Diamond\mathbf{O}}(\mathrm{IK}, \mathrm{X}) = \mathrm{Y}$
- $\mathrm{Evnt}_2 : \mathsf{ALLQ}$ is TDP consistent .

We claim if $\mathrm{Evnt}_1 \wedge \mathrm{Evnt}_2$ holds, then $\widetilde{\mathrm{X}} = \mathrm{X}$. This implies our result since

$$\Pr[\widetilde{\mathrm{X}} = \mathrm{X}] \geq \Pr[\mathrm{Evnt}_1 \wedge \mathrm{Evnt}_2] \geq 1 - \Pr[\overline{\mathrm{Evnt}_1}] - \Pr[\overline{\mathrm{Evnt}_2}] \geq 1 - \frac{1}{\kappa^2}.$$

It remains to prove the above claim. We show if $\mathrm{Evnt}_1 \wedge \mathrm{Evnt}_2$ then (I) ALLQ is TDP consistent, (II) $(\mathrm{IK}, \widetilde{\mathrm{TK}}) \in \mathsf{G}^{\mathsf{ALLQ}}(1^\kappa)$, (III) $\widetilde{\mathrm{X}} = \mathsf{D}^{\mathsf{ALLQ}}(\widetilde{\mathrm{TK}}, Y)$, (IV) $\mathrm{X} = \mathsf{S}^{\mathsf{ALLQ}}(\mathrm{IK}, \mathrm{R_x})$, (V) $\mathsf{E}^{\mathsf{ALLQ}}(\mathrm{IK}, \mathrm{X}) = \mathrm{Y}$. Then, by the correctness of the construction $(\mathsf{G}, \mathsf{S}, \mathsf{E}, \mathsf{D})$ we obtain $\widetilde{\mathrm{X}} = \mathrm{X}$, and the proof will be complete.

First, note that (I) follows by definition of Evnt_2.

To prove (II) and (III), first note that we have $(\mathrm{IK}, \widetilde{\mathrm{TK}}) \in \mathsf{G}^{\widetilde{\mathbf{O}}}(1^\kappa)$ and $\widetilde{\mathrm{X}} = \mathsf{D}^{\widetilde{\mathbf{O}}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$. Now since $\widetilde{\mathbf{O}} \subseteq \mathsf{AllQ}$ and since ALLQ is TDP consistent, we have $(\mathrm{IK}, \widetilde{\mathrm{TK}}) \in \mathsf{G}^{\mathsf{ALLQ}}$ and $\widetilde{\mathrm{X}} = \mathsf{D}^{\mathsf{ALLQ}}(\widetilde{\mathrm{TK}}, \mathrm{Y})$. Note that the mere fact that $\widetilde{\mathbf{O}} \subseteq \mathsf{ALLQ}$ will not be sufficient to conclude these two last statements (II) and (III); the reason is that there may be collisions between $\widetilde{\mathbf{O}}$ and $\mathsf{ALLQ} \setminus \widetilde{\mathbf{O}}$ (e.g., a query qu may receive different responses from the two oracles), rendering the corresponding executions ambiguous.

Similarly, from the facts that ALLQ is TDP consistent and that Evnt_1 holds, we conclude (IV) and (V). □

We now show how to prove Lemma 7, starting with Part 1. We give the proof of Part 2 of the lemma in the full version. To this end, we define some variables and events to help us describe things more concisely.

Sub-oracles $\widetilde{\mathbf{O}_1}$, $\widetilde{\mathbf{O}_2}$, $\widetilde{\mathbf{O}_3}$, $\widetilde{\mathbf{O}_4}$ *and set* Rand. We define four sub-oracles of $\widetilde{\mathbf{O}}$, which capture some of the query/response pairs that were added to $\widetilde{\mathbf{O}}$ as a result of faking answers for those queries that created conflict with the real oracle $\mathbf{O}$. Recall that for removing such conflicts, we sampled elements uniformly at random from $\{0,1\}^{5\kappa}$ and used those for faking answers. Informally, the set Rand contain those points sampled for these purposes. We now formally define these pieces of notation.

