J. Cryptology (2006) 19: 97–114
DOI: 10.1007/s00145-005-0317-1

Journal of
CRYPTOLOGY

The Impossibility of Basing One-Way Permutations on Central Cryptographic Primitives*

Yan-Cheng Chang
Division of Engineering and Applied Sciences,
Faculty of Arts and Sciences,
Harvard University,
Cambridge, MA 02135, U.S.A.
ycchang@eecs.harvard.edu

Chun-Yuan Hsiao
Computer Science Department, Boston University,
Boston, MA 02215, U.S.A.
cyhsiao@cs.bu.edu

Chi-Jen Lu
Institute of Information Science, Academia Sinica,
Taipei, Taiwan
cjlu@iis.sinica.edu.tw

Communicated by Mihir Bellare

Received 17 March 2003
Online publication 2 August 2005

Abstract. We know that trapdoor permutations can be used to construct all kinds of basic cryptographic primitives, including trapdoor functions, public-key encryption, private information retrieval, oblivious transfer, key agreement, and those known to be equivalent to one-way functions such as digital signature, private-key encryption, bit commitment, pseudo-random generator and pseudo-random functions. On the other hand, trapdoor functions are not as powerful as trapdoor permutations, so the structural property of permutations seems to be something special that deserves a more careful study. In this paper we investigate the relationships between one-way permutations and all these basic cryptographic primitives. Following previous works, we focus on an important type of reductions called black-box reductions. We prove that no such reductions exist from one-way permutations to either trapdoor functions or private information retrieval. Together with previous results, all the relationships with one-way permutations have now been established, and we know that no such reductions exist from one-way

* A conference version of this paper appeared in *Advances in Cryptology—ASIACRYPT* '02, pages 110–124, Lecture Notes in Computer Science, volume 2501, Springer-Verlag, Berlin, 2002. Work by Yan-Cheng Chang and Chun-Yuan Hsiao was done while they were with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

permutations to any of these primitives except trapdoor permutations. This may have the following meaning, with respect to black-box reductions. We know that one-way permutations imply none of the primitives in "public cryptography," where additional properties are required on top of "one-wayness" [17], so permutations cannot be traded for any of these additional properties. On the other hand, we now know that none of these additional properties can be traded for permutations either. Thus, being a permutation seems to be something orthogonal to those additional properties on top of one-wayness. Like previous non-reducibility results [27], [17], [28], [29], [20], [11], [10], [12], [8], [9], our proofs follow the oracle separation paradigm of Impagliazzo and Rudich [17].

Key words. Cryptographic primitives, Black-box reductions, One-way permutations, Trapdoor functions, Private information retrieval.

1. Introduction

Modern cryptography has provided us with all kinds of protocols for various interesting and important tasks involving security issues. However, almost all of these protocols have their security based on some intractability assumptions which all imply $\mathcal{P} \neq \mathcal{NP}$. So unconditional proofs of security for these protocols may seem far beyond our reach. One important line of research then is to understand the relationships among these assumptions. However, there are many interesting cryptographic tasks, and even a single task may have several variants. So potentially the whole picture could become very messy and have little help in clarifying our understanding. Instead, we want to focus on the most basic cryptographic tasks in their most primitive forms, which can serve as building blocks for more advanced protocols. We also restrict ourselves to the classical world of cryptography, and leave questions in quantum cryptography for future studies.

According to [11], such basic cryptographic primitives can be roughly divided into two categories: private cryptography and public cryptography.[1] Private cryptography is represented by private-key encryption, and includes one-way permutation (OWP), one-way function (OWF), pseudo-random generator (PRG), pseudo-random function (PRF), bit commitment (BC), and digital signature (DS). Public cryptography is represented by public-key encryption (PKE), and includes trapdoor permutation (TDP), trapdoor function (TDF), oblivious transfer (OT), private information retrieval (PIR), and key agreement (KA). "One-wayness" turns out to be essential as these primitives all are known to imply one-way functions [16], [26], [3], [2], [11]. For private cryptography, one-wayness basically is also sufficient as one-way functions can be used to construct all the primitives therein, except one-way permutations. For public cryptography, additional properties are required on top of one-wayness, and the relationships among primitives appear to be rather complicated. We know that trapdoor permutations imply all of them, but some implications among others are known to fail, in the sense to be discussed next.

It is not quite clear how to prove that one primitive Q does not imply the other primitive P, or equivalently P cannot be reduced to Q, especially when both primitives exist under some plausible assumptions. After all, if the primitive P exists, there is a protocol of P

[1] This classification is just a convenient one for us and is by no means a precise or complete one. The situation becomes complicated when one wants to talk about variations of primitives meeting additional requirements (e.g., [29] and [8]).

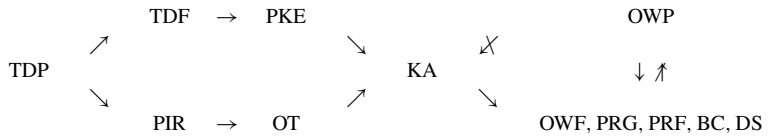

Fig. 1. Relationships between OWP and other cryptographic primitives.

based on Q that simply ignores Q. Facing the difficulty of showing non-existence of *all* reductions, Impagliazzo and Rudich [17] turned to a restricted but important subclass of reductions called *black-box reductions*, and gave a method to separate primitives with respect to them. Informally speaking, a black-box reduction from P to Q is a construction of P out of Q that ignores the internal structure of the implementation of Q. Furthermore, the security of P's implementation can also be guaranteed in a black-box way that one can use any adversary breaking P as a subroutine to break Q. In fact, in cryptography almost all constructions of one primitive from another known so far are done in this way,[2] so it makes sense to focus on reductions of this kind. Hereafter, all the reductions or implications we refer to in this paper will be black-box ones. Following [17], to prove that no black-box reduction exists from P to Q, it suffices to construct an oracle relative to which Q exists whereas P does not. Using this approach, Impagliazzo and Rudich [17] showed that no such black-box reduction exists from KA to OWP. As every primitive in public cryptography implies KA [3], [6], [11], this provides a strong evidence that primitives in public cryptography require strictly more than one-wayness. Since then, more and more separations between cryptographic primitives [29], [11], [12], [8] as well as results on efficiency lower bound [28], [20], [10], [9] have been established following this oracle separation paradigm.

We know that trapdoor permutations imply all those basic cryptographic primitives, but it is not the case for trapdoor functions as they do not imply OT [11] and thus PIR [6]. So there seems to be something special for being a permutation which deserves further study. We also know that one-way functions do not imply one-way permutations [27], [18], so permutation does not seem to be a property that one can have for free. We know that one-way permutations imply none of the primitives in public cryptography [17], so on top of one-wayness, one cannot trade permutations for any of the additional properties required in public cryptography. Then the question we want to ask is: Can any of those additional properties required in public cryptography be traded for permutations? Formally, can any of the primitives except TDP in public cryptography imply OWP? Figure 1 summarizes the relationships known so far between primitives and OWP.[3] We will show that neither TDF nor PIR implies OWP, so the answer to that question is actually no!

