The Relationship between Public Key Encryption and Oblivious Transfer

Preliminary Version

Yael Gertner*
University of Pennsylvania
ygertner@saul.cis.upenn.edu

Sampath Kannan*
University of Pennsylvania
kannan@central.cis.upenn.edu

Tal Malkin
AT&T Labs — Research
tal@research.att.com

Omer Reingold
AT&T Labs — Research
omer@research.att.com

Mahesh Viswanathan*
University of Pennsylvania
maheshv@saul.cis.upenn.edu

November 26, 2000

Abstract

In this paper we study the relationships among some of the most fundamental primitives and protocols in cryptography: public-key encryption (i.e. trapdoor predicates), oblivious transfer (which is equivalent to general secure multi-party computation), key agreement and trapdoor permutations. Our main results show that public-key encryption and oblivious transfer are *incomparable* under black-box reductions. These separations are tightly matched by our positive results where a restricted (strong) version of one primitive *does imply* the other primitive. We also show separations between oblivious transfer and key agreement. Finally, we conclude that neither oblivious transfer nor trapdoor predicates imply trapdoor permutations. Our techniques for showing negative results follow the oracle separations of Impagliazzo and Rudich [IR89, Rud91].

[1]Supported by grants ONR N00014-97-0505 (MURI), NSF CCR 98-20885, and ARO DAAG55-98-1-0393

1 Introduction

The cryptographic research of the last few decades has proposed solutions to almost every "conceivable" cryptographic task (and to quite a few tasks that first seemed "inconceivable"). The correctness and security of these solutions was proven under a growing number of (unproven) computational assumptions put forth by the community. To some extent, this state of affairs is unavoidable: the security of "almost any" cryptographic protocol implies the existence of one-way functions[1][IL89] (which in particular implies $P \neq NP$). Therefore, for most cryptographic protocols, unconditional proofs of security seem well beyond the reach of complexity theory. Nevertheless, it might be possible that many of these assumptions are related or even equivalent to one another. Exploring the relationship between these assumptions, and determining those that are integral to cryptography, would thus greatly clarify our understanding of the cryptographic world, and is considered to be one of the most fundamental goals.

For some primitives, this goal has been met with much success. For example, private key encryption (Private KE), pseudo-random generators (PSRG), pseudo-random functions and permutations, bit commitment, and digital signatures (Sig), have all been shown to exist if and only if one-way functions exist [Yao82, GM84, GGM86, LR88, IL89, NY89, Rom90, Nao91, HILL99]. Thus, the existence of one-way functions is a powerful assumption that captures and unifies a large class of cryptographic primitives.

It is, therefore, natural to wonder if all cryptographic primitives are equivalent to one-way functions. However, this seems unlikely. For example, it is safe to assert that a construction of trapdoor permutations out of one-way functions, if at all possible, would require immense innovation: trapdoor permutations just seem to have much more structure than one-way functions. Yet, constructing a formal proof based on this intuition is non-trivial. In fact, it is not even clear what is the formal meaning to such a claim, given the common belief that both one-way functions and trapdoor permutations exist (hence a "reduction" can just ignore the one-way functions and build trapdoor permutations from scratch).

Indeed overcoming these challenges has proved to be very difficult. While we know of no techniques that prove the failure of *all* reductions, Impagliazzo and Rudich [IR89] gave a method for separating primitives under a restricted but important subclass of reductions, namely, *black-box reductions*. Informally, a black-box reduction of a primitive P to a primitive Q is a construction of P out of Q that ignores the internal structure of the implementation of Q. The exact definition of black-box reductions and the Impagliazzo and Rudich methodology for black-box separations are rather substle (see Section 2.2). In the introduction however we continue with a more intuitive treatment of these notions and identify black-box reductions with relativizing reductions (ones that work relative to any oracle). Hence, in order to prove that a black-box reduction of P to Q is impossible, it suffices to demonstrate an oracle relative to which the primitive Q exists whereas P does not. We stress that such black-box seperations have their limitations (as discussed below). Nevertheless, as noted by Impagliazzo and Rudich, almost all constructions in cryptography *are black box*. Therefore, if there are no black-box reductions of P to Q then a proof that the existence of Q implies the existence of P is most likely very difficult.

