On the Impossibilities of Basing One-Way Permutations on Central Cryptographic Primitives

Yan-Cheng Chang[1], Chun-Yun Hsiao[1], and Chi-Jen Lu[2]

[1] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. {r88023,r88067}@csie.ntu.edu.tw
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan.
cjlu@iis.sinica.edu.tw

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 seem 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 work, 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 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" [12], 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, permutation seems to be something orthogonal to those additional properties on top of one-wayness. Like previous non-reducibility results [12, 21, 23, 17, 7, 9, 8, 6], our proofs follow the oracle separation paradigm of Impagliazzo and Rudich [12].

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 securities 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 be 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 will also restrict ourselves to the classical world of cryptography, and leave the questions in quantum cryptography for future studies.

According to [7], such basic cryptographic primitives can be roughly divided into two categories: private cryptography and public cryptography.[3] Private cryptography is represented by private-key encryption, and includes one-way permutation (OWP), one-way function (OWF), pseudo-random generator (PRG), pseudo-random functions (PRF), bit commitment (BC), and digital signature (DS). Public cryptography is represented by public-key encryption (PKE), and includes trapdoor permutations (TDP), trapdoor functions (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 [11, 19, 1, 2, 7]. 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 clear what it means that one primitive Q does not imply the other primitive P, or equivalently P can not 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 based on Q that simply ignores Q. Impagliazzo and Rudich [12] introduced a restricted but important subclass of reductions called *black-box reductions*. 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, 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. 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 [12] showed that no such black-box reduction exists from KA to OWP. As every primitive

[3] We want to remark that 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. [23, 6]).

in public cryptography implies KA [1, 4, 7], this provides a strong evidence that primitives in public cryptography requires strictly more than one-wayness. Since then, more and more separations between cryptographic primitives have been established following this paradigm [21, 23, 17, 7, 9, 8, 6].

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 [7] and thus PIR [4]. 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 [20, 16], 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 [12], so on top of one-wayness, one can not 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. We will show that neither TDF nor PIR implies OWP, so the answer to that question is actually no!

$$\begin{tikzcd}
 & \text{TDF} \arrow[r] & \text{PKE} \arrow[dr] & & \text{OWP} \arrow[dl, "/" marking] \arrow[dd, shift left] \\
\text{TDP} \arrow[ur] \arrow[dr] & & & \text{KA} \arrow[dr] & \\
 & \text{PIR} \arrow[r] & \text{OT} \arrow[ur] & & \text{OWF, PRG, PRF, BC, DS} \arrow[uu, shift left, "/" marking]
\end{tikzcd}$$

Fig. 1. Relationships between OWP and other cryptographic primitives

We first construct an oracle, relative to which an injective trapdoor function (iTDF) exists whereas OWP does not. As iTDF implies PKE [24][4] and PKE (two-pass KA) implies KA, we 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 [4], 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 [15]. 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, which make almost all of conceivable cryptographic tasks possible, are still unable to yield OWP. Our results suggest that permutation is really a special property that is orthogonal to other additional properties required in cryptography. Furthermore, the reducibility from OWP to each primitive was already known before, so now all the relationships, with respect to black-box reductions, between one-way permutations and those basic

[4] In fact, TDF with polynomial pre-image size suffices to imply PKE [1].

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 [12].

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 [12] and Gertner *et al.* [7]. It is known that a random function is one-way with high probability, even relative to a $\mathcal{PSPACE}$-complete function [12]. 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 relative to such an oracle [20, 16]. 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 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 primitive involved in this paper. Then in Section 3 and 4, we prove that no black-box reductions exist from OWP to iTDF 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 i-th 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 distribution S, we write $s \in S$ 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 function is called negligible if it vanishes faster than any inverse polynomial. We say that two distributions X and Y over $\{0,1\}^k$ cannot be distinguished if for any PPTM M,

$$\left| \Pr_{x \in X}[M(x) = 1] - \Pr_{y \in Y}[M(y) = 1] \right| \leq \delta(k),$$

for some negligible function $\delta(k)$. We say a function is easy to compute if it is computable in polynomial time. We say that a function f is hard to invert if for any PPTM M,

$$\Pr_{x \in U_k}[f(M(f(x))) = f(x)] \leq \delta(k),$$

for some negligible function $\delta(k)$.