We first construct an oracle, relative to which (injective) TDF exists whereas OWP does not. As TDF implies PKE [30], [3] and PKE (two-pass KA) implies KA, we

[2] As pointed out in [17], one important exception is that the existence of OWF implies the existence of zero-knowledge proof systems for all languages in $\mathcal{NP}$.

[3] Some known non-reducible relationships (with respect to black-box reductions) are omitted in the figure; e.g., PKE $\not\rightarrow$ OT, PKE $\not\leftarrow$ OT, PKE $\not\rightarrow$ TDF [11], [12]. Such omissions do not affect the known relationship between OWP and each primitive, which is the focus of our paper.

establish the impossibility of having black-box reductions from OWP to either TDF, PKE, or KA. Next, we construct an oracle, relative to which PIR exists whereas OWP does not. Because PIR implies OT [6], we establish that no black-box reduction exists from OWP to either PIR or OT. One immediate corollary is that PIR does not imply TDP, in contrast to the known result that TDP does imply PIR [22].[4] So according to our results, none of the primitives in public cryptography implies OWP in a black-box way. This is interesting in the sense that all the powerful primitives, except TDP, in public cryptography, are still unable to yield OWP.[5]

Our results suggest that permutation is really a special property that is orthogonal to other additional properties required in cryptography. Furthermore, as the reducibility from each primitive to OWP was already known before, now all the relationships, with respect to black-box reductions, between one-way permutations and those basic cryptographic primitives have been established. However, we want to stress that we are still far from being able to settle the real relationships among primitives, and in fact, to have separations beyond black-box reductions would require some major breakthrough in complexity theory [17].

For each separation between primitives, we need to find a suitable oracle that is powerful enough for making one primitive possible, but still not so for the other. We basically follow the approach of Impagliazzo and Rudich [17] and Gertner et al. [11]. It is known that a random function is one-way with high probability, even relative to a $\mathcal{PSPACE}$-complete function [17]. Then OWF exists relative to an oracle containing a random function and a $\mathcal{PSPACE}$-complete function, but on the other hand, OWP does not exist relative to such an oracle [27], [18]. We want to separate OWP from TDF and PIR. Each time we look for a special function which realizes the additional property required by that primitive but does not yield permutations. By adding such a function to the oracle, we can build the corresponding primitive, TDF or PIR, but relative to the oracle, OWP still does not exist. Our strategy of finding such special functions is based on the observation that both TDF and PIR can be seen as two-party primitives while OWP involves only one party. So we look for those functions that are useful in a two-party setting but are useless in a one-party case.

The rest of the paper is organized as follows. In Section 2 we describe our notation and provide definitions for the cryptographic primitives involved in this paper. Then, in Sections 3 and 4, we prove that no black-box reductions exist from OWP to TDF and PIR, respectively.

2. Notation and Definitions

Let $[n]$ denote the set $\{0, 1, \ldots, n-1\}$. For $x \in \{0,1\}^n$, let $x[i]$ denote the ith bit of x if $i \in [n]$, and an arbitrary value, say 0, otherwise. We write $poly(n)$ to denote a polynomial in n. We write $*$ for $\{0,1\}^*$ and $(*, q, *)$ for those (u, q, v) with $u, v \in \{0,1\}^*$. For a

[4] TDP is only known to imply "weak" PIR, in the sense that the resulting PIR in [22] has high communication complexity. More specifically, the server needs to send at least a constant fraction of the entire database to the user. Under various number-theoretical assumptions, more efficient PIRs are known [21], [4].

[5] We do not consider variants of primitives satisfying additional conditions, such as dense PKE, homomorphic PKE, non-interactive zero-knowledge proof of knowledge, etc.

distribution S, we write $s \in_R S$ to denote sampling s according to the distribution S. For any $n \in \mathbb{N}$, let U_n denote the uniform distribution over $\{0,1\}^n$.

Parties in cryptographic primitives are assumed to run in polynomial time, and are modeled by probabilistic polynomial-time Turing machines (PPTM). Each cryptographic primitive is associated with a security parameter k, for evaluating how secure that primitive is. A *negligible* function is a function which vanishes faster than any inverse polynomial.

Definition 1. A function $\delta\colon \mathbb{N} \to \mathbb{R}$ is called *negligible* if for any constant $c \geq 0$ there exists an integer k_c such that $\delta(k) < k^{-c}$ for every $k \geq k_c$.

In the following we give standard definitions of the cryptographic primitives studied in this paper. More details can be found in standard textbooks (e.g., [13]) or the original papers. The most fundamental primitive is *one-way function*, which is essential to all cryptographic primitives.

Definition 2 (OWF). A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is called *one-way* if the following two conditions hold:

1. There exists a (deterministic) polynomial-time algorithm which on any input x outputs $f(x)$.
2. For every probabilistic polynomial-time algorithm A there exists a negligible function δ such that for sufficiently large n,

$$\Pr_{x \in_R U_n}[A(1^n, f(x)) \in f^{-1}(f(x))] \leq \delta(n).$$

From one-way functions, we define primitives with additional properties. Call a function $f\colon \{0,1\}^* \to \{0,1\}^*$ a *permutation* if for any $n \in \mathbb{N}$, f maps $\{0,1\}^n$ to $\{0,1\}^n$ in a bijective way, or, equivalently, f acts as a permutation on $\{0,1\}^n$.

Definition 3 (OWP). A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is called a *one-way* permutation if in addition to being *one-way*, it is a permutation.

Instead of talking about a single function $f\colon \{0,1\}^* \to \{0,1\}^*$, sometimes it is convenient to talk about a collection of functions, each defined over some finite domain. We can define a collection of functions or permutations to be one-way in the following way.

Definition 4. Let S be a set of indices and for $i \in S$ let D_i and R_i be finite sets. A collection of functions $\mathcal{F} = \{f_i\colon D_i \to R_i\}_{i \in S}$ is called *one-way* if it satisfies the following four conditions:

1. There is a PPTM I which on input 1^k outputs an index $i \in \{0,1\}^k \cap S$.
2. There is a PPTM J which on input $i \in S$ outputs an $x \in D_i$.
3. There is a PPTM which on input $i \in S$ and $x \in D_i$ outputs $f_i(x)$.
4. For any PPTM A there exists a negligible function δ such that for sufficiently large k,

$$\Pr_{i \in_R I(1^k), x \in_R J(i)}[A(1^k, i, f_i(x)) \in f_i^{-1}(f_i(x))] \leq \delta(k).$$

$\mathcal{F}$ is called a collection of one-way permutations if in addition to being one-way, every f_i in it is a permutation.