Using the methodology described above, Impagliazzo and Rudich [IR89] showed that not all cryptographic primitives are equivalent. Specifically, they show that there are no black-box reductions from one-way functions (OWF) to key agreement (KA). This result immediately implies separations between OWF (and other equivalent primitives) and several other primitives which are known to imply KA. In other words, the cryptographic primitives can be divided into two worlds: "private cryptography" consisting of OWF and all equivalent primitives, and "public cryptography" consisting of "harder" primitives such as KA, public key encryption (PKE), Oblivious Transfer (OT), secure function evaluation (SFE), and trapdoor permutations (TD Perm).[2] These implications are summarized in Figure 1.

An interesting problem to consider, therefore, is the relationship between the primitives in the public cryptography world (see e.g. [Gol97]). This is especially importent because PKE and OT are two of the most fundamental primitives in cryptography. Essentially all of cryptography is based on them: public-key encryption allows *parties that "just met"* to exchange secret messages in the presence of an eavesdropper, and oblivious transfer allows parties to securely compute functions [Kil88, Gol98].

[1] The range of cryptographic protocols that can be proven secure in a purely information-theoretic sense (such as encryption with one-time pad) is extremely limited compared with the realm of possibilities "computational" cryptography offers.

[2] The names "private" vs. "public" may be misleading: we note that digital signatures, which are based on public keys, are in fact equivalent to OWF.

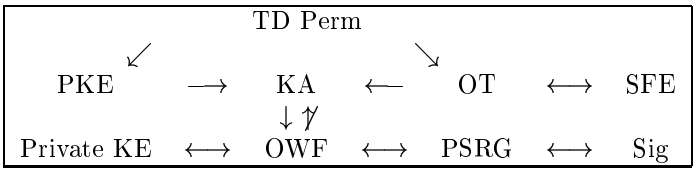

Figure 1: Separation between "Private Cryptography" and "Public Cryptography".

The first indication that the situation is not as simple in the "public" world as in the "private" one, was given by Rudich [Rud91].[3] Extending the techniques of [IR89], Rudich proved a black-box separation between KA in $k+1$ passes (i.e. by exchanging $k+1$ messages between the parties) and KA in k passes, for any k. This implies a separation between KA and PKE since PKE is equivalent to KA in one round (i.e. two passes). Another result relating primitives in the "public" world is of Even, Goldreich and Lempel [EGL85]. They showed that PKE with additional underlying properties imply OT. However, in general, very little was known on the relationship between PKE and OT. Thus, most of the relationships between the primitives in the "public" world were left unresolved. Figure 2 summarizes these relationships that were known before our results.

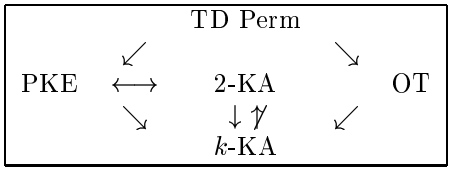

Figure 2: Previously known Relationships in the Public Cryptography World.

In this paper we make significant progress by resolving the missing relationships, with respect to black-box reductions. We show rather surprising separations between PKE and OT. This two-sided separation implies that in fact PKE and OT are *incomparable* with respect to black-box reductions, which makes it a unique situation. It is especially surprising given the result of [EGL85] that was interpreted by many as saying that PKE implies OT. Among our additional results are positive results in special cases, corollaries separating OT and KA, and separating trapdoor permutations from PKE and OT. These results are detailed in Section 1.2.

One way to interpret the results of this paper is as an indication that obtaining a simple and clean view of cryptography is even further beyond our reach than might have been suspected. Our results imply that we can not even divide all of cryptography into an hierarchy of assumptions (since PKE and OT are incomparable). Furthermore, we demonstrate that the picture of "public crypto" is not likely to simplify in some of its most central concepts.