In the following, we give brief definitions of the cryptographic primitives studied in this paper. More formal treatment can be found in standard textbooks or the original papers. The most fundamental primitive is *one-way function*, which is essential to all cryptographic primitives.

Definition 1. *A one-way function (OWF) is a function that is easy to compute but hard to invert.*

From one-way functions, we define primitives with additional properties. A one-way permutation is a one-way function that is itself a permutation.

Definition 2. *A one-way permutation (OWP) is a one-way function f with the additional requirement that for every $k \in \mathbb{N}$, f maps $\{0,1\}^k$ to $\{0,1\}^k$ in a one-to-one and onto way.*

Trapdoor functions are one-way functions which, when given some additional *trapdoor* information, are easy to invert.

Definition 3. *A collection of trapdoor functions (TDF) is a collection of function families $\mathcal{F} = \{\mathcal{F}_k | k \in \mathbb{N}\}$ satisfying the following properties.*

- *There is a PPTM I, that on input 1^k outputs a pair (f, t), where f is (an index of) a function in $\mathcal{F}_k$ and t is a string called the* trapdoor *for f.*
- *Each f is easy to compute, and when the trapdoor t is given, f is also easy to invert.*
- *For a random $(f, t) \in I(1^k)$, f is hard to invert without knowing the trapdoor t.*

Next, we describe private information retrieval, which was introduced by Chor *et al.* [3]. This is a two-party protocol, where User wants to secretly learn some bit of Server's database, conditioned on a non-trivial upper bound on Server's communication complexity.

Definition 4. *Private information retrieval (PIR) is a protocol involving two parties. Server has a database $x \in \{0,1\}^n$ while User has an index $i \in [n]$ and wants to learn the bit $x[i]$ in the following way.*

- *Server sends less than n bits to User.*
- *User keeps the index secret in the sense that Server cannot distinguish the distributions of messages sent from User when the indices are i and i' respectively, for any $i' \neq i$.*[5]

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 [20, 16] that

[5] The security parameter here can be set to $k = poly(n)$.

no OWP exists relative to an oracle with a $\mathcal{PSPACE}$-complete problem and some random functions. 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 $F(\cdot, \cdot)$.
- A length-tripling random function $H(\cdot)$.
- A function G defined as follows.

$$\forall(u, v): \quad G(u, v) = \begin{cases} w \text{ if } \exists w : u = F(w, H(v)), \\ \perp \text{ otherwise.} \end{cases}$$

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 w.l.o.g. that both F and H are injective, because one can show that length-tripling functions are injective on sufficiently long inputs with measure one .

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 [12, 7].

3.1 TDF in Γ

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

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

It remains to show that F_t is hard to invert without knowing the trapdoor t.

Consider any oracle PPTM M as an inverter. Without the oracle G, F_t is a random function and is likely to be one-way, by a standard argument (e.g. [20]). The idea is that unless M^Γ can guess the trapdoor t correctly, G is unlikely to provide useful information for inverting F_t. Formally, for a negligible function $\delta(k)$, we want to upper-bound the probability

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

which by Markov inequality is at most

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

We need the following lemma.

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

Proof. Define the following probability event:

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

We first show that this bad event is unlikely to happen.