Next, we define collections of trapdoor functions, which are collections of one-way functions with the additional property that when given some *trapdoor* information of a function, it becomes easy to invert.

Definition 5 (TDF/TDP). Let S be a set of indices and for $i \in S$ let D_i and R_i be finite sets. $\mathcal{F} = \{f_i \colon D_i \to R_i\}_{i \in S}$ is called a collection of *trapdoor functions* if it satisfies the following five conditions:

1. There is a PPTM I which on input 1^k outputs a pair (i, t_i), where $i \in \{0,1\}^k \cap S$ is the index for the function f_i and $t_i \in \{0,1\}^{k^c}$, for some constant c, is called the *trapdoor* for f_i.
2. There is a PPTM J which on input $i \in S$ outputs an $x \in D_i$.
3. There is a PPTM which on input $i \in S$ and $x \in D_i$ outputs $f_i(x)$.
4. There is a PPTM such that given input $(i, f_i(x), t_i)$, for $i \in S$, $x \in D_i$ and the trapdoor t_i for f_i, it outputs some $x' \in f^{-1}(f_i(x))$.
5. For any PPTM A there exists a negligible function δ such that for sufficiently large k,
$$\Pr_{i \in_R I(1^k), x \in_R J(i)} \left[A(1^k, i, f_i(x)) \in f_i^{-1}(f_i(x)) \right] \leq \delta(k).$$

$\mathcal{F}$ is called a collection of trapdoor permutations if in addition to being a collection of trapdoor functions, every f_i in it is a permutation.

In this paper when we talk about trapdoor functions (permutations), we always mean a collection of them satisfying the conditions above. We drop the terms "a collection of" for notational convenience.

Finally, we describe private information retrieval, which was introduced by Chor et al. [5]. Here we consider the version in which there are two parties, one user and one server, and the security is only guaranteed in a computational sense [21], [4], [2], [6], [22]. The task is for the user to learn some bit of the server's database secretly, conditioned on a non-trivial upper bound on the server's communication complexity.

Definition 6 (PIR). *Private information retrieval* is a protocol between two parties *Server* and *User*, both running in polynomial time. Server has a database $x \in \{0,1\}^n$, User has an index $i \in [n]$, and the security parameter k is given to them as 1^k. They then communicate with each other, and finally User decides on a value for $x[i]$. Let $C(x, i, 1^k)$ denote the distribution of their communication. A PIR protocol must satisfy the following two conditions:

- Correctness: For any $x \in \{0,1\}^n$ and for any $i \in [n]$, User always learns $x[i]$ and Server always sends less than n bits to User.
- User's security: For any $PPTM$ A there exists a negligible function δ such that for any $x \in \{0,1\}^n$, for any $i, j \in [n]$, and for sufficiently large k,
$$\left| \Pr_{z \in_R C(x,i,1^k)} [A(1^k, x, z) = 1] - \Pr_{z' \in_R C(x,j,1^k)} [A(1^k, x, z') = 1] \right| \leq \delta(k).$$

2.1. *Black-Box Reductions*

Next, we discuss the notion of black-box reductions. The formalization which we adopt here is due to Gertner et al. [11].[6]

Definition 7. There exists a *black-box reduction* from primitive P to primitive Q, if there exist two oracle PPTMs M and A_Q satisfying the following two conditions:

1. Correctness: For any N that implements Q, M^N implements P.
2. Security: For any N that implements Q and for any A_P, if A_P breaks M^N (as an implementation of P) then $A_Q^{A_P,N}$ breaks N (as an implementation of Q).

Note that the term "implement" above is only used to mean that the functional specification of the primitive is met, and it says nothing about security. For example, to implement OWP, it suffices to provide a polynomial-time algorithm that computes a permutation, and whether or not it is hard to invert is irrelevant. So the first condition only says that if N satisfies the functional requirement of Q, then M^N satisfies the functional requirement of P. The security issue is addressed by the second condition. It essentially says that if M^N is not secure (as an implementation of P), neither is N (as an implementation of Q); or, equivalently, if N is secure, so is M^N.

Most reductions we know today fall into this category. For example, when proving "OWP implies PRG," we are given an OWP f and asked to build an oracle PPTM M such that M^f computes a length-increasing function, say from n bits to $n+1$ bits, for every n. Then we proceed by building another oracle PPTM A such that for any distinguisher D that can distinguish $M^f(U_n)$ from U_{n+1} with a non-negligible probability, $A^{D,f}$ can invert f with a non-negligible probability.

To show that there is no black-box reduction from P to Q, it suffices to construct an oracle Γ relative to which Q exists whereas P does not. The reason is the following. Assume that there is an oracle PPTM N such that N^Γ securely implements Q, but no PPTM with oracle access to Γ can securely implement P. Now suppose, for the sake of contradiction, that there is a black-box reduction from P to Q, with M and A_Q being the associated oracle PPTMs. M^{N^Γ} is an implementation of P, but it cannot be secure and must be broken by some oracle PPTM A with access to Γ. However, according to the definition of black-box reduction, this implies that N^Γ can be broken by $A_Q^{A^\Gamma,N^\Gamma}$, which is an oracle PPTM with access to Γ. This contradicts the assumption that N^Γ is a secure implementation of Q.

3. TDF Does Not Imply OWP

In this section we construct an oracle Γ relative to which there are (injective) trapdoor functions but no one-way permutations. It is shown in [27] and [18] that no OWP exists relative to an oracle with a $\mathcal{PSPACE}$-complete problem and some random functions.

[6] Recently, Reingold et al. [24] gave a more careful taxonomy of the ways in which black-box reductions can be formalized. We refer interested readers to their paper. The black-box reduction we use here corresponds to their fully-black-box reduction.

We add a function G into such an oracle to do the inverting job when provided with the trapdoor, and we want G to be useless in constructing OWP. Our oracle Γ consists of the following:

- A $\mathcal{PSPACE}$-complete problem.
- A length-tripling random function $F(\cdot,\cdot)$, with inputs of the form $\{0,1\}^n \times \{0,1\}^{3n}$ for every n. This will play the role as a collection of one-way functions. The first argument is the input, while the second is the index of a function in the collection.
- A length-tripling random function $H(\cdot)$. We call it using the trapdoor information as input and the output will be used as the index of a function in the collection of one-way functions.
- A function G defined as follows:

$$\forall (u,v) \in \{0,1\}^* \times \{0,1\}^*, \qquad G(u,v) = \begin{cases} w & \text{if} \quad \exists w\colon\ u = F(w, H(v)), \\ \perp & \text{otherwise.} \end{cases}$$

  G will be used to invert F on u when given the trapdoor information v.