1.1 The Primitives and Protocols

In their seminal work, Diffie and Hellman [DH76] gave the foundations for public-key cryptography. In particular, they introduced one-way functions (OWF) and trapdoor OWF. Informally, a family of functions is one way if a function f_k chosen from this family is easy to compute on any input but hard to invert on the average. Such a family is a family of *trapdoor* OWF if in addition the key-generation algorithm produces a trapdoor information $t = t(k)$ that allows its holder to invert the function f_k. One-way *permutations* and trapdoor one-way permutations are defined in the same way (for families of permutations). Diffie and Hellman also introduced the notion of key-agreement (KA) and provided the famous Diffie-Hellman KA

[3] In fact, the recent separation of [IR00, KSS00] between one-way functions and permutations implies that the situation is even not completely simple in the "private" world.

protocol. Informally, a KA is a protocol that allows two parties, Alice and Bob, to agree on a secret key in the presence of a polynomial-time (passive) eavesdropper (the key should be indistinguishable from random to such an eavesdropper).

Goldwasser and Micali [GM84] introduced the notion of semantically secure encryption schemes. Informally, these are encryption schemes such that any information on the message m that can be efficiently computed from its encryption $c = E(m)$ can also be efficiently computed without access to c. In this paper, the term public-key encryption (PKE) refers to semantically secure schemes. Goldwasser and Micali [GM84] also defined trapdoor predicates and showed their equivalence to semantically secure encryption schemes. Informally, trapdoor predicates are families of *probabilistic* functions over $\{0,1\}$ such that: (1) Given a key k it is easy to sample the output of the corresponding function p_k on either 0 or 1 but hard to distinguish these two distributions. (2) Given the trapdoor information $t = t(k)$, it is easy to distinguish an output on 0 from an output on 1. Yao [Yao82] showed that trapdoor permutations imply trapdoor predicates.

Recently, Bellare et al. [BHSV98] showed that one-way functions imply trapdoor one-way functions (with super-polynomial pre-image size). In addition, they showed that trapdoor one-way functions with polynomial pre-image size imply trapdoor predicates. These results imply that in some sense, trapdoor one-way functions are interesting if and only if their pre-image size is polynomial (indeed, most works in cryptography focus on trapdoor *permutations*). Finally, they showed that trapdoor predicates imply injective trapdoor functions in the random oracle model.

Oblivious transfer has several equivalent formulations [Rab81, CK88, EGL85, Cré87, BCR86, CS91]. The version used in this paper is that of one-out-of-two oblivious transfer ($\binom{2}{1}$-OT). Informally, a $\binom{2}{1}$-OT is a protocol between two parties Alice and Bob where Alice has as input two secret strings s_0 and s_1 and Bob has a secret bit b. If both parties follow the protocol, Bob learns the secret s_b whereas Alice learns nothing. In addition, even a cheating (polynomial-time) Bob (i.e. one that deviates from the protocol in an arbitrary way) cannot learn more than a single value in $\{s_0, s_1\}$ and even a cheating Alice does not learn anything during the run of the protocol. A somewhat weaker notion than OT is that of OT *secure against honest but curious parties* (also known as semi-honest parties). In this case, even cheating parties are assumed to follow the protocol with the following exceptions: (1) They keep track of all intermediate data during the run of the protocol. (2) After the last pass is completed, they may perform any (efficient) computation in order to extract additional information from their records. Using zero-knowledge proofs any honest OT protocol can be transformed into a general (malicious) OT protocol.

Oblivious transfer implies KA [Rab81, Blu83], signing contracts [EGL85], and in general any secure multi-party function evaluation [Yao86, Kil88, GMW87]. Even et al. [EGL85] showed how to construct OT from PKE that has the additional property that the distribution of cipher texts is "independent" of the encryption key (for two different keys, the corresponding distributions on cipher texts are indistinguishable). As a consequence, trapdoor permutations imply OT (see [Gol98]). Finally, several types of secure function evaluation in various models have been shown to imply OT [Kil91, KKMO00, BMM99, Kil00].

1.2 Our Results

We make significant progress in understanding both directions of the relationship between OT and PKE, as well as the relationships among these and KA, trapdoor permutations and functions. Specifically, the main questions we address are the following:

QUESTION 1: Does OT imply PKE?

