On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates

Extended Abstract

Yael Gertner [*] Tal Malkin[†] Omer Reingold[†]

Abstract

We prove that, somewhat surprisingly, there is no black-box reduction of (poly-to-one) trapdoor functions to trapdoor predicates (equivalently, to public-key encryption schemes). Our proof follows the methodology that was introduced by Impagliazzo and Rudich [19], although we use a new, weaker model of separation.

1 Introduction

In this paper we study the following problem: Do trapdoor predicates imply (poly-to-one) trapdoor functions? In recent decades, we have gained better understanding of the relationships among these primitives and related ones, and about the possible implications of a (positive or negative) answer to this problem. The cryptographic research has also developed tools and frameworks for studying this type of problems (such as the Impagliazzo and Rudich [19] methodology we use here). Yet, despite renewed efforts in relevant directions, this fundamental problem remained open (see, e.g., Goldwasser [17]). This paper gives a *negative answer* to the above problem *with respect to black-box reductions*.

In order to place our result in context, we briefly review what was known about the primitives and the relationship between them prior to our work. We then proceed with a more detailed exposition, further elaborating on black box reductions, previous efforts, and our own result.

THE PRIMITIVES. Trapdoor (one-way) functions were introduced by Diffie and Hellman [7], in their seminal work which laid the foundations to public-key cryptography. Informally, a family of functions is *one-way* if a function F_k chosen from this family is easy to compute on any input but hard to invert on the average. Such a family is a family of *trapdoor (one-way) functions (TDF)*, if the key-generation algorithm, in addition to k (which describes the function), produces a trapdoor information tk that allows its holder to invert the function F_k. In this paper, by trapdoor functions (or TDF) we refer to (one-way) trapdoor functions which are *poly-to-one*, namely where the number of pre-images for every point in the range is polynomial (this is well justified, due to the work of Bellare et. al. [3] discussed in Section 1.2).

Trapdoor predicates, informally, are families of *probabilistic* functions over $\{0,1\}$ such that: (1) Given a key k it is easy to sample the output of the corresponding function p_k on either 0 or 1 but hard to distinguish these two distributions. (2) Given the trapdoor information tk, it is easy to distinguish an output on 0 from an output on 1. Goldwasser and Micali [18] defined trapdoor predicates and showed their equivalence to semantically secure public-key encryption (*PKE*). We will, therefore, use trapdoor predicates and PKE interchangeably throughout the paper.

THE RELATIONSHIP. PKE schemes are often viewed as essentially synonymous to trapdoor functions. Historically, almost all PKE schemes were indeed based on assumptions that were known to naturally yield trapdoor functions (exceptions will be discussed below). Yao [23] showed that injective trapdoor functions imply trapdoor predicates, and Bellare, Halevi, Sahai, and Vadhan [3] extended this result to show that general (poly-to-one) trapdoor functions imply trapdoor predicates (this is done using the Goldreich-Levin [15] hard-core bit).

In the other direction, however, the question was left open. Intuitively, the difference between trapdoor functions and trapdoor predicates is that trapdoor functions are deterministic and, given the trapdoor, allow recovery of a *complete inverse*. In contrast, trapdoor predicates are probabilistic and recover only the *bit* that the predicate was applied to (or the *message*, for a many-bit PKE scheme), but not necessarily the *randomness* used to generate the output. Therefore, in order to construct trapdoor functions from trapdoor predicates some "de-randomization" would need to occur.

[*] University of Pennsylvania ygertner@saul.cis.upenn.edu

[†] AT&T Labs — Research, 180 Park Avenue, Florham Park, NJ 07932 {tal,omer}@research.att.com

This is in fact, what was done by Bellare, Halevi, Sahai, and Vadhan [3] *in the random oracle model*. That is, they prove that, if (magically) a truly random function is available for use, trapdoor predicates can be turned into trapdoor functions. Efforts to eliminate the random oracle proved fruitless, not leaving much guidance towards proving a positive result in the standard model. (In hindsight, this is not surprising, given the negative result of our paper). Recently, Gertner, Kannan, Malkin, Reingold, and Viswanathan [10] clarified the problem further by showing that there is no black-box construction of trapdoor *permutations* from PKE. Moreover, they also proved that there is no black-box construction of trapdoor permutations from injective trapdoor functions, indicating that trapdoor permutations are indeed a stronger primitive than trapdoor functions (even injective ones). They leave open the problem of whether trapdoor *functions* can be constructed from PKE.

In this paper, we resolve this problem negatively, with respect to black-box reductions. In other words, we show that public-key encryption does not require trapdoor functions, and can be based on a strictly weaker assumption (with respect to black-box reductions). Before further discussing our result, we describe the relevant background and give further details of previous results.

1.1 Black-Box Separations

Almost all modern cryptography is complexity-theory based (the exceptions are constructs that can be proven information-theoretically secure, e.g. encryption with a one-time pad). In turn, complexity-theoretical cryptography is based on unproven assumptions (at least as strong as the existence of one-way functions, which implies $\mathcal{P} \neq \mathcal{NP}$). Hence, it is not surprising that one of the most fundamental goals in cryptography has been to identify the minimal assumptions under which major cryptographic primitives can be proven secure.

Consider two primitives P and Q, each of which can be constructed under a (possibly different) unproven, yet plausible, assumption. Demonstrating that (the existence of) Q implies (the existence of) P may take the form of a reduction. That is, showing how to transform any implementation of Q to an implementation of P (this transformation is usually an efficient one), such that if the implementation of Q is secure, the resulting implementation of P is also secure. It is much less clear how to demonstrate or even state the converse, namely that Q *does not imply* P. Indeed, in what sense the implication does not exist, if it is plausible that both primitives exist? (after all, a reduction could ignore Q and build P from scratch).