In Γ the functions F and H are random while the function G is completely determined by F and H. Call a query to G *invalid* if its answer is $\perp$, and *valid* otherwise. Note that we can assume without loss of generality that both F and H are injective, because one can show that length-tripling random functions are injective on sufficiently long inputs with measure one.[7]

G is designed in this way for the following purpose. The function $F(\cdot, H(t))$ can be inverted if one has t, because for any x,

$$G(F(x, H(t)), t) = x.$$

Without knowing t, queries to G are likely to be invalid and thus useless. As we will see, this makes the construction of trapdoor functions possible. On the other hand, the function G is not helpful in a one-party primitive (OWP in particular), for the following reason. To have a valid query $G(y,t)$, y is likely to come from a query $F(x, H(t))$ for some x, but then one knows $x = G(y,t)$ already, which makes such a query to G unnecessary. Our approach basically follows those of [17] and [11].

3.1. *TDF in* Γ

On input 1^k, the trapdoor-function generator I outputs the pair $(H(t), t)$, where $t \in_R U_k$ is the trapdoor and $H(t)$ is the index for the function $F(\cdot, H(t))$. For convenience, we write $F_{H(t)}(\cdot)$ to denote the function $F(\cdot, H(t))$, and assume its domain is $\{0,1\}^k$. Given the index $H(t)$, the function $F_{H(t)}$ is easy to compute, just by querying the oracle $F(\cdot, H(t))$. Having the trapdoor t, $F_{H(t)}$ is easy to invert, with the help from the oracle G as

$$\begin{aligned} G(F_{H(t)}(x), t) &= G(F(x, H(t)), t) \\ &= x. \end{aligned}$$

It remains to show that $F_{H(t)}$ is hard to invert without knowing the trapdoor t.

[7] The reason is the following. For any $n \in \mathbb{N}$, there are at most 2^{2n} pairs of inputs of length n, each being mapped to the same output with probability $1/2^{3n}$. Since $\sum_n 2^{2n}/2^{3n}$ converges, by the *Borel–Cantelli lemma* (see the Appendix), measure one of length-tripling random functions are injective when inputs are long enough.

Consider any oracle PPTM M as an inverter. We know from [27] that a random function is one-way with high probability, even relative to a $\mathcal{PSPACE}$-complete oracle. So without the oracle G, $F_{H(t)}$ is a random function and is one-way with high probability Now we will prove that even with the oracle G, $F_{H(t)}$ is still one-way with high probability The idea is that unless M^Γ can guess the trapdoor t correctly, G is unlikely to provide useful information for inverting $F_{H(t)}$. Formally, for a negligible function $\delta(k)$, we want to upper-bound the probability

$$\Pr_{\Gamma}\left[\Pr_{x,t}\left[M^\Gamma(F_{H(t)}(x), H(t)) = x\right] > \delta(k)\right],$$

which by the Markov inequality is at most

$$\operatorname*{Ex}_{\Gamma}\left[\Pr_{x,t}\left[M^\Gamma(F_{H(t)}(x), H(t)) = x\right]\right]/\delta(k) \;=\; \Pr_{\Gamma,x,t}\left[M^\Gamma(F_{H(t)}(x), H(t)) = x\right]/\delta(k).$$

We need the following lemma.

Lemma 1. $\Pr_{\Gamma,x,t}[M^\Gamma(F_{H(t)}(x), H(t)) = x] \le k^c 2^{-k}$, *for some constant* c.

Proof. Define the following probability event:

- B_1: M^Γ on input $(F_{H(t)}(x), H(t))$ queries H on t or G on $(*, t)$.

B_1 corresponds to the bad event that M "knows" the trapdoor t. The following claim shows that it is unlikely to happen.

Claim 1. $\Pr_{\Gamma,x,t}[B_1] \le poly(k)2^{-k}$.

Proof. Fix any x, t and consider any restriction Γ_0 of Γ that leaves only $H(t)$ random. Note that $G(*, t)$ is not fixed yet as it depends on the value of $H(t)$. We claim that although M has $(F_{H(t)}(x), H(t))$ as the input, the probability of B_1 happening remains the same if the input is replaced by $(F_h(x), h)$ for a random h independent of $H(t)$. The idea is that whether or not B_1 happens is completely determined by the input value given to M; it is independent of the value of $H(t)$ or $G(*, t)$, for the following reason. If M never queries H on t or G on $(*, t)$, B_1 does not happen. Otherwise, the first time M makes such a queries, B_1 happens and remains so afterwards, regardless of the value returned by the query. Therefore, we have

$$\begin{aligned}
\Pr_{\Gamma,x,t}[B_1] \;&=\; \operatorname*{Ex}_{x,t,\Gamma_0}\left[\Pr_{H(t)}\left[M^{\Gamma_0}(F_{H(t)}(x), H(t)) \text{ queries } H \text{ on } t \text{ or } G \text{ on } (*, t)\right]\right] \\
&=\; \operatorname*{Ex}_{x,t,\Gamma}\left[\Pr_{h}\left[M^{\Gamma}(F_h(x), h) \text{ queries } H \text{ on } t \text{ or } G \text{ on } (*, t)\right]\right] \\
&=\; \operatorname*{Ex}_{x,\Gamma,h}\left[\Pr_{t}\left[M^{\Gamma}(F_h(x), h) \text{ queries } H \text{ on } t \text{ or } G \text{ on } (*, t)\right]\right] \\
&\le\; poly(k)2^{-k},
\end{aligned}$$

where the second equality follows from the discussion above, and the last inequality is because M makes at most $poly(k)$ queries. □

Next, we want to show that if the bad event B_1 does not happen, M^Γ is unlikely to invert the input correctly. We may assume without loss of generality that M^Γ always uses its output to query $F_{H(t)}$ at the final step before it stops. This does not affect its inverting probability, which is then bounded above by the probability of the following event:

- B_2: M^Γ on input $(F_{H(t)}(x), H(t))$ queries $F_{H(t)}$ on x.

It remains to prove the following claim.

Claim 2. $\Pr_{\Gamma,x,t}[B_2|\neg B_1] \leq poly(k)2^{-k}$.

Proof. The proof is very similar to that of Claim 1, by observing the correspondence between $(x, F_{H(t)})$ and (t, H). Fix any x, t and any restriction Γ_1 of Γ that leaves only $F_{H(t)}(x)$ random. Again $G(*, t)$ is not determined yet, but it has no effect as it is not queried conditioned on $\neg B_1$. Then whether or not M^Γ queries $F_{H(t)}$ on x is completely determined by the input value, because all oracle answers that may matter have been fixed. The rest is similar. □

With these two claims, we have

$$\begin{aligned}\Pr_{\Gamma,x,t}\left[M^\Gamma(F_{H(t)}(x), H(t)) = x\right] &\leq \Pr_{\Gamma,x,t}[B_1] + \Pr_{\Gamma,x,t}[B_2|\neg B_1] \\ &\leq poly(k)2^{-k} + poly(k)2^{-k} \\ &\leq k^c 2^{-k},\end{aligned}$$

for some constant c. This completes the proof of Lemma 1. □

Let $\delta(k) = k^{c+2}2^{-k}$. Then we have

$$\Pr_{\Gamma}\left[\Pr_{x,t}\left[M^\Gamma(F_{H(t)}(x), H(t)) = x\right] > \delta(k)\right] \leq \frac{1}{k^2}.$$

As $\sum_k (1/k^2)$ converges, the Borel–Cantelli lemma (the Appendix) tells us that with probability one over Γ, $\Pr_{x,t}[M^\Gamma(F_{H(t)}(x), H(t)) = x]$ is at most $\delta(k)$ and thus negligible for sufficiently large k. There are only countably many machine M's, each of which can only succeed as an inverter over a measure zero of Γ, so we have the following.[8]

Lemma 2. *Relative to measure one of random* Γ, *injective trapdoor functions exist*.