In the specific case in which the oblivious transfer protocol consists of only one round (i.e. one message from Bob to Alice and one from Alice to Bob) we show:

Theorem 1 *One round OT even for honest parties implies PKE.*

In the general case (when the oblivious transfer protocol consists of any number of rounds) we show:

Theorem 2 *There is an oracle relative to which there exists multi-round OT (malicious) and no PKE.*

QUESTION 2: Does PKE imply OT?

We first generalize the result of [EGL85] by considering PKE that satisfies one of the following two properties:

- Property A: Public keys can be sampled "separately of private keys", (i.e. while preserving the semantic security of the encryption).
- Property B: Cipher texts can be sampled "separately of plain texts", (i.e. without gaining knowledge on the corresponding plain text).

Theorem 3 *PKE with properties A or B implies OT.*

We note that Properties A and B are both natural and have a similar flavor. Furthermore, as we show later, property B is implied by the property on PKE assumed by [EGL85]. Therefore, Theorem 3 strengthens the result of [EGL85]. In fact, our proof of the theorem was inspired by [EGL85].

In the general case, when the PKE does not have special properties, we show:

Theorem 4 *There is an oracle relative to which there exists PKE and no OT (even for honest parties).*

QUESTION 3: What is the Relationship Between OT and KA?

Generalizing Theorem 1, we obtain:

Theorem 5 *For every k, k-pass OT for honest parties implies k-pass KA.*

Extending Theorem 2, we obtain:

Theorem 6 *For every $k \geq 2$ there is an oracle relative to which there exists k-pass OT (malicious or honest) and no $(k-1)$-pass KA.*

As an immediate (but interesting) corollary of Theorem 4, we get:

Corollary 7 *There is an oracle relative to which there exists KA and no OT (even for honest parties).*

QUESTION 4: Do PKE and OT Imply Trapdoor Permutations?

Since trapdoor permutations imply both PKE and OT we obtain the following direct corollaries of our results:

Corollary 8 *There is an oracle relative to which there exists PKE but no trapdoor permutations.*

Corollary 9 *There is an oracle relative to which there exists OT but no trapdoor permutations.*

In fact, given the result of [BHSV98] we can strengthen this corollary and obtain:

Corollary 10 *There is an oracle relative to which there exists OT but no trapdoor functions with polynomial pre-image size.*

Finally, an interesting and important generalization of Theorem 4, is the following result.

Theorem 11 *There is an oracle relative to which there exist* injective trapdoor functions *and no OT (even for honest parties).*

Therefore, even a very minor weakening of trapdoor permutations does not seem to be enough as a building block for OT.

Some Remarks on Our Results

We note that in all our results, PKE may be replaced by trapdoor predicates, as these are equivalent [GM84]. Furthermore, since any two-party function that cannot be computed information-theoretically is equivalent to OT (in the malicious model) [BMM99], OT may be replaced by any such function (e.g., OR).

We have stated all negative results in terms of oracle separations. These imply that there are no black-box reductions between the corresponding primitives. As discussed in Section 2.2, there are two standard types of black-box reductions (one a weakening of the other). We note that a more careful statement of our results can be shown to apply to both these types (see Section 2.2 for more details).

As discussed above, the importance of black-box separations lies in the fact that almost all cryptographic reductions are black-box. Another motivation is that these are the only type of reduction possible when the assumption is *physical*, e.g. an OT channel which transmits bits with probability half. Nevertheless, it is important to note that such results have their limitations — some reductions used in cryptography are *not black-box*. An important (and almost single) example of a non black-box reduction is the proof that "all of NP has zero knowledge proofs" [GMW91] (see [IR89] for a more detailed discussion of oracle separations in cryptography). In our case, it is particularly important to notice that *the known reduction of (malicious) OT to honest OT* (the scenario where Alice and Bob are honest but curious) *is also not black-box* (since it relies on zero-knowledge proofs). This is exactly the reason that in each one of the theorems we use honest OT or malicious OT as to *strengthen* the result. For example, Theorem 1 gives a stronger result when stated for honest OT than when stated for malicious OT (since OT for malicious players trivially implies OT for honest but curious players). However, in Theorem 2 we show a negative result. Therefore it is stronger when stated for malicious OT since even when the OT is strong enough to resist malicious players, it does not imply PKE.