While we cannot completely overcome this difficulty, Impagliazzo and Rudich [19] gave a method for separating primitives under a restricted but important subclass of reductions, namely, *black-box reductions*. Intuitively, a black-box reduction of P to Q is a construction of P out of Q that ignores the internal structure of the implementation of Q, and just uses it as a "subroutine". Furthermore, the security of the resulting implementation of P can be based solely on the security of the implementation of Q and this proof has a black-box flavor as well (that is, an adversary that breaks Q can be constructed using only oracle calls to Q and the adversary that breaks P). The exact definition of black-box reductions and the methodology for black-box separations are somewhat subtle, and details are given in Section 2.1. An important property of black-box reductions is that they work relative to any oracle. Therefore, one way to prove that a black-box reduction of P to Q is impossible is to prove the existence of an oracle relative to which Q exist whereas P does not. Nevertheless, as we observe in this work, it also suffices to demonstrate that, assuming a reduction existed, there is an oracle relative to which the primitive Q exists, whereas the supposed construction of P is *not* secure. We note that [19] and subsequent works used a few stronger variants of this separation methodology (see Section 2.1).

As noted by Impagliazzo and Rudich, almost all constructions in cryptography *are black box*. Moreover, the only reductions which are not black-box, are highly inefficient (though still polynomial time). Therefore, if there are no black-box reductions of P to Q then a proof that the existence of Q implies the existence of P is most likely very difficult to find. In other words, it is unlikely that P can be constructed from Q using known techniques, and even more unlikely that such a construction could be efficient. Still, we stress that black-box separations have their limitations. Indeed, some reductions used in cryptography are *not black-box*. An important (and almost single) example of a non black-box reduction is the proof that "all of NP has zero knowledge proofs" [16] and its implications (see [19] for a more detailed discussion of oracle separations in cryptography).

Using the above methodology, Impagliazzo and Rudich [19] proved that there is no black-box reduction of one-way functions to key agreement, thus separating the large body of primitives which are known to be equivalent to one-way functions, from the primitives which are known to imply key-agreement, in particular, TDF and PKE.

Subsequent works that used black-box separations include [21, 22, 20, 9, 10], some of which will be elaborated upon in the sequel.

1.2 Efforts Towards Understanding the Relationship Between PKE and TDF

As mentioned above, Bellare et. al. [3] used Goldreich-Levin [15] to show that poly-to-one trapdoor functions imply trapdoor predicates (thus generalizing the result of Yao [23] which applied only to injective trapdoor functions). On the other hand, Bellare et. al. [3] showed that trapdoor functions with super-polynomial pre-image size can be constructed from one-way functions (thus, by Impagliazzo and Rudich [19], separating between such super-poly-to-one trapdoor functions and trapdoor predicates). In a sense, this result means that trapdoor functions are "interesting" only when they are poly-to-one (as we assume by default throughout this paper).

Bellare et. al. [3] also addressed the converse, asking whether trapdoor predicates imply trapdoor functions (the question we focus on in this paper). They prove that such an implication exists in the random oracle model.[1] In this model, it is assumed that a truly random function is available for all parties to use. Essentially, their trapdoor function construction uses this random oracle to de-randomize PKE, by feeding the input message to the random oracle function and using its output as randomness for encrypting the message; thus, the randomness is determined by the message (via the random oracle), and so recovering just the message is enough to find an inverse to the function. The next natural step for proofs in this model, is trying to replace the random oracle with an actual (efficient) construction, such that the proof still carries through. In this case, the natural candidate seems to be a pseudo-random generator. However, Bellare et. al. [3] show that such a generic replacement would not work. Specifically, they show that (assuming the existence of PKE) there exist a PKE scheme P and pseudorandom generator G, such that their transformation of P into a TDF using G instead of the random oracle is not secure (as it becomes easy to invert).

Recently, Gertner et. al. [10] have explored the relationships among several cryptographic primitives. In particular, they prove that PKE and oblivious transfer are incomparable under black-box reductions, and as a consequence (since trapdoor permutations imply both these primitives), there is no black-box construction of trapdoor *permutations* from PKE. They leave open the problem of whether trapdoor *functions* can be constructed from PKE. Moreover, [10] also prove that there is no black-box construction of trapdoor permutations from injective trapdoor functions, indicating that trapdoor permutations are indeed a stronger primitive than trapdoor functions (even injective ones).

1.3 Our Result

We answer the long standing open problem of whether trapdoor predicates imply trapdoor functions, with respect to black-box reductions. Our answer is negative. Informally, our result means that in an essential way trapdoor predicates cannot be de-randomized. We display our result along with previous ones that were discussed above, in Figure 1.

As was pointed out by Bellare et. al. [3], their positive result in the random oracle model implies that proving a black-box separation (which is the goal we set to achieve) is a difficult task. Intuitively, a careless attempt to use an oracle towards separation would fail, since the oracle may be used to de-randomize the public-key cryptosystem into a trapdoor function (as in [3] or otherwise). Indeed, our separation is quite complex, and it holds in a weaker model than previous separations; specifically, rather than using a single separating oracle, we construct for every candidate black box reduction an oracle relative to which trapdoor predicates exist, but the given construction does not yield poly-to-one TDF. Thus, in our model the candidate construction cannot access the oracle that exhibits the separation. Still, our model is strong enough to imply that no black-box reductions from trapdoor functions to trapdoor predicates exist. More generally, our model may be useful to separate other primitives, where stronger separations have not been found. More details about our black-box separation methodology appear in Section 2.1.

In our proof we make the following assumption regarding the reduction of TDF to PKE (see Assumption 2 in Section 3.2). Informally, we assume that the structural properties of the TDF family depend only on the structural properties of the given PKE. More specifically, by structural properties of the TDF family we mean the correctness of the inversion algorithm (given the trapdoor key), and the polynomial bounds on the pre-image size and on the running times of the trapdoor function and the inverting algorithm; By structural properties of the PKE system we mean its correctness (namely the fact that decryption works correctly with respect to the key generation and encryption), the running time of the key generation, encryption, and decryption functions, and their output lengths. We find this assumption to be fairly reasonable (it is hard to imagine a reduction where these properties depend on the security of the PKE scheme in an intrinsic way). In fact, the actual assumption that we need can be made substantially weaker.