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

Proof. Note that whether or not M^{Γ} queries H on t or G on $(*, t)$ does *not* depend on either $H(t)$ or $G(*, t)$. Instead, it is completely determined by the input together with those $H(t')$ and $G(*, t')$ for every $t' \neq t$. Fix any x, t and any restriction Γ_0 of Γ that leaves only $H(t)$ random. Note that $G(*, t)$ is not fixed yet as it depends on $H(t)$, but it has no effect on B_1. Then whether or not B_1 happens depends only on the input, because all oracle answers that may matter have been fixed. Therefore,

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

where the last inequality is because M makes at most $\mathit{poly}(k)$ 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 w.o.l.g. that M^{Γ} always uses its output to query F_t at the final step before it stops. This does not affect its inverting probability, which is bounded above by the probability of the following event:

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

So it remains to prove the following claim.

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

Proof. The proof is very similar to that of Claim 1, by observing the correspondence between (x, F_t) and (t, H). Fix any x, t and any restriction $\mathit{\Gamma}_1$ of $\mathit{\Gamma}$ that leaves only $F_t(x)$ 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 $M^{\mathit{\Gamma}}$ queries F_t on x is completely determined by the input, because all oracle answers that may matter have been fixed. The rest is similar. □

With these two claims, we have

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

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

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

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

Now as $\sum_k \frac{1}{k^2}$ converges, the *Borel-Cantelli Lemma* tells us that with probability one over $\mathit{\Gamma}$, $\Pr_{x,t}[M^{\mathit{\Gamma}}(F_t(x), H(t)) = x]$ is negligible for sufficiently large k. There are only countably many machines M's, each of which can only succeed as an inverter over a measure zero of $\mathit{\Gamma}$, so we have the following.[6]

Lemma 2. *Relative to measure one of random $\mathit{\Gamma}$, injective trapdoor functions exist.*

3.2 No OWP in $\mathit{\Gamma}$

In this section we show that no OWP exists relative to $\mathit{\Gamma}$. It was shown in [20, 16] 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 $(F(x, H(t)), t)$ without querying F at $(x, H(t))$ first. But with x, the answer to the query $G(F(x, H(t)), t)$, one can eliminate this application of G. We can see the random oracle $\mathit{\Gamma}$ as a family of oracles, with each oracle in the family being a possible instance of $\mathit{\Gamma}$.

Assume for the contrary that OWP exists relative to $\mathit{\Gamma}$. According to [20], this implies that for any constant $\delta > 0$, there exists a machine M that computes

[6] 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.

OWP on measure $1-\delta$ of oracles in Γ. Let Γ' denote this subset of oracles relative to which M is a 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. Then we will show that a good inverter exists for N, which can also inverts M well, so M cannot be one-way.

Consider inputs from $\{0,1\}^n$. Suppose M's running time is bounded by n^c, for some constant $c \geq 2$ independent of n. For this constant c, let N be the machine that simulates M step by step, keeps track of the queries to F, and answers any query to G, say on (u, v), by the following. Look for w with $u = F(w, H(v))$, by going through previous queries to F or searching the space $\{0,1\}^{|u|/3}$ if $|u| \leq 3c \log n$. If such w is found, N answers $G(u, v)$ with it. Otherwise N assumes (u, v) an invalid query and answers it with $\perp$. This takes at most polynomial time.

For any input $x \in \{0,1\}^n$, $N(x) \neq M(x)$ only if M every queries G on some valid (u, v) with u longer than $3c \log n$ but not obtained by previous queries to F. Then for any fixed random choice of M, $N(x) \neq M(x)$ for at most $n^c 2^{c \log n}/2^{3c \log n} = 1/n^c \leq 1/n^2$ of oracles in Γ, and hence for at most $1/((1-\delta)n^2) \leq 2/n^2$ of oracles in Γ', for $\delta \leq 1/2$. Although we can then show that relative to most oracles M and N agree on most inputs, but N may not be a permutation relative to most oracles. So we can not apply [20] directly to invert N, and some modification is needed. First, we can have the following.