[8] Like previous work on this subject, we only consider uniform adversaries. The analysis does not appear to work against non-uniform adversaries, as there are uncountably many of them.

3.2. *No OWP in* Γ

In this section we prove that no OWP exists relative to Γ. It was shown in [27] and [18] that no OWP exists relative to an oracle with a $\mathcal{PSPACE}$-complete problem and some random functions. We proceed by showing that the function G does not help us build OWP either. The idea is that it is unlikely to have a valid long input $(F(x, H(t)), t)$ to G without querying F at $(x, H(t))$ first. However, with x, which is the answer to the query $G(F(x, H(t)), t)$, one can eliminate this application of G.

We can think of the random oracle Γ as a family of oracles, with each oracle in the family being a possible instance of Γ.

Assume to the contrary that OWP exists relative to some positive measure of oracles in Γ. According to [27], this implies that for any constant $\delta > 0$, there exists a machine M that computes OWP on measure $1 - \delta$ of oracles in Γ. For some δ to be determined later, let Γ' denote this subset of oracles with measure $1 - \delta$ relative to which some M computes an OWP. We will show that for this M, there is another machine N which never queries G but still produces the same outputs for most inputs in most oracles. Then we will show that a good inverter exists for N, which can also invert M well, so M cannot be one-way.

Consider inputs from $\{0, 1\}^n$. Suppose M's running time is bounded above by n^{c_1}, for some constant c_1 independent of n. Choose $c = \max(c_1, 3)$. Let N be the machine that simulates M step by step, keeping track of the queries to F, but answers any query to G, say on (u, v), in the following way. Look for w with $u = F(w, H(v))$, by going through previous queries to F or searching through the space $\{0, 1\}^{|u|/3}$ if $|u| \leq 3c \log n$. If such w is found, N uses it as the answer for $G(u, v)$. Otherwise N assumes (u, v) is an invalid query to G and uses $\perp$ as the answer. This takes at most polynomial time.

We next argue that for any fixed input $x \in \{0, 1\}^n$, $N(x)$ agrees with $M(x)$ relative to most of the oracles in Γ.

Claim 3. *For every* $x \in \{0, 1\}^n$, $\Pr_\Gamma[N(x) \neq M(x)] \leq 1/n^2$.

Proof. Call a query (u, v) of M to G *lucky* if it is a valid query and u is longer than $3c \log n$ but not obtained by a previous query to F. Note that for any $x \in \{0, 1\}^n$, $N(x) \neq M(x)$ only if M ever makes a lucky query. M makes at most n^c queries to G. We next show that the ith query is the first lucky query with probability at most $n^c/(n^{3c} - i)$. Note that before the ith query, say on (u, v), we know less than i values in $\{0, 1\}^{|u|}$ that are not in the image of F. As F is length-tripling, u is in the image of F with probability at most $2^{|u|/3}/(2^{|u|} - i) \leq 2^{c \log n}/(2^{3c \log n} - i) = n^c/(n^{3c} - i)$. By the union bound, the probability we want to bound is at most $n^c \cdot n^c/(n^{3c} - n^c) = 1/(n^c - n^{-c}) \leq 1/n^2$, since $c \geq 3$. □

Recall that Γ' is a subset of Γ with measure $1 - \delta$. From the claim, for any $x \in \{0, 1\}^n$, $N(x) \neq M(x)$ for at most $1/((1 - \delta)n^2) \leq 2/n^2$ fraction of oracles in Γ', if $\delta \leq \frac{1}{2}$. It can be used to show that relative to most oracles M and N agree on most inputs. However, this is not enough because N may not be a permutation relative to most oracles, so we cannot apply the result of [27] directly to invert N. Some modification is needed. First, we need the following.

Lemma 3. *There are at most* $2/n$ *fraction of* n*-bit strings* y *such that* $N^{-1}(y) \neq M^{-1}(y)$ *relative to at least* $2/n$ *fraction of oracles in* Γ'.

Proof. Consider the Boolean matrix A with rows indexed by $y \in \{0,1\}^n$ and columns indexed $\gamma \in \Gamma'$, such that $A_{y,\gamma} = 1$ if and only if $N^{-1}(y) \neq M^{-1}(y)$ relative to γ. For each $x \in \{0,1\}^n$, $N(x) \neq M(x)$ relative to at most $2/n^2$ fraction of oracles in $\gamma \in \Gamma'$. Any such pair of x and γ with $N(x) \neq M(x)$ contributes at most two 1's to the matrix A (in particular, to the entries $A_{N(x),\gamma}$ and $A_{M(x),\gamma}$). So the total fraction of 1's in A is at most $4/n^2$. By the pigeonhole principle, there are at most $2/n$ fraction of rows in A which have at least $2/n$ fraction of columns of 1's. □

For any y, $M^{-1}(y)$ is unique relative to any oracle in Γ' since it is a permutation. So by Lemma 3, there are at least $1 - 2/n$ fraction of n-bit strings y such that $N^{-1}(y)$ is unique relative to at least $1 - 2/n$ fraction of oracles in Γ', and hence for at least $1 - 2/n - \delta > 1 - \varepsilon$ fraction of oracles in Γ, for any constant $\varepsilon > \delta$ and sufficiently large n. Observe that based on [18], the proofs of Theorem 9.2 and 9.3 in [27] actually yield the following stronger statement.

Lemma 4. *Relative to an oracle* Ψ *that contains only a* $\mathcal{PSPACE}$*-complete problem and random functions, there is a constant* λ *such that for every oracle polynomial-time machine* N*, there exists an oracle polynomial-time machine* N' *with the following property. For any* $\varepsilon < \lambda$ *and for any* y*, if* $N^{-1}(y)$ *is unique for* $1 - \varepsilon$ *fraction of* Ψ*, then* $N'(y) = N^{-1}(y)$ *for* $1 - \sqrt{\varepsilon}$ *fraction of* Ψ.