Finally, an interesting result that follows from our proof, is that there is no black-box reduction of trapdoor functions to 1-round (i.e., 2-pass) honest oblivious transfer (OT). A similar result (in a stronger separation model) for k-pass OT ($k \geq 3$) is given in [10], following from their result that k-pass OT does not imply PKE in a black-box manner. However, their proof does not apply to 2-pass OT (in fact they show that 2-pass OT *does* imply PKE). We show that because of the properties of the PKE in our oracle, it yields 2-pass OT, implying that there is no black-box construction of trapdoor functions out of 2-pass OT.

Our result that PKE does not require trapdoor functions

[1] They in fact prove that in the random oracle model trapdoor predicates imply *injective* trapdoor functions.

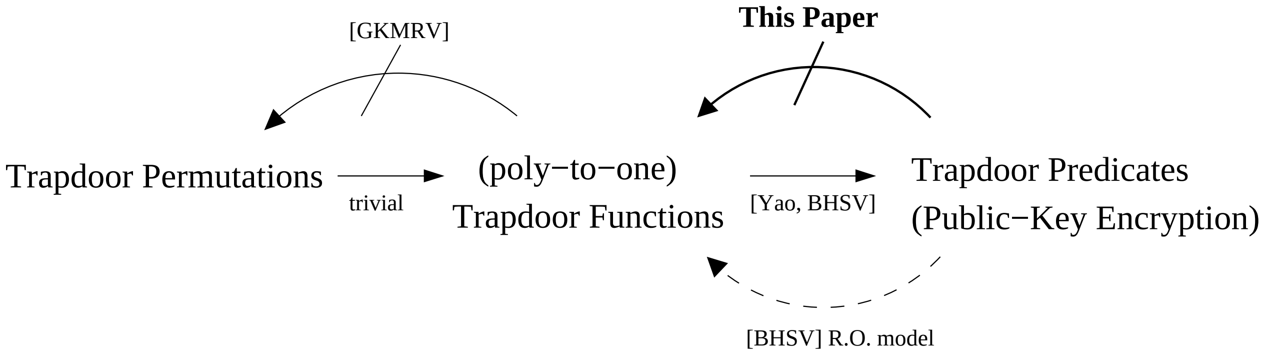

Figure 1. Previously known relations and our result. All (non) implications are with respect to black-box reductions. The dotted implication is in the random-oracle model.

further motivates the search of new assumptions with the hope of basing cryptography (and in particular PKE) on the weakest assumptions possible. Below we briefly review previous efforts towards this end (apart from the related work we already described).

1.4 Efforts Towards Basing PKE on Weaker Assumptions

Historically, almost all PKE schemes were based on assumptions that were known to naturally yield trapdoor functions. One notable exception was the El-Gamal scheme [8], which is based on the hardness of taking discrete logarithms, and is semantically secure under the Decisional Diffie-Hellman assumption. (The more recent Cramer-Shoup scheme [5] is also based on the same assumption).

The problem of constructing PKE based on weaker assumptions experienced a revival with the work of Ajtai [1], who constructed a one-way function based on the worst-case hardness of lattice reduction. Following that, Ajtai and Dwork [2] proposed a public-key cryptosystem which is secure based on the hardness of the unique shortest vector problem. This provides a new example of PKE which is based on an assumption that is not known to yield trapdoor functions.[2]

These results open a new domain of plausible hardness assumptions on which PKE (and other cryptographic primitives) may be based. Moreover, these results are based on *worst case* instances of lattice reduction problems. Therefore, they along with several results about the $\mathcal{NP}$-hardness of approximating related lattice problems, provide a glimmer of hope towards basing cryptography on the assumption that $\mathcal{P} \neq \mathcal{NP}$. However, there is a gap between the known hardness results and those hardness assumptions required for the above schemes to be secure, and in fact Goldreich and Goldwasser [12] point to difficulties in trying to bridge this gap. For further details we refer the reader to the discussion of Goldreich and Goldwasser [13], who, in light of the abovementioned results, revisit the possibility of basing cryptography (and in particular PKE) on the assumption that $\mathcal{P} \neq \mathcal{NP}$.[3] Their conclusion is that this problem remains open.

2 Preliminaries

2.1 Black-Box Reductions

A (standard) reduction of a primitive P to a primitive Q consists of a construction of P out of Q, namely a construction of a polynomial time machine $M(C)$ for every polynomial time machine C, and a proof that whenever C yields an implementation of Q, $M(C)$ yields an implementation of P. This is equivalent to showing that for every adversary that breaks $M(C)$, there exists an adversary that breaks C. A *black-box reduction* of P to Q is a reduction that ignores the internal structure of Q's implementation, and, moreover, the proof of correctness is also black-box; that is, the adversary breaking Q ignores the internal structure of the implementation of Q and of the (alleged) adversary breaking P.

[2] Interestingly, the PKE scheme of Goldreich Goldwasser, and Halevi [14], which was also inspired by Ajtai's work, *is* in fact based on a conjectured trapdoor function. This trapdoor function relies on an assumption related to a random version of the closest vector problem.

[3] In their work, [13] generalize an old result of Brassard [4], showing that if a restricted form of PKE can be based on an $\mathcal{NP}$-hard problem, and certain conditions on the scheme and on the reduction apply, then $\mathcal{NP} = co\mathcal{NP}$.

Ignoring the internal structure is modeled by granting only oracle access (rather than, say, the description of the algorithm), to the relevant machine. Somewhat more formally, a black-box reduction of P to Q, is a construction of polynomial time oracle machines M and A_Q such that whenever A_P breaks M^C (as an implementation of P), $A_Q^{A_P,C}$ breaks C itself (as an implementation of Q). In particular, both the construction and the proof are independent of the computational complexity of C and A_P. Clearly, if a a black-box reduction of P to Q exists, we can conclude that Q implies P, since for any implementation C of Q, M^C is an implementation of P.