- $\widetilde{\mathbf{O}_1}$: We let $\widetilde{\mathbf{O}_1}$ contain any query/response pair $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}')$ added to $\widetilde{\mathbf{O}}$ as a result of Line 4.4.2..
- $\widetilde{\mathbf{O}_2}$: We let $\widetilde{\mathbf{O}_2}$ contain any query/response pair added to $\widetilde{\mathbf{O}}$ as a result of Condition (B) of Line 4.5.3.1..
- $\widetilde{\mathbf{O}_3}$: We let $\widetilde{\mathbf{O}_3}$ contain any query/response pair added to $\widetilde{\mathbf{O}}$ as a result of Condition (B) of Line 4.5.3.3..
- $\widetilde{\mathbf{O}_4}$: We let $\widetilde{\mathbf{O}_4}$ contain any query/response pair $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y})$ added to $\widetilde{\mathbf{O}}$ as a result of Line 4.5.3.5..
- Rand: We let Rand contain all x such that $((*, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} *) \in \widetilde{\mathbf{O}_2} \cup \widetilde{\mathbf{O}_3}$ or $((*, *) \underset{\mathbf{e}}{\rightarrow} \mathrm{x}) \in \widetilde{\mathbf{O}_1}$. Intuitively, the set Rand contains all points that were sampled uniformly at random for making up fake answers.

Events Surp_1*,* Surp_2*,* Surp_3*,* Surp_4*.* We define some events which we will prove can only happen with negligible probability.

- Surp_1: the event that for some $((\mathrm{ik}, *) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}') \in \widetilde{\mathbf{O}_1}$ we have $\mathbf{v}(\mathrm{ik}, \mathrm{y}') = \top$ or for some $((\mathrm{ik}, \mathrm{x}') \underset{\mathbf{e}}{\rightarrow} *) \in \widetilde{\mathbf{O}_2} \cup \widetilde{\mathbf{O}_3} \cup \widetilde{\mathbf{O}_4}$ we have $\mathbf{v}(\mathrm{ik}, \mathrm{x}') = \top$.
- Surp_2: the event that during the execution $\mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R}_\mathrm{x})$ a query $\mathrm{qu} = ((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} ?)$ or a query $((*, \mathrm{x}) \underset{\mathbf{d}}{\rightarrow} ?)$ is made where $\mathrm{x} \in \mathsf{Rand}$;
- Surp_3: the event that there exists $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}) \in \widetilde{\mathbf{O}_4}$ such that $(\mathrm{ik}, \mathrm{y})$ is not embedded in Query.
- Surp_4: For $\mathrm{x} \neq \mathrm{x}'$ and $\mathrm{y} \neq \perp$: $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}) \in \widetilde{\mathbf{O}}$ and $((\mathrm{ik}, \mathrm{x}') \underset{\mathbf{e}}{\rightarrow} \mathrm{y}) \in \widetilde{\mathbf{O}}$.

Let $\mathrm{Surp} = \mathrm{Surp}_1 \vee \mathrm{Surp}_2 \vee \mathrm{Surp}_3 \vee \mathrm{Surp}_4$.

Set Dif*.* Let $\mathsf{Q} := \mathsf{Q}_\mathsf{g} \cup \mathsf{Q}_\mathsf{s} \cup \mathsf{Q}_\mathsf{e}$, where recall that the sets Q_g, Q_s and Q_e are formed during Line 2 of the algorithm Break. Let Dif be the set of queries formed as follows: For any query/response pair $(\mathrm{qu} \underset{*}{\rightarrow} *) \in \mathsf{Q}$, add the query qu to Dif. Moreover, for any $(\mathrm{ik}, \mathrm{x})$ that occurs in $\mathsf{Query} \cup \mathsf{Q}$:

(A) if for some r: $\mathbf{s}(\mathrm{ik}, \mathrm{r}) = \mathrm{x}$ add $((\mathrm{ik}, \mathrm{r}) \underset{\mathbf{s}}{\rightarrow} ?)$ to Dif;
(B) add $((\mathrm{ik}, \mathrm{x}) \underset{\mathbf{e}}{\rightarrow} ?)$ to Dif;
(C) add $((\mathrm{tk}, \mathrm{x}) \underset{\mathbf{d}}{\rightarrow} ?)$ to Dif, where $\mathrm{tk} = \mathbf{g}^{-1}(\mathrm{ik})$;
(D) if for some x′: $\mathbf{e}(\mathrm{ik}, \mathrm{x}') = \mathrm{x}$, add $((\mathrm{ik}, \mathrm{x}') \underset{\mathbf{e}}{\rightarrow} ?)$ to Dif.[4]

Events Match *and* MissQ*.* Equipped with the set Dif we now define the following two events.