The main importance of the partial implications given in Theorems 1 and 3 is that they complement the separations of Theorems 2 and 4. Nevertheless, Theorem 1 that deals with one-round OT became somewhat more interesting on its own given the recent proposals of one round OT (even for malicious parties) based on the decisional Diffie-Hellman assumption (and other homomorphic encryption schemes) [AIR00, Nao00], and PKE with Property A (as we show below).

We stress that all of our separations are proven in the *uniform model* (as is the case in [IR89, Rud91]). In other words, all adversaries are assumed to be probabilistic polynomial-time machines rather than polynomial-size circuits. We see no reason why these separations would not apply to the non-uniform model. However, we note that proofs in the non-uniform model seem substantially harder (see e.g. [GT00]).

Some More Oracle Separations.

The Impagliazzo and Rudich methodology was first applied in [IR89, Rud91] to separate OWF from KA and to separate KA in k passes from KA in $k+1$ passes, for any k. These results have several interesting corollaries as described in the introduction. In [Sim98], Simon gives an oracle separation between OWF and collision intractable functions. Kim, Simon and Tetali [KST99], extend the work of [IR89] and use oracle separations as a mean to study the *efficiency* of black-box constructions (rather than their existence). In particular [KST99] show limits on the efficiency of black-box constructions of universal one-way hash functions based on one-way permutations. These bounds are improved by Gennaro and Trevisan [GT00] who also give bounds on the efficiency of black-box constructions of pseudo-random generators based on one-way permutations. In both cases the bounds are tight (in that they match the efficiency of known constructions). Finally, Impagliazzo and Rudich [IR00] use a result of Kahn, Saks and Smyth [KSS00], to give a black-box separation between OWF and one-way permutations.

Outline

In Section 2 we provide some details on black-box reductions and the oracle separation paradigm. We also give definitions and notation. In Section 3 we describe partial implications between primitives, in Section 4 we prove that OT does not imply PKE under black-box reductions, and in Section 5 we prove that PKE does not imply OT under black-box reductions. Finally, in Section 6 we prove our extensions regarding relationships with KA and trapdoor functions.

2 Preliminaries

2.1 Notation and Definitions

In this section we give some additional details on the definitions of the primitives and protocols that are the focus of this paper (these primitives were already described in the introduction). Our definitions here are rather informal. For a more formal treatment, the reader is referred to [Gol98] and references therein.

Notation and Conventions

We will abbreviate probabilistic polynomial time Turing Machine with the notation PPTM. An oracle PPTM is a PPTM that has access to a given oracle, such that in one time step it may receive the answer to a single query to the oracle. When discussing a primitive or a protocol relative to an oracle Γ, we assume that all the machines that are involved (including adversaries that try to break the primitive) are in fact oracle machines with access to the same fixed oracle Γ. We will sometimes refer to an oracle PPTM as simply a PPTM, as appropriate by the context.

We use the notation $poly(\cdot)$ to refer to some polynomially bounded function and $neg(\cdot)$ to refer to some function that is smaller than $1/p(\cdot)$ for any polynomial $p(\cdot)$ (for all sufficiently large inputs).

Public-Key Encryption

A public-key encryption scheme, consists of three probabilistic polynomial time algorithms (G, E, D) such that (for simplicity we fix ℓ to be both the security parameter and input length):

- G is the algorithm for generating keys. Given 1^ℓ as input, G outputs the pair (pk, sk) of public key and secret key s.t. $|pk| = |sk| = poly(\ell)$.
- E and D are the encryption and decryption algorithms. For every message m of length ℓ, for every pair (pk, sk) generated by G on input 1^ℓ, and all possible coin tosses of E, it should hold that $D_{sk}(E_{pk}(m)) = m$.