The Oracle Separation Paradigm. Let P, Q be two cryptographic primitives. A *relativizing reduction* of P to Q is a reduction that works relative to *any oracle* Γ (namely a construction of $M^\Gamma(C)$ such that for any Γ and C, whenever C^Γ is an implementation of Q relative to Γ, $M^\Gamma(C)$ is an implementation of P relative to Γ). It is not hard to see that this notion is less strict than black-box reductions. Therefore, in order to prove that there is no black-box reduction of P to Q, it suffices to construct an oracle Γ and prove that relative to Γ the primitive Q exists, whereas P does not.

The oracle separations in [19] and in all subsequent works used paradigms that were at least as powerful as the one described above. In particular, they all prove that there are no relativizing reductions between the relevant primitives they deal with, from which they conclude that there is no black-box reduction, either. We note that in some cases, most notably in the work of Gennaro and Trevisan [9], even stronger conclusions can be drawn (namely a wider variety of reductions is ruled out).

In contrast, in this paper we use the following separation paradigm: We show that for every candidate black-box construction of P out of Q (in our case P is TDF and Q is PKE), there is an oracle Γ such that relative to Γ an implementation of Q exists, whereas the candidate construction of P does not work. Thus, using the same notation as above, for every efficient M, we construct Γ and an efficient C such that C^Γ cannot be broken as an implementation of Q relative to Γ, yet M^{C^Γ} *can* be broken as an implementation of P (namely we show an efficient A_P such that A_P^Γ breaks M^{C^Γ}). This immediately implies that no black box reduction exists, since for any M, there is no A_Q that works in a black box manner (since $A_Q^{C^\Gamma, A_P^\Gamma}$ can be efficiently implemented with access to Γ alone, it cannot break C^Γ).

While our paradigm implies that no black-box reduction exists, it does not imply that no relativizing reduction exist. Indeed, we prove that for every construction M there exists an oracle Γ relative to which the construction fails, while to exclude all relativizing reductions one must show that there exists one oracle Γ relative to which every construction M fails. Thus, we do not rule out the possibility that for every oracle there is a different construction.

We see no inherent reason why our result cannot be extended to show that there are no relativizing reductions of trapdoor functions to trapdoor predicates. However, it seems like such a result cannot be achieved through a simple and straight-forward extension of our work, especially given that our current proof is already quite involved (throughout the exposition of our result, we try to demonstrate why more naive approaches fail).

2.2 Weak Encryption

We already informally described TDF. Formal definitions are omitted from this extended abstract, and we refer the reader to, e.g., [11].

While the "right" definition of PKE security is semantic security as defined by Goldwasser and Micali [18], it will be convenient for us to use a weaker notion, which only guarantees that it is hard to recover the entire message m from its ciphertext c (although it may be easy to gain partial information about m). The following definition differs from standard semantically secure PKE in the last requirement:

Definition 1 *A* **weak PKE** *scheme is a triple $\langle G, E, D\rangle$ of probabilistic polynomial time algorithms (without loss of generality D is deterministic) such that:*

- *$G(1^n)$ outputs a pair (sk, pk).*
- *For every message m of length n, any (sk, pk) output by G on input 1^n, and all coin tosses of E, it holds that $D(sk, E(pk, m)) = m$.*
- *For every probabilistic polynomial time A and for every polynomial $poly(\cdot)$ there exists an n_0 such that $\forall n > n_0$*

  $$\Pr[(sk, pk) \leftarrow G(1^n), m \leftarrow \{0,1\}^n, c \leftarrow E(pk, m) : A(pk, c, 1^n) = m] \;<\; 1/poly(n)$$

  (where probability is taken over the random coins of G, E, A, and over the random choice of m).

When the secret key generated by G is simply its randomness, we will slightly abuse notation and write $G(sk) = pk$.

The notion of weak PKE is not strong enough in terms of security, but it is sufficient for our purpose here, since, as we claim below, semantically secure PKE is reducible to weak PKE. Therefore, if there is a black-box reduction of trapdoor functions to semantically secure PKE, there is also a black-box reduction of trapdoor functions to weak PKE.

Lemma 1 *For every triple $\langle G, E, D\rangle$ there exists a triple $\langle G', E', D'\rangle$ such that if the former is a weak PKE scheme, the latter is a semantically secure PKE scheme.*

We conclude that it suffices for us to prove our result with respect to weak PKE, namely show that for any candidate black-box construction of TDF out of weak PKE, there is an oracle relative to which weak PKE exists, but the resulting construction is not TDF. In the remainder of this paper, by PKE we refer to weak PKE.

3 The Oracle Separation

In this section we formally define our separation of PKE from TDF. The methodology we use is the following (see discussion in Section 2.1): First we define an oracle O that is composed of a triple of oracles $\langle G,E,D\rangle$ that is a PKE scheme relative to O (with probability 1). Suppose now that one could construct, using a black box reduction, a (poly-to-one) TDF from any (weak) PKE. Let $\langle G'^O, F^O, T^O\rangle$ be the TDF that are obtained from $\langle G,E,D\rangle$ by this reduction. We define an oracle R such that (with probability 1) relative to $\langle O,R\rangle$, the triple $\langle G,E,D\rangle$ remains a (weak) PKE, whereas $\langle G'^O, F^O, T^O\rangle$ is not a poly-to-one TDF, thus deriving a contradiction to the hypothesis that a black box reduction exists.

The rest of the section is organized as follows: In Section 3.1 we define the oracle O. In Section 3.2 we discuss the (alleged) TDF $\langle G'^O, F^O, T^O\rangle$. In Section 3.3, we define the oracle R, which is in the heart of our separation. In Section 3.4 we give our main separation theorem.