- Match: $\widetilde{\mathbf{O}} \Diamond \mathbf{O}$ agrees with $\mathsf{HitQ}(\mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R}_\mathrm{x})) \cup \mathsf{HitQ}(\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X}))$.
- MissQ:

$$\exists \langle qu \rangle \in \mathsf{Dif} \text{ s.t. } \langle qu \rangle \notin \mathsf{Freq} \text{ and } \langle qu \rangle \in \mathsf{HitQ}(\mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R}_\mathrm{x})) \cup \mathsf{HitQ}(\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X})).$$

Lemma 8. *If* $\overline{\mathrm{Match}}$ *holds, then* $\mathrm{MissQ} \vee \mathrm{Surp}$ *holds.*

Lemma 9. *We have* $\Pr[\mathrm{MissQ}] \leq \frac{1}{8\kappa^2}$*.*

Lemma 10. *We have* $\Pr[\mathrm{Surp}] \leq \frac{1}{2^\kappa}$

Proof (of Part 1 of Lemma 7). Let $\alpha(n)$ denote the probability of this part of the lemma. We have $\alpha(n) \geq \Pr[\mathrm{Match}]$. From Lemmas 8, 9 and 10 $\Pr[\overline{\mathrm{Match}}] \leq \frac{1}{4\kappa^2}$. The proof is complete. □

We give the proof of Lemma 8 in the full version. We now prove Lemma 9, for which we will use the following standard lemma.

[4] Note that we do not claim that Dif can be built efficiently. We merely introduce Dif to define a related event.

Lemma 11. *Let $x_1, \ldots, x_{t+1}$ be independent, Bernoulli random variables, where* $\Pr[x_i = 1] = p$*, for all* $i \leq t+1$*. Then*

$$\Pr[x_1 = 0 \wedge \cdots \wedge x_t = 0 \wedge x_{t+1} = 1] \leq \frac{1}{t}.$$

Proof (of Lemma 9). Let $\mathsf{Exec}^{\mathbf{O}}(\mathrm{IK})$ be the following random execution: Sample $\mathrm{X}' \leftarrow \mathsf{S}^{\mathbf{O}}(\mathrm{IK})$ and run $\mathsf{E}^{\mathbf{O}}(\mathrm{IK}, \mathrm{X}')$. Recall that Freq is formed by running $\mathsf{Exec}^{\mathbf{O}}(\mathrm{IK})$ independently $t := \kappa^{2\gamma+8}$ times. Also, note that R_x is a uniformly random string, and thus $(\mathsf{S}^{\mathbf{O}}(\mathrm{IK}; \mathrm{R_x}); \mathsf{E}^{\mathbf{O}}(\mathrm{IK}, X))$ corresponds to a random execution of $\mathsf{Exec}^{\mathbf{O}}(\mathrm{IK})$.

Using simple inspection, we may verify $|\mathsf{Dif}| \leq 6\kappa^{\gamma}$. Now applying Lemma 11 for each element of Dif and taking a union bound, we will have $\Pr[\mathrm{MissQ}] \leq 6\kappa^{\gamma}\frac{1}{\kappa^{2\gamma+8}} \leq \frac{1}{8\kappa^2}$, as desired. □

Proof (of Lemma 10). We can easily show that each of the events Surp_1, Surp_2 and Surp_4 happens with probability at most $\frac{1}{2^{3n}}$: Arguing about the probability of each of these events amounts to arguing that a randomly chosen element in $\{0,1\}^{5\kappa}$ happens to lie in a sparse subset of $\{0,1\}^{5\kappa}$. Thus, we omit the details for these parts.

We focus on bounding the probability of Surp_3. Recall that

- Surp_3: a query $((\mathrm{tk}, \mathrm{y}) \underset{\mathbf{d}}{\rightarrow} ?)$ is made for which Line 4.5.3.5. is hit and for which $(\mathrm{ik}, \mathrm{y})$ is not embedded in Query, where $(\mathrm{ik}, \mathrm{y})$ is defined as in Line 4.5.3.5.. Also, recall that the notion of embeddedness from Definition 7.

We will show that whenever the event Surp_3 holds, we can forge a pair $(\mathrm{ik}, \mathrm{y})$ in the sense of Lemma 4, obtaining $\Pr[\mathrm{Surp}_3] \leq \frac{1}{2^{3n}}$.