Then the rest follows closely the proof of Theorem 9.4 in [27]. Let λ be the constant specified in Lemma 4, and choose δ, ε such that $\delta < \varepsilon < \lambda$ and $\varepsilon + \sqrt[4]{\varepsilon} < 1$. Note that N never queries the function G in Γ. It only queries the $\mathcal{PSPACE}$-complete problem and the random functions, so Lemma 4 can be applied. As a result, for any n and for at least $1 - 2/n$ fraction of n-bit string y, an oracle PPTM can find $N^{-1}(y)$, which is $M^{-1}(y)$,[9] relative to at least $1 - \sqrt{\varepsilon}$ fraction of oracles in Γ. We need the following, which is a variant of the pigeonhole principle.

Claim 4. *Suppose* B *is a Boolean matrix in which there are at least* $1 - 2/n$ *fraction of the rows that have at least* $1 - \sqrt{\varepsilon}$ *fraction of* 1*'s. Then there are at least* $1 - \sqrt[4]{\varepsilon}$ *fraction of the columns that have at least* $1 - 2/n - \sqrt[4]{\varepsilon}$ *fraction of* 1*'s.*

Proof. It is not hard to see that there are totally at most $2/n + \sqrt{\varepsilon}$ fraction of 0's in B. Then the number of columns that have more than $2/n + \sqrt[4]{\varepsilon}$ fraction of 0's is at most

$$\frac{2/n + \sqrt{\varepsilon}}{2/n + \sqrt[4]{\varepsilon}} \leq \frac{\sqrt{\varepsilon}}{\sqrt[4]{\varepsilon}} = \sqrt[4]{\varepsilon}.$$

Thus, at least $1 - \sqrt[4]{\varepsilon}$ fraction of columns have at least $1 - 2/n - \sqrt[4]{\varepsilon}$ fraction of 1's. □

[9] Recall that for these y and Γ such that we can find $N^{-1}(y)$, we have $N^{-1}(y) = M^{-1}(y)$.

From the claim, there are at least $1 - \sqrt[4]{\varepsilon}$ fraction of oracles in Γ relative to which we can compute $M^{-1}(y)$ for at least $1 - 2/n - \sqrt[4]{\varepsilon}$ fraction of n-bit strings y for infinitely many n. As $\sqrt[4]{\varepsilon} < 1 - \varepsilon < 1 - \delta$, M is one-way relative to less than $1 - \delta$ fraction of oracles in Γ, a contradiction. So our assumption in the beginning that OWP exists relative to some positive measure of oracles in Γ must be false. Thus, with probability one over Γ, no one-way permutation exists relative to Γ. Together with Lemma 2, we have the following theorem.

Theorem 5. *There is no black-box reduction from OWP to (injective) TDF.*

4. PIR Does Not Imply OWP

In this section we construct an oracle Φ relative to which PIR exists but OWP does not. Similarly to Section 3 we add a special function G to an oracle with a $\mathcal{PSPACE}$-complete problem and some random functions. Formally, the oracle Φ consists of the following:

- A $\mathcal{PSPACE}$-complete oracle.
- A length-tripling random function $F(\cdot, \cdot)$.
- A random function $T\colon \{0,1\}^* \to \{0,1\}$.
- A family of random functions $H = \{H_k\colon \{0,1\}^* \to \{0,1\}^k | k \in \mathbb{N}\}$.
- A family of functions $G = \{G_k | k \in \mathbb{N}\}$ defined as follows:

$$\forall (u, v) \in \{0,1\}^* \times \{0,1\}^*,$$

$$G_k(u, v) = \begin{cases} u[s] \oplus T(H_k(u), t) & \text{if} \quad \exists s, t\colon v = F(s, t), \\ \perp & \text{otherwise.} \end{cases}$$

The idea behind this design is the following. In PIR, User shall use F to encrypt her index i as $F(i, m)$ for a randomly chosen M, and Server shall call G with $F(i, m)$ and his database x to get an encryption of $x[i]$, which can only be decrypted by User with the knowledge of m. The formal description of the protocol will be given shortly. As in the previous section, we will show that the function G is not useful for a one-party primitive, and thus not useful for building OWP.

Although the oracle Φ is designed to enable PIR, we stress that the definition of Φ does not depend on any instance of PIR. In Φ the functions F, T, H are random, and the function G is completely determined by F, T, H. When we want to carry out a particular PIR instance, the oracle functions will then be queried at some particular places. For example, with database x and index i, G will be queried at $(x, F(i, m))$ for a random m.

Note that G is a family of functions, but later when we refer to it, we usually mean some $G_k \in G$, and similarly for H. G is well defined if F is injective, which is not an issue as with probability one, it is so for sufficiently long inputs, and we can make G output 0 on those short inputs. Call a query (u, v) to G *valid* if $G(u, v) \neq \perp$.

4.1. *PIR in* Φ

The following is a 2-pass PIR using the oracle Φ, where Server has $x \in \{0,1\}^n$ and User has $i \in [n]$. Let k be the security parameter. For this parameter, we let H denote H_k and

let G denote G_k:

Server		User
	$\xleftarrow{\alpha}$	$\alpha = F(i, m)$, for $m \in_R U_k$
$\beta_1 = G(x, \alpha), \beta_2 = H(x)$	$\xrightarrow{\beta_1, \beta_2}$	$x[i] = \beta_1 \oplus T(\beta_2, m)$

The idea is the following. User needs to send her index i to Server in some way in order to obtain the bit $x[i]$. As User does not want Server to learn her index i, she would like to have it encrypted. So User chooses a random private key m and uses the random function F to encrypt i as $F(i, m)$. Server receives $F(i, m)$ but has no idea about i. How can Server send information about $x[i]$ to User without explicitly knowing the index i? The function G does the magical work, which takes any x together with $F(i, m)$ and returns the bit

$$G(x, F(i, m)) = x[i] \oplus T(H(x), m),$$

an encryption of $x[i]$. We want $x[i]$ encrypted, since otherwise Server may recover i by calling G using several different x's (User's security will be formally proved later). On the other hand, User has the key m, so after receiving $G(x, F(i, m))$ and $H(x)$, she can query $T(H(x), m)$ and derive

$$x[i] = G(x, F(i, m)) \oplus T(H(x), m).$$

The total number of bits sent by Server to User is

$$|\beta_1| + |\beta_2| = 1 + k,$$

which is okay when $n > 1 + k$.

It remains to prove User's security. Note that Server cannot affect what User would send, so whether Server is malicious or not makes no difference on User's security. If Server never queries the function G, the proof is standard as the rest of the oracle consists of merely random functions and a $\mathcal{PSPACE}$-complete function. The idea is that unless Server can guess User's private key m correctly, queries to G are unlikely to provide useful information. To see this, assume Server does not know m. The function H serves as a random hash and it is unlikely for Server to find distinct x', x'' such that $H(x') = H(x'')$ due to the large image of $H(\cdot)$, for sufficiently large k. Then for a query $G(x', F(i, m))$, the answer $x'[i] \oplus T(H(x'), m)$ is likely to look random as $T(H(x'), m)$ is likely so, and such a query is unlikely to be useful. That is, unless G is queried at (x', α) and (x'', α) for distinct x', x'' such that $H(x') = H(x'')$, G looks like a random function too.