3.1 The oracle O

Definition 2 *(of the oracle O)*

- *A random length tripling function, G. We interpret G as a key generating algorithm $G(sk)=pk$.*
- *A random length tripling injective function, E. We interpret E as an encryption algorithm $E(pk,r,m)=c$.*
- *The decryption algorithm D: If $G(sk)=pk$ and for some r and m of the same length as pk, $E(pk,r,m)=c$, then $D(sk,c)=m$; otherwise, $D(sk,c)=\perp$.*

The definition of the oracle O is quite straightforward. It is fairly easy to show that the triple $\langle G,E,D\rangle$ is a PKE scheme relative to O. In fact (as will be used in the paper), $\langle G,E,D\rangle$ is a PKE scheme even against a *computationally unbounded* adversary that can query O only in a polynomial number of places. A more challenging task would be to prove that $\langle G,E,D\rangle$ is a weak PKE scheme relative to $\langle O,R\rangle$. The restriction that E is injective ensures perfect decryption. We note that this restriction does not change the distribution of E in an essential way (a random length tripling function is with probability 1 injective on all but a finite number of input lengths). That E is length tripling implies some additional properties of $\langle G,E,D\rangle$ that are important for our proof. For example, the only feasible way to come up with a ciphertext (an element in the range of E) is by querying E. Letting G be a length tripling function is an arbitrary choice (made to unify some of our notation).

3.2 The Trapdoor Function

Recall that our hypothesis is that a black box reduction of TDF to any weak PKE exists. Now, let $\langle G'^O, F^O, T^O\rangle$ be the TDF obtained from $\langle G,E,D\rangle$ by this reduction: The key generating algorithm G'^O takes as input an n-bit random string s and outputs (tk,k) where k is the function description (i.e. the key) and tk is the trapdoor key. The function F_k^O on input x outputs $F_k^O(x)=y$, and the inversion algorithm T_k^O has the property that $F_k^O(T_k^O(tk,y))=y$ whenever there exist some s and x such that $G'^O(s)=(tk,k)$ and $y=F_k^O(x)$. We view n as the security parameter of the TDF. Without loss of generality, k is also an n-bit string and the domain of F_k^O is $\{0,1\}^n$.

In our proof we use the following two assumptions about the family $\langle G'^O, F^O, T^O\rangle$:

Assumption 1 *The key generating algorithm G'^O does not query D. Furthermore, tk is composed of all the queries and answers made to G and E during the run of G'. Without loss of generality, whenever F_k^O queries for $D(sk,c)$ it first queries for $G(sk)$.*

Assumption 1 is only made for the sake of presentation. That is, we can prove our results without it. Note that, since G and E are chosen at random, G' cannot substantially gain from a query for the value of D on some input (sk,c): In case G' already "knows" that $G(sk)=pk$ and $c\in\{E(pk,\cdot,m)\}$ (by previously making the corresponding queries) then it also knows that $D(sk,c)=m$; Otherwise, with overwhelming probability $D(sk,c)=\perp$. In both cases, since the reply to the query is predictable, there is no point in making it.

Assumption 2 *For any possible value $\bar{O}$ of the oracles O, if $G^{\bar{O}}(s)=(tk,k)$ and $y=F_k^{\bar{O}}(x)$ then $F_k^{\bar{O}}(T_k^{\bar{O}}(tk,y))=y$. Furthermore, for some constant $\alpha>0$, for any possible value $\bar{O}$ of the oracles O, for any n-bit string s and $(tk,k)=G'^{\bar{O}}(s)$ the following holds:*

1. *The running time of $G'^{\bar{O}}$ on any n-bit input and the running times of $F_k^{\bar{O}}$ and $T_k^{\bar{O}}(tk,\cdot)$ (on their domain) is bounded by n^α.*
2. *$F_k^{\bar{O}}$ is n^α-to-one.*

Assumption 2 is on the structure of the reduction our proof excludes. As discussed in Section 1, we find this assumption to be fairly reasonable. In fact, the actual assumption that we need is substantially weaker. For example, we only need to assume that $T_k^{\bar{O}}(tk, y)$ inverts a polynomial fraction of the outputs of $F_k^{\bar{O}}$. However, this and additional possible weakening of the assumption introduce further complications in our separation. For the sake of readability, we describe the simpler version of the separation in this extended abstract.

3.3 The Oracle R

The heart of our oracle separation is the definition of the oracle R such that relative to $\langle O, R\rangle$, the triple $\langle G, E, D\rangle$ remains a (weak) PKE whereas $\langle G'^O, F^O, T^O\rangle$ is not a TDF. The formal definition we give in Subsection 3.3.2 is quite complex and at least at first sight may not be very intuitive. In order to clarify this definition, we start by discussing some relevant intuition.

Since relative to $\langle O, R\rangle$ we want there to be an adversary that breaks the trapdoor function $\langle G'^O, F^O, T^O\rangle$, it seems natural to start by defining R to break the trapdoor function itself. The most straightforward definition of the oracle R would therefore be: on input (k, y), where k is a key of F^O and y is an output of F_k^O, the oracle R returns some input x such that $F_k^O(x) = y$ (if such an x exists). With such a definition all that is left to prove is that $\langle G, E, D\rangle$ is still a weak PKE relative to $\langle O, R\rangle$. However, this claim is in fact false. Consider the following example for which R can break the weak PKE. The key generating algorithm G'^O starts by invoking G (of the PKE) on a random input sk to obtain $pk = G(sk)$. It then keeps pk as part of the public key k. The function F_k^O is defined such that $F_k^O(x) = 0 \ldots 0$ if $G(x) = pk$, and otherwise it returns some arbitrary function of x. In this example, an adversary A that gets as input the public key pk of the PKE can derive the corresponding secret key sk by querying R on $(pk, 0 \ldots 0)$ (and thus A breaks $\langle G, E, D\rangle$ even as a weak PKE).

We conclude that this definition of R is too strong. Instead we use a weaker definition, where R is allowed to refuse to answer some set of "dangerous" queries. A dangerous query would be a query (k, y) of an adversary A with input (c, pk) (a ciphertext and public key of the PKE) that may give information about the corresponding secret key or message (such as the query $(pk, 0 \ldots 0)$ in the example above). The challenge is to define the set of queries that R refuses to answer in such a way that (1) This set includes all the queries that are indeed "dangerous" (and thus the weak PKE is still secure), but nevertheless (2) R provides a valid answer to a non-negligible fraction of the queries (k, y), (and thus R still breaks the trapdoor functions). We note that while in the previous example it was easy to identify the dangerous query, our analysis will have to handle more subtle examples. Loosely speaking, this analysis demonstrates that the dangerous queries (k, y) can be divided into ones where (c, pk) is "encoded" into k in some malicious way and ones where c is "encoded" into y (the case where pk is also "encoded" into y turns out to be less interesting). We elaborate on these cases below.