Lemma 3. *There are less than $2/n$ fraction of n-bit strings y such that $N^{-1}(y) \neq M^{-1}(y)$ for more than $2/n$ 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 $A_{y,\gamma} = 1$ iff $N^{-1}(y) \neq M^{-1}(y)$ relative to γ. For each $x \in \{0,1\}^n$, $N(x) \neq M(x)$ for at most $2/n^2$ of oracles in Γ', and this contributes at most $2^{-n}4/n^2$ fraction of 1's to A. As there are 2^n different x's, the total fraction of 1's in A is at most $4/n^2$. By the pigeon-hole principle, less than $2/n$ of rows in A have more than $2/n$ 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 more than $1 - 2/n$ fraction of n-bit strings y such that $N^{-1}(y)$ is unique for more than $1 - 2/n$ of oracles in Γ', and hence for more than $1 - 2/n - \delta > 1 - \epsilon$ of oracles in Γ, for any constant $\epsilon > \delta$ and sufficiently large n. Observe that based on [16], the proofs of Theorem 9.2 and 9.3 in [20] actually yield the following stronger statement.

Lemma 4. *Assume $\mathcal{P} = \mathcal{NP}$. There is a constant λ such that for every machine N, there exists a machine N' with the following property. For any $\epsilon < \lambda$ and for any y, if $N^{-1}(y)$ is unique for $1 - \epsilon$ of random oracles, then $N'(y) = N^{-1}(y)$ for $1 - \sqrt{\epsilon}$ of random oracles.*

Then the rest follows closely the proof of Theorem 9.4 in [20]. Choose $\delta < \lambda$ such that there exists ϵ with $\delta < \epsilon < \lambda$ and $\epsilon + \sqrt[4]{\epsilon} < 1$. We have $\mathcal{P} = \mathcal{NP}$ relative to Γ, so for any n, there are more than $1 - 2/n$ of n-bit string y such that we can find $N^{-1}(y) = M^{-1}(y)$ for more than $1 - \sqrt{\epsilon}$ of oracles in Γ. By the pigeon-hole principle, there are more than $1 - \sqrt[4]{\epsilon}$ of oracles in Γ relative to which we can compute $M^{-1}(y)$ for more than $1 - 2/n - \sqrt[4]{\epsilon}$ fraction of n-bit strings y for infinitely many n. That is, M is one-way relative to less than $\sqrt[4]{\epsilon} < 1 - \epsilon < 1 - \delta$ fraction of oracles in Γ, a contradiction. Thus, with probability one over Γ, no one-way permutation exists relative to Γ. Together with Lemma 2, we have the following theorem.

Theorem 1. *There is no black-box reduction from OWP to iTDF.*

4 PIR does not imply OWP

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

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

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

The idea behind this design is the following. In PIR, User shall use F to encrypt her index i as $F(i,m)$, and Server shall call G with $F(i,m)$ and his database x to get

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

an encryption of $x[i]$, which can only be decrypted by User. As in the previous section, we will next 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 outputs 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.

$$\begin{array}{rcl} \textbf{Server} & & \textbf{User} \\ & \xleftarrow{\alpha} & \alpha = F(i,m), \text{ for } m \in U_k \\ \beta_1 = G(x,\alpha), \beta_2 = H(x) & \xrightarrow{\beta_1,\, \beta_2} & x[i] = \beta_1 \oplus T(\beta_2, m) \end{array}$$

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 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. 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 such x', 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 machine M as a distinguisher.

Let $\delta(n) = 2^{-k/4}$, a negligible function in n. 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

$$\begin{aligned}\Pr_{\Phi}\left[\exists i,j : \left|\operatorname*{Ex}_m[\triangle_m^{i,j}]\right| > \delta(n)\right] &\le \sum_{i,j}\Pr_{\Phi}\left[\left|\operatorname*{Ex}_m[\triangle_m^{i,j}]\right| > \delta(n)\right] \\ &\le \sum_{i,j}\operatorname*{Ex}_{\Phi}\left[\left(\operatorname*{Ex}_m[\triangle_m^{i,j}]\right)^2\right]/\delta^2(n).\end{aligned}$$

So we need the following lemma.