Formally, we show that Server cannot distinguish the messages from User having indices i and j respectively. Consider any distinguisher M. Assume for the moment that M is a deterministic circuit family. Let $\delta = 2^{-k/4}$. For any $i, j \in [n]$, define $\triangle_m^{i,j} = M^{\Phi}(F(i, m)) - M^{\Phi}(F(j, m))$, which is a random variable of Φ. Then

$$\operatorname*{Ex}_m[\triangle_m^{i,j}] = \Pr_m[M^{\Phi}(F(i, m)) = 1] - \Pr_m[M^{\Phi}(F(j, m)) = 1].$$

We want to bound the probability

$$\Pr_{\Phi}\left[\exists i, j\colon \left|\operatorname{Ex}_{m}[\triangle_m^{i,j}]\right| > \delta\right],$$

which by the union bound is at most

$$\begin{aligned}\sum_{i,j}\Pr_{\Phi}\left[\left|\operatorname{Ex}_{m}[\triangle_m^{i,j}]\right| > \delta\right] &= \sum_{i,j}\Pr_{\Phi}\left[\left(\operatorname{Ex}_{m}[\triangle_m^{i,j}]\right)^2 > \delta^2\right] \\ &\le \sum_{i,j}\operatorname{Ex}_{\Phi}\left[\left(\operatorname{Ex}_{m}[\triangle_m^{i,j}]\right)^2\right]/\delta^2,\end{aligned}$$

where the last line is due to the Markov inequality. Then we need the following lemma.

Lemma 6. $\forall i, j, \operatorname{Ex}_{\Phi}[(\operatorname{Ex}_m[\triangle_m^{i,j}])^2] \le poly(n)2^{-k}$.

Proof. Fix any $i, j \in [n]$. Write $\triangle_m$ for $\triangle_m^{i,j}$ and note that $\operatorname{Ex}_{\Phi}[(\operatorname{Ex}_m[\triangle_m])^2] = \operatorname{Ex}_{\Phi,m,m'}[\triangle_m \triangle_{m'}]$. Define the following probability events, with Φ, m, m' chosen randomly:

- B_1: On input $F(i,m)$ or $F(j,m)$, M^{Φ} queries $F(*,m)$ or $T(*,m)$, or queries $G(x',*)$ and $G(x'',*)$ with $x' \neq x''$ and $H(x') = H(x'')$.
- B_2: On input $F(i,m')$ or $F(j,m')$, M^{Φ} queries $F(*,m)$, $T(*,m)$, or $G(*, F(*,m))$.

These are the bad events, which happen with probability at most $poly(n)2^{-k}$. Next we show that the expectation of $\triangle_m \triangle_{m'}$ is small if neither bad event happens.

Consider any restriction Φ_0 of Φ with $F(*,m)$ and $T(*,m)$ still random but the rest fixed. M's computation is determined by the input and the answers to its oracle queries.

Assume the condition $\neg B_1$. We will show that the computation of $M^{\Phi_0}(F(i,m))$ and the computation of $M^{\Phi_0}(F(j,m))$ have the same distribution. We do this by establishing a one-to-one correspondence between all the possible computations of $M^{\Phi_0}(F(i,m))$ and all the possible computations of $M^{\Phi_0}(F(j,m))$. Initially, their inputs $F(i,m)$ and $F(j,m)$ have the same distribution. Consider the branch of $M^{\Phi_0}(F(i,m))$ and the branch of $M^{\Phi_0}(F(j,m))$ that start with the same input value (i.e., $F(i,m) = F(j,m)$) and then get the same oracle answers, up to some query. Their first difference can only happen when M queries $G(x', F(i,m))$ and $G(x', F(j,m))$, respectively, for some x'. The oracle answers $x'[i] \oplus T(H(x'),m)$ and $x'[j] \oplus T(H(x'),m)$ have the same distribution as $T(H(x'),m)$ remains free up to this point, because under the condition of $\neg B_1$, no $G(x'',*)$ with $x' \neq x''$ and $H(x') = H(x'')$, nor $T(*,m)$ is queried. Again, we can choose the branch of $M^{\Phi_0}(F(i,m))$ and the branch of $M^{\Phi_0}(F(j,m))$ that have the same value for $G(x', F(i,m))$ and $G(x', F(j,m))$, and then repeat the same process. We can show by induction that for each possible computation of $M^{\Phi_0}(F(i,m))$, there is a corresponding computation of $M^{\Phi_0}(F(j,m))$, and vice versa. This implies that the computations of $M^{\Phi_0}(F(i,m))$ and $M^{\Phi_0}(F(j,m))$ have the same distribution. Thus, given $\neg B_1$, $\operatorname{Ex}_{\Phi_0}[M^{\Phi_0}(F(i,m))] = \operatorname{Ex}_{\Phi_0}[M^{\Phi_0}(F(j,m))]$ and $\operatorname{Ex}_{\Phi_0}[\triangle_m] = 0$.

Next, consider any $m' \neq m$. Given $\neg B_2$, $\triangle_{m'}$ is fixed under Φ_0 as it does not depend on $F(*,m)$ or $T(*,m)$. Let $B = B_1 \cup B_2$. Then given $\neg B$, $\operatorname{Ex}_{\Phi_0}[\triangle_m \triangle_{m'}] =$

$\mathrm{Ex}_{\Phi_0}[\triangle_m]\triangle_{m'} = 0$ for any restriction Φ_0. Thus, we have $\mathrm{Ex}_{\Phi,m,m'}[\triangle_m\triangle_{m'}|\neg B] \leq \Pr_{m,m'}[m = m'] = 2^{-k}$ and

$$\begin{aligned}\mathop{\mathrm{Ex}}_{\Phi,m,m'}[\triangle_m\triangle_{m'}] &\leq \Pr_{\Phi,m,m'}[B] + \mathop{\mathrm{Ex}}_{\Phi,m,m'}[\triangle_m\triangle_{m'}|\neg B] \\ &\leq poly(n)2^{-k}.\end{aligned}$$

□

From the lemma, we have $\Pr_\Phi[\exists i, j\,|\,\mathrm{Ex}_m[\triangle_m^{i,j}]| > \delta] \leq poly(n)2^{-k/2}$. As $\sum_n poly(n)2^{-k/2}$ converges for, say, $k = \Omega(\log^2 n)$, the Borel–Cantelli lemma tells us that with probability one over Φ, $|\mathrm{Ex}_m[\triangle_m^{i,j}]| \leq \delta$ for any $i, j \in [n]$ for sufficiently large n and sufficiently large k. Note that the analysis so far works for any deterministic polynomial-sized circuit family, so it works for any (uniform) probabilistic polynomial-time machine, too. There are only countably many (uniform) machines as distinguishers, each of which succeeds with measure zero over Φ. Then with probability one over Φ, Server cannot learn User's index for sufficiently large n. So we have the following.