Handling invalid keys The first obstacle in the definition of R is to define its response to a query (k, y) where k is an invalid key (i.e. k cannot be obtained by invoking G'^O). The two most natural possibilities seem to be: (1) Invert F_k^O regardless of whether k is valid or not. (2) Reject any input (k, y), where k is invalid. As it turns out, both of these solutions can give rise to dangerous queries:

1. An invalid key k may not have a corresponding trapdoor key that allows for the efficient inversion of F_k^O. In such a case, inverting F_k^O may be dangerous. For example, we may have invalid keys of the form $k = pk$ for which $F_k^O(m, r) = E(pk, r, m)$. If R is willing to invert F_k^O for such invalid keys it can be easily used to break the encryption scheme $\langle G, E, D\rangle$.

2. An oracle R that rejects invalid keys gives a test for the validity of keys. This in itself may be enough to break $\langle G, E, D\rangle$. Consider for example a key generating algorithm G'^O that first invokes G on a random input sk to obtain $pk = G(sk)$. Further assume that part of the public key k generated by G'^O includes the string (pk, i, σ) where σ is the i-th bit of sk for a random index i. It is not hard to see that a validity test for keys generated by this specific G'^O gives an efficient way to invert G, thus shuttering the security of $\langle G, E, D\rangle$.

We now discuss a (somewhat unexpected) solution to the problem of invalid keys: Instead of inverting the function F_k relative to the oracle O where k may or may not be a valid key, the oracle R will invert it relative to another oracle $\tilde{O}$ where k *is definitely a valid key*. This is of course useless in breaking $\langle G'^O, F^O, T^O\rangle$ unless F_k^O and $F_k^{\tilde{O}}$ are sufficiently similar. To obtain this property, $\tilde{O}$ is defined such that it agrees with O on a list of queries L that contains all the queries that are likely to be asked in $F_k^O(\bar{x})$ for a random $\bar{x}$ (the list L will also contain all the "short" queries for some technical reasons we omit in this intuitive discussion).

To be a bit more specific, R will select a trapdoor key tk' and will define an oracle $\tilde{O}$ such that (tk', k) is in the range of $G'^{\tilde{O}}$. In order to invert $F_k^{\tilde{O}}$ on y, the oracle R will simulate $T_k^{\tilde{O}}(tk', y)$ (that is, simulate T_k relative to the oracle $\tilde{O}$ and using the trapdoor key tk').

Dangerous queries revisited Consider an adversary A with input (c, pk) and access to R and O that tries to break the PKE. The way the definition of R uses T is an important step towards ensuring that A could be *simulated without access to* R (which will imply that A must fail). Loosely, the only scenario where this simulation cannot be carried through is when A queries R on (k, y) and the simulation of $T_k^{\tilde{O}}(tk', y)$ requires R in order to evaluate $D(sk, c)$ (for the sk corresponding to pk). Therefore such queries (k, y) are exactly the "dangerous" ones. As mentioned above, these queries fall into two types:

(1) A query (k, y) **that encodes** (c, pk) **into** k**:** This essentially means that G'^O, when generating (tk, k), queries E on (m, r, pk) and gets c as output. This kind of queries are handled by the definition of tk' and $\tilde{O}$. Loosely, $\tilde{O}$ is altered on any query that is likely to be related to tk (given the key k). Therefore, the simulation of $T_k^{\tilde{O}}(tk', y)$ will give no new information on tk.

(2) A query (k, y) **that encodes** c **into** y**:** The most natural way y can encode c is if the evaluation of $y = F_k^O(x)$ queries E on (m, r, pk) and gets c. Intuitively however, the adversary A should not be able to create such a value y without knowing r: since F_k^O is invertible, it should be possible to retrieve r by applying $T_k^O(tk, y)$. Since the message c in itself is not sufficient for this purpose (even given the secret key sk), y should contain additional information that implies r. Since A only knows c it is therefore bound to fail in generating such a y. A bit more formally, assuming towards contradiction that A succeeds in creating such a query, we will prove the existence of another algorithm A' with input c and *the secret key* sk that retrieves r without querying R.

As it turns out, c can be "encoded" into y even when c *is not obtained* by a query to E during the evaluation of $y = F_k^O(x)$. Loosely speaking, to handle such queries, R keeps track of every ciphertext c that is part of an "informative" query to D made during the simulation of $x' = T_k^{\tilde{O}}(tk', y)$. For every such c, the oracle R verifies that c is also obtained by a query to E during the evaluation of $F_k^O(x')$. If this is not the case, R refuses to answer the query (k, y). We prove that this does not usually happen for a random query (k, y).

3.3.1 Notation

In order to define the oracle R formally, we need to introduce some notation.

Definition 3 *The* **query list** *of a procedure M with oracle access to the oracle $\vec{B} = \{B_1, B_2, \ldots\}$ is the list of all values (q, i, a) such that M queries B_i on q and $a = B_i(q)$. The oracle $\vec{B}$ is* **consistent** *with a list of values of the form (q, i, a), if for any such value (q, i, a) in the list $a = B_i(q)$.*

Definition 4 **The** B**-list** *of a procedure M with oracle access to the oracle B and possibly other oracles is the list of all values (q, a) such that M queries B on q and $a = B(q)$. A list L is* **a** B**-list** *if it consists of values (q, a) such that $a = B(q)$.*

Definition 5 *Let $\hat{E}$ and $\hat{D}$ be two oracles, let L_1 be an $\hat{E}$-list, and let L_2 be a $\hat{D}$-list.* $\mathbf{L_2} \prec \mathbf{L_1}$ *if for every entry $((*, c), m) \in L_2$, $((*, *, m), c) \in L_1$. Otherwise,* $\mathbf{L_2} \not\prec \mathbf{L_1}$.