Lemma 5. *$\forall i, j$, $\mathrm{Ex}_{\Phi}[\mathrm{Ex}_m[(\triangle_m^{i,j}])^2] \le poly(n)2^{-k}$.*

Proof. Fix any $i, j \in [n]$. Write $\triangle_m$ for $\triangle_m^{i,j}$ and note that $\mathrm{Ex}_{\Phi}[(\mathrm{Ex}_m[\triangle_m])^2] = \mathrm{Ex}_{\Phi,m,m'}[\triangle_m\triangle_{m'}]$. 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 on $(*, m)$ or knows distinct x', x'' with $H(x') = H(x'')$.
- B_2: On input $F(i,m')$ or $F(j,m')$, M^{Φ} queries either $F(*,m)$, $T(*,m)$, 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 $\neg B_1$. Consider any possible run of $M^{\Phi_0}(F(i,m))$ and $M^{\Phi_0}(F(j,m))$, starting with $F(i,m) = F(j,m)$ and then getting same oracle answers, up to some query. Assume that now for some x', $G(x', F(i,m))$ and $G(x', F(j,m))$ are queried respectively, as other oracle answers are fixed under Φ_0. The 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. By induction, $M^{\Phi_0}(F(i,m))$ and $M^{\Phi_0}(F(j,m))$ have the same distribution of computations. So given $\neg B_1$, $\mathrm{Ex}_{\Phi_0}[M^{\Phi_0}(F(i,m))] = \mathrm{Ex}_{\Phi_0}[M^{\Phi_0}(F(j,m))]$ and $\mathrm{Ex}_{\Phi_0}[\triangle_m] = 0$.

Consider any $m' \ne m$. 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 $\neg B$, $\mathrm{Ex}_{\Phi_0}[\triangle_m\triangle_{m'}] = \mathrm{Ex}_{\Phi_0}[\triangle_m]\triangle_{m'} = 0$ for any restriction Φ_0. Thus,

$$\begin{aligned}\operatorname*{Ex}_{\Phi,m,m'}[\triangle_m\triangle_{m'}|\neg B] &\le \Pr_{m,m'}[m = m'] \\ &= 2^{-k},\end{aligned}$$

and we have

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

□

Then, $\Pr_\Phi[\exists i,j|\,\mathrm{Ex}_m[\triangle_m^{i,j}]| > \delta(n)] \leq poly(n)2^{-k/2}$. 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, $|\,\mathrm{Ex}_m[\triangle_m^{i,j}]| \leq \delta(n)$ for any $i,j \in [n]$ for sufficiently large n. There are only countably many machines M's 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 6. *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 the contrary that there is a PPTM M, with time bound n^c, that computes a OWP. We construct 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 s,t by exhaustive search, N replace $G(u,v)$ by $u[s] \oplus T(H(u),t)$. Then, as in Section 3.2, N has the same output as M does 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 2. *There is no black-box reduction from OWP to PIR.*

Together with Theorem 1 and previous results, we have the following.

Corollary 1. *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.*

References

1. 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, 1998.
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 31st Annual ACM Symposium on Theory of Computing*, pages 89-98, 1999.
3. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 41-50, 1995.

4. 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, 2000.
5. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644-654, 1976.
6. 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, 2002.
7. Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 325-335, 2000.
8. Yael Gertner, Tal Malkin, and Omer Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 126-135, 2001.
9. Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 305-313, 2000.
10. 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.
11. Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 230-235, 1989.
12. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44-61, 1989.
13. Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 20-31, 1988.
14. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 364-373, 1997.
15. 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, 2000.
16. 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 15th Annual IEEE Conference on Computational Complexity*, pages 98-103, 2000.
17. Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 535-542, 1999.
18. Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151-158, 1991.
19. John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387-394, 1990.
20. Steven Rudich. Limits on the provable consequences of one-way functions. *Ph.D. thesis*, U.C. Berkeley, 1988.

21. 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, 1991.
22. 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.
23. 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, 1998.
24. Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80-91, 1982.