Lemma 7. *Our protocol is a PIR relative to measure one of* Φ.

4.2. *No OWP in* Φ

The proof that no OWP exists in Φ is almost identical to the one in Section 3.2. Assume to the contrary that there is a PPTM M, with time bound n^c, that computes an OWP. We construct a PPTM N by simulating M and replacing any query to G at (u, v) by $\perp$ if v is longer than $3c \log n$ and not obtained from a previous query to F. If v is obtained from a previous query $F(s, t)$ or short enough to find the corresponding (s, t) by an exhaustive search, N replaces $G(u, v)$ by $u[s] \oplus T(H(u), t)$. Then, as in Section 3.2, N and M have the same output on most inputs, but N can be inverted on most inputs. It follows that M is not one-way, a contradiction. So we have the following.

Theorem 8. *There is no black-box reduction from OWP to PIR.*

Together with Theorem 5 and previous results, we have the following.

Corollary 9. *There is no black-box reduction from OWP to any of the basic primitives, including TDF, PKE, PIR, OT, KA, OWF, PRG, PRF, BC, and DS.*

Acknowledgments

We thank anonymous referees of Asiacrypt '02 and *Journal of Cryptology* for their valuable comments. The second author thanks Leonid Reyzin for helpful discussions.

Appendix. The Borel-Cantelli Lemma

We need the following lemma, known as the Borel–Cantelli lemma, which can be found in many textbooks on measure theory, such as [1]. It says that if the sum of the probabilities

of a sequence of events converges, then the probability that infinitely many of these events happen is zero. We give a proof here for completeness.

Lemma 10. *Let $B_1, B_2, \ldots$ be a sequence of probability events on the same probability space. Then $\sum_{n=1}^{\infty} \Pr[B_n] < \infty$ implies $\Pr[\bigwedge_{k=1}^{\infty} \bigvee_{n \geq k} B_n] = 0$.*

Proof. Let $B = \bigwedge_{k=1}^{\infty} \bigvee_{n \geq k} B_n$ and let $A_k = \bigvee_{n \geq k} B_n$. Note that $B = \bigwedge_{k=1}^{\infty} A_k$; so $\Pr[B] \leq \Pr[A_k]$ for all k. Suppose $\sum_{n=1}^{\infty} \Pr[B_n] < \infty$. Then for every $\varepsilon > 0$ there exists a $k_0 \in \mathbb{N}$ such that $\sum_{n \geq k_0} \Pr[B_n] < \varepsilon$, so we have

$$\Pr[B] \leq \Pr[A_{k_0}] = \Pr\left[\bigvee_{n \geq k_0} B_n\right] \leq \sum_{n \geq k_0} \Pr[B_n] < \varepsilon.$$

Since ε can be arbitrarily small, we must have $\Pr[B] = 0$. □

References

[1] Malcolm Adams and Victor Guillemin. *Measure Theory and Probability*. Wadsworth & Brooks/Cole Advanced Books and Software, Monterey, CA, 1986.

[2] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. One-way functions are essential for single-server private information retrieval. In *Proceedings of the* 31*st Annual ACM Symposium on Theory of Computing*, pages 89–98, 1999.

[3] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO* '98, volume 1462 of Lecture Notes in Computer Science, pages 283–298. Springer-Verlag, Berlin, 1998.

[4] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology—EUROCRYPT* '99, volume 1592 of Lecture Notes in Computer Science, pages 402–414. Springer-Verlag, Berlin, 1999.

[5] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of the* 36*th Annual ACM Symposium on Theory of Computing*, pages 41-50, 1995.

[6] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT* '00, volume 1807 of Lecture Notes in Computer Science, pages 122–138. Springer-Verlag, Berlin, 2000.

[7] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644-654, 1976.

[8] Marc Fischlin. On the impossibility of constructing non-interactive statistically-secret protocols from any trapdoor one-way function. In Bart Preneel, editor, *Topics in Cryptology—CT-RSA* '02, volume 2271 of Lecture Notes in Computer Science, pages 79–95. Springer-Verlag, Berlin, 2002.

[9] Rosario Gennaro, Yael Gertner, and Jonathan Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *Proceedings of the* 45*th Annual ACM Symposium on Theory of Computing*, pages 417–425, 2003.

[10] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the* 41*st Annual IEEE Symposium on Foundations of Computer Science*, pages 305–313, 2000.

[11] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings of the* 41*st Annual IEEE Symposium on Foundations of Computer Science*, pages 325–335, 2000.

[12] Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *Proceedings of the* 42*nd Annual IEEE Symposium on Foundations of Computer Science*, pages 126–135, 2001.

[13] Oded Goldreich. *Foundations of Cryptography*: *Basic Tools*. Cambridge University Press, Cambridge, 2001.

[14] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *Proceedings of the* 27*th Annual IEEE Symposium on Foundations of Computer Science*, pages 174–187, 1986.

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

[16] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proceedings of the* 30*th Annual IEEE Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[17] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the* 21*st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.

[18] Jeff Kahn, Michael E. Saks, and Cliff Smyth. A dual version of Reimer's inequality and a proof of Rudich's conjecture. In *Proceedings of the* 15*th Annual IEEE Conference on Computational Complexity*, pages 98–103, 2000.

[19] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the* 20*th Annual ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[20] Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *Proceedings of the* 40*th Annual IEEE Symposium on Foundations of Computer Science*, pages 535–542, 1999.

[21] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the* 38*th Annual IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[22] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT* '00, volume 1807 of Lecture Notes in Computer Science, pages 104–121. Springer-Verlag, Berlin, 2000.

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

[24] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *Proceedings of the* 1*st Theory of Cryptography Conference*, volume 2951 of Lecture Notes in Computer Science, pages 1–20. Springer-Verlag, Berlin, 2004.

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

[26] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the* 22*nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[27] Steven Rudich. Limits on the provable consequences of one-way functions. Ph.D. thesis, University of California, Berkeley, CA, 1988.

[28] Steven Rudich. The use of interaction in public cryptosystems (extended abstract). In Joan Feigenbaum, editor, *Advances in Cryptology—CRYPTO* '91, volume 576 of Lecture Notes in Computer Science, pages 242–251. Springer-Verlag, Berlin, 1991.

[29] Daniel R. Simon. Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology—EUROCRYPT* '98, volume 1403 of Lecture Notes in Computer Science, pages 334–345. Springer-Verlag, Berlin, 1998.

[30] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the* 23*rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.