Definition 6 *Let $\hat{O}$ be the oracle $\hat{O} = \langle \hat{G}, \hat{E}, \hat{D} \rangle$ and let $\hat{L}$ be a query list. A query (sk, c) to $\hat{D}$ made by $T_k^{\hat{O}}(tk', y)$ is* **informative wrt.** $\hat{\mathbf{L}}$ *if (1) $\hat{D}(sk, c) \neq \perp$, (2) T did not previously query $\hat{E}$ on some q s.t. $\hat{E}(q) = c$, (3) $(*, 2, c) \notin \hat{L}$.*

3.3.2 Formal Definition of R

As discussed above to answer a query (k, y), the oracle R simulates $T_k^{\tilde{O}}(tk', y)$. Most of R's definition (Steps 1 to 3 below) is devoted to defining how tk' and $\tilde{O}$ are selected.

Let $\ell_1(\cdot)$, $\ell_2(\cdot)$ and $\ell_3(\cdot)$ be three polynomials that are determined by our analysis (and depend on the constant α in Assumption 2). On input (k, y), where k is n-bit long, R performs the following procedure:

Step 1. Let $\ell_1 = \ell_1(n)$, $\ell_2 = \ell_2(n)$ and $\ell_3 = \ell_3(n)$. Pick uniformly at random ℓ_1 n-bit strings $u_1, \ldots, u_{\ell_1}$. Run F_k on $u_1, \ldots u_{\ell_1}$ and let L be the union of the query lists of $F_k^O(u_1), \ldots, F_k^O(u_{\ell_1})$. Furthermore, add into L all the triples $(q, 1, G(q))$ and $(q, 2, E(q))$ for all queries q that are up to $\log \ell_3$-bit long.

Remarks: With high probability, L contains all the queries that are frequently asked by F_k^O. Since $\tilde{O}$ will be defined to agree with O on all except for a polynomial number of queries, and since it will be defined to be consistent with L, it will follow that $F_k^O(x) = F_k^{\tilde{O}}(x)$ on almost all inputs. Furthermore, L contains all the queries that are shorter than $\log \ell_3$ bits and therefore $\tilde{O}$ and O agree on these queries.

Step 2. Sample a trapdoor key tk' by picking uniformly at random from the space $\langle s, \bar{O}, tk' \rangle$, where s is a random input to G', and the oracle $\bar{O}$ has the same distribution as O under the restriction that $\bar{O}$ is consistent with L, and $G'^{\bar{O}}(s) = (tk', k)$. If such tk' does not exist, output $\perp$ and halt. Recall that tk' is composed of all the queries made by $G'^{\bar{O}}(s)$ to $\bar{G}$ and $\bar{E}$ (see Assumption 1 in Section 3.2).

- Set $TK = tk'$ and let the list C contain all the ciphertexts c such that $(*, 2, c) \in tk'$.

Furthermore, sample $\ell_2 - 1$ additional trapdoor keys $tk^1, \ldots, tk^{\ell_2 - 1}$ under the same distribution independently at random.

- For every $(q,\sigma,*) \in tk^i$, if $(q,\sigma,*) \notin TK \cup L$, set $TK = (q,\sigma,a) \cup TK$, where a is a uniform string three times longer than q.
- Add to C all the ciphertexts c such that $(*,2,c) \in tk^i$ for some i and $(*,2,c) \notin L$.

Remarks: $\tilde{O}$ will be defined to be consistent with TK and, in particular, it will be consistent with tk'. Therefore, tk' will be a trapdoor key that corresponds to k in $\tilde{O}$ (i.e. (tk', k) will be an output of $G'^{\tilde{O}}$).

C contains all the ciphertexts c that are in tk'. This will be used in the definition of $\tilde{O}$ to prevent collision between outputs of $\tilde{E}$ taken from tk' and outputs of $\tilde{E}$ inherited from E.

TK and C also have an additional part that is defined by $tk^1, \ldots, tk^{\ell_2 - 1}$ and plays a different role: TK contains all the queries that are in the trapdoor keys $tk^1, \ldots, tk^{\ell_2 - 1}$, whereas C contains all the ciphertexts c from $tk^1, \ldots, tk^{\ell_2 - 1}$ that are not already in L. With high probability, these queries (resp. ciphertexts) include all the queries (resp. ciphertexts) that are likely to be in tk. Therefore, the way TK and C are incorporated into O will guarantee that $\tilde{O}$ (and in particular, $T_k^{\tilde{O}}$) does not not leak any additional information about tk.

Step 3.

Definition 7 *(of the oracle $\tilde{O}$) Let $\tilde{O}$ be the oracle $\langle \tilde{G}, \tilde{E}, \tilde{D} \rangle$ where*

- *$\tilde{G}$: If $(sk, 1, pk) \in TK$ then $\tilde{G}(sk) = pk$. Otherwise $\tilde{G}(sk) = G(sk)$.*
- *$\tilde{E}$: If $((pk, r, m), 2, c) \in TK$ then $\tilde{E}(pk, r, m) = c$. Otherwise let $c = E(pk, r, m)$, if $c \notin C$ then $\tilde{E}(pk, r, m) = c$. Otherwise, select $\tilde{E}(pk, r, m)$ uniformly at random.*
- *$\tilde{D}$: If $\tilde{E}(\tilde{G}(sk), r, m) = c$ then $\tilde{D}(sk, c) = m$. Otherwise $\tilde{D}(sk, c) = \perp$.*

If $\tilde{E}$ is not injective output $\perp$ and halt.

Remarks: The functions $\tilde{E}$ and E are almost identical. The only exceptions are the values of $\tilde{E}$ that are taken from tk' and those outputs of $\tilde{E}$ that are chosen at random by R (either in the definition of TK or when selecting $\tilde{E}(pk, r, m)$ in case $E(pk, r, m) \in C$).