In order for Line 4.5.3.5. — during the simulated execution of $\mathsf{D}^{\cdot}(\widetilde{\mathrm{TK}}, \mathrm{Y})$ — to be hit with the underlying values $(\mathrm{ik}, \mathrm{y})$, all of the following must hold at that point:

(I) $((\mathrm{ik}, *) \underset{\mathbf{e}}{\rightarrow} \mathrm{y}) \notin \mathsf{Real}$ — this is because otherwise Line 4.5.3.2. would have been hit.

(II) $((\mathrm{ik}, *) \underset{\mathbf{s}}{\rightarrow} \mathrm{y}) \notin \mathsf{Real}$ — this is because otherwise Line 4.5.3.3. would have been hit.

(III) $((\mathrm{tk}_{\mathrm{real}}, *) \underset{\mathbf{d}}{\rightarrow} *) \notin \mathsf{Real}$, where $\mathrm{tk}_{\mathrm{real}} = \mathbf{g}^{-1}(\mathrm{ik})$ — this is because otherwise Line 4.5.3.1. would have been hit (by Assumption 5).

(IV) $\mathbf{v}(\mathrm{ik}, \mathrm{y}) = \top$ — this is because otherwise Line 4.5.3.4. would have been hit.

We now show show how the above conditions enable us to forge in the sense of Lemma 4. In particular, the above conditions immediately imply that $(\mathrm{ik}, \mathrm{y})$ is not embedded in Real. Also, notice that the set Real contains all those query/response pairs made by $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^{\kappa}, \mathrm{IK}, \mathrm{R})$ (to its real oracles) up to the point the event Surp_3 holds. Moreover, since Surp_3 holds, then, by definition, the pair $(\mathrm{ik}, \mathrm{y})$ is not embedded in Query either, which contains all the query/response pairs used to produce IK. We may now design a forgery attack as follows. The forger $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa)$ first samples $(\mathrm{IK}, *) \leftarrow \mathsf{G}^{\mathbf{O}}(1^\kappa)$ and then simulates $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R})$ for $\mathrm{R} \leftarrow \{0,1\}^*$. Whenever the event Surp_3 holds with the underlying pair $(\mathrm{ik}, \mathrm{y})$, then $\mathcal{B}$ will halt and return $(\mathrm{ik}, \mathrm{y})$. Note that $\mathsf{Break}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa, \mathrm{IK}, \mathrm{R})$ can efficiently recognize the occurrence of the event Surp_3. The success probability of $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{v}}(1^\kappa)$ is the probability that Bad holds. □

6 Acknowledgements

I am grateful to the anonymous reviewers for their useful comments, and especially to one reviewer for their very elaborate comments. I would also like to thank Bruce Kapron for commenting on an earlier draft of the paper.

References

[AS16] Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 512–541, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 296–315, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[BPR+08] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *49th FOCS*, pages 283–292, Philadelphia, PA, USA, October 25–28, 2008. IEEE Computer Society Press.

[BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 474–502, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[BY93] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 442–460, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[CL17] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. Cryptology ePrint Archive, Report 2017/631, 2017. http://eprint.iacr.org/2017/631.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[Fis02] Marc Fischlin. On the impossibility of constructing non-interactive statistically-secret protocols from any trapdoor one-way function. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 79–95, San Jose, CA, USA, February 18–22, 2002. Springer, Heidelberg, Germany.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317, St. Louis, Missouri, October 22–24, 1990. IEEE Computer Society Press.

[FS12] Dario Fiore and Dominique Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 636–653, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *42nd FOCS*, pages 126–135, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

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

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

[Hai04] Iftach Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 394–409, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[HHRS07] Iftach Haitner, Jonathan J. Hoch, Omer Reingold, and Gil Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *48th FOCS*, pages 669–679, Providence, RI, USA, October 20–23, 2007. IEEE Computer Society Press.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. Finding collisions on a public road, or do secure hash functions need secret coins? In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 92–105, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[KKM12] Saqib A. Kakvi, Eike Kiltz, and Alexander May. Certifying RSA. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 404–414, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999.

[Rab79] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979.

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

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.

[Vah10] Yevgeniy Vahlis. Two is a crowd? A black-box separation of one-wayness and security under correlated inputs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 165–182, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd FOCS*, pages 80–91, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.