Since the length of each list tk^i and tk' is at most n^α, the lengths of TK and C are bounded by $\ell_2 n^\alpha$. The number of outputs of $\tilde{E}$ that are chosen at random by R is at most the size of the lists TK and C put together which is at most $2\ell_2(n)n^\alpha$.

Step 4. Define $x' = T_k^{\tilde{O}}(tk', y)$. Let QT be the list of informative queries to $\tilde{D}$ with respect to TK and L that were made by $T_k^{\tilde{O}}(tk', y)$. Define $F_k^O(x') = z$ and let QFX' be the E-list of $F_k^O(x')$. If $z = y$ and $QT \prec QFX'$, output x'. Otherwise output $\perp$.

3.4 The Main Theorem

We are now ready to give the main theorem of our separation:

Theorem 2 *With probability 1 over the choice of the oracles $\langle O, R \rangle$, we have that relative to $\langle O, R \rangle$ the triple $\langle G, E, D \rangle$ is a weak PKE, whereas $\langle G'^O, F^O, T^O \rangle$ is not a poly-to-one one-way TDF.*

The proof of Theorem 2 follows the intuition given above (mainly in Section 3.3). Nevertheless, the complete proof is significantly more involved and is omitted here for lack of space.

3.5 Separating 1-round honest OT and TDF

As an extension of our main separation, we prove that there is no black-box reduction of trapdoor functions to 2-pass (i.e., 1-round) honest oblivious transfer. Below we briefly describe what was previously known about the relationship between OT and TDF, and how our construction yields the extension to 2-pass OT. All implications we refer to below are with respect to black-box reductions.

Gertner et. al. [10] show that k-pass OT for $k \geq 3$ does not imply PKE. Since TDF imply PKE (as proved by Bellare et. al. [3]), it follows that k-pass OT for $k \geq 3$ does not imply TDF. However, this result does not extend to 2-pass OT (which does imply PKE), leaving the relationship between 2-pass OT and TDF open. For the other direction, while [10] prove that in genreal PKE does not imply OT, they also prove that when the given PKE satisfies certain properties, it does imply OT. In particular, they consider a property which was also defined (in another context) by Damgaard and Nielsen [6] who called it oblivious public key generation. A PKE (G, E, D) satisfies oblivious public key generation, if it is possible to efficiently select a string pk with a distribution which is indistinguishable from that of a public key generated by G, while preserving the semantic security of the encryption $E(pk, \cdot, \cdot)$. Namely, even the algorithm that selects pk does not learn anything on a message m from its encryption $E(pk, r, m)$ (with random r). It is proven in [10] that a PKE satisfying oblivious public key generation implies honest 2-pass OT.[4]

[4] The OT protocol begins by Bob sending to Alice two strings pk, pk', one of which he obtained by running G and one of which was sampled obliviously. Alice uses them to encrypt her two secrets and sends to Bob, who can decrypt one of them but not the other.

We show that the PKE $\langle G, E, D\rangle$ of our oracle construction does in fact satisfy oblivious public key generation: it is easy to sample a random string pk of length $3n$, which is indistinguishable from a string that was generated by $G(sk)$ for a random sk of length n. Furthermore, choosing pk in this manner maintains the semantic security of $E(pk, \cdot, \cdot)$, as can be proven using fairly standard techniques. Thus, the same proof of the main theorem holds for the stronger result: weak PKE with the oblivious public key generation property does not imply (in black box reduction) trapdoor functions. In addition, it follows from the proof of Lemma 1 that weak PKE satisfying oblivious public key generation implies semantically secure PKE satisfying this property. Since semantically secure PKE satisfying this property implies 2-pass honest OT, we have a separation between 2-pass honest OT and trapdoor functions.

References

[1] M. Ajtai. Generating hard instances of lattice problems. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1996.

[2] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1997.

[3] M. Bellare, S. Halevi, A. Sahai, and S. Vadhan. Many-to-one trapdoor functions and their relations to public-key cryptosystems. In *Advances in Cryptology – Crypto '98 Proceedings*, Lecture Notes in Computer Science, 1998.

[4] G. Brassard. Relativized cryptography. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 383–391, 1979.

[5] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – Crypto '98 Proceedings*, Lecture Notes in Computer Science, 1998.

[6] I. Damgård and J. Nielsen. Improved non-commiting encryption schemes based on a general complexity assumption. In *Advances in Cryptology – Crypto '00 Proceedings*, Lecture Notes in Computer Science, pages 432–450, 2000.

[7] W. Diffie and M. Hellman. New directions in cryptography. pages 644–654, 1976.

[8] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. pages 469–472, 1985.

[9] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2000.

[10] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2000.

[11] O. Goldreich. Foundations of cryptography: fragments of a book. In *http://www.wisdom.weizmann.ac.il/ oded/frag.html*, 1995.

[12] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1998.

[13] O. Goldreich and S. Goldwasser. On the possibility of basing cryptography on the assumption that $p \neq np$. In *Cryptology ePrint Archive, Report 1998/005. available from http://eprint.iacr.org/.*, 1998.

[14] O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems. In *Advances in Cryptology – Crypto '97 Proceedings*, Lecture Notes in Computer Science, 1997.

[15] O. Goldreich and L. Levin. A hard predicate for all one-way functions. In *Proceedings of the ACM Symposium on the Theory of Computing*, 1989.

[16] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proofs. *Journal of ACM*, 1991.

[17] S. Goldwasser. New directions in cryptography: Twenty some years later. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS-97)*, pages 314–325, 1997.

[18] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[19] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, 1989.

[20] J. H. Kim, D. Simon, and P. Tetali. Limits on the efficiency of one-way permuation-based hash functions. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1999.

[21] S. Rudich. The use of interaction in public cryptosysytems. In *Advances in Cryptology – Crypto '91 Proceedings*, pages 242–251, 1991.

[22] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions. In *Proceedings of EUROCRYPT*, 1998.

[23] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Symposium on Foundations of Computer Science, IEEE*, 1982.