Intuitively, a public-key encryption scheme is *semantically secure* [GM84] if anything that can be computed on the plain text m given the cipher text $c = E_{pk}(m)$ can also be computed without access to c. More formally, (G, E, D) is semantically secure (against a chosen plain text attack), if for every PPTM A (the adversary that tries to extract information from the cipher text) there exists another PPTM A' such that for every efficiently sampalable distribution D on plain texts, for every efficient a priori information function h, and every predicate T (the information A tries to gain)

$$Pr(A(E_{pk}(m), pk, h(m), 1^\ell) = T(m)) < Pr(A'(h(m), 1^l) = T(m)) + neg(\ell)$$

Where $(pk, sk) = G(1^\ell)$ and m is a message (of length ℓ) sampled from D.

Two-Party Protocols

A two-party protocol, is a probabilistic process where two parties, Alice and Bob, exchange messages in turns. Each message sent by a party is a function of its input, its random string and previous messages exchanged by the parties during the *run* of the protocol. A two-party protocol is defined by these round functions. A protocol is efficient if the round functions can be computed in probabilistic polynomial time. A *pass* of the protocol consists of a *single* message sent from one party to the other. Therefore, each run of the protocol consists of the the two parties alternating passes. A *round* of the protocol consists of two passes. A protocol is called *k-rounds* (resp. *k-passes*) if every run of the protocol consists of exactly k-rounds (resp. k-passes). The output of each party is computed after its last pass. In this paper, we shall denote a k-pass protocol by k-protocol; so, k-KA refers to a k-pass Key Agreement protocol, while k-OT refers to a k-pass Oblivious Transfer protocol.

The sequence of all messages that are exchanged during a run is called *the conversation*. The *view* of each party consists of its input, random string, and the conversation. In case the protocol is performed between oracle machines, the view of each party also includes all the (query,answer) pairs asked by that party during the run.

Key Agreement

A key agreement protocol is an efficient protocol between two parties Alice and Bob. The input of both Alice and Bob is the security parameter ℓ written in unary. The outputs of both Alice and Bob are k-bit strings (for some $k = poly(\ell)$). If both output strings are the same, Alice and Bob are said to *agree on a secret*. In this case, the common output is called *the key*. A secure key agreement protocol (KA) is such that

Correctness: Alice and Bob agree with probability 1 (when they follow the protocol).

Secrecy: No PPTM Eve (the passive eavesdropper), that is only given ℓ and the conversation between Alice and Bob, can distinguish with non-negligible advantage the key from a uniformly distributed k-bit string

Oblivious Transfer

Oblivious transfer (OT) has several equivalent flavors. The one we use here is $\binom{2}{1}$-OT and we refer to it as simply OT. An OT protocol is an efficient protocol between two parties Alice and Bob. The inputs of both Alice and Bob include the security parameter ℓ written in unary. In addition, Bob has the bit b, and Alice has as input two k-bit strings s_0 and s_1 (where $k = poly(\ell)$) [4] The protocol is *correct* if when Alice and Bob follow the protocol, Bob outputs the string s_b with probability 1; note, that in OT Alice does not output anything.

We define the security of the protocol like in secure function evaluation. In other words, the security of the protocol is compared to the security in an ideal model, where a trusted party Carol, receives the secrets from Alice and Bob's bit, and sends s_b to Bob. The protocol is secure if a cheating Alice (resp. Bob) *running the protocol* can be simulated by a cheating Alice (resp. Bob) *in the ideal world*. In more detail:

- The view of any cheating Alice can be efficiently simulated by an algorithm that only has access to ℓ and to Alice's input (i.e. the strings s_0 and s_1 and any additional a priori information). This means that the output of this machine is indistinguishable from the view of the cheating Alice.
- The view of any cheating Bob can be efficiently simulated by an algorithm that only has access to ℓ, to Bob's input and to any single value in $\{s_0, s_1\}$ of its choice.

The definition of an *OT protocol secure against honest but curious parties* is as above with the simplifying assumption that even the cheating parties follow the protocol (i.e. send the correct messages specified by the protocol). Notice that the view of each party includes all the information that the party had access to during the run of the protocol. One can therefore think of an honest but curious party as one that follows the protocol with the following restrictions: (1) It does not erase any intermediate data during the run of the protocol. (2) After the last pass is completed, it may perform any (efficient) computation in order to extract additional information from its records (including oracle access if this is an oracle machine). We sometimes refer to such a protocol as an *honest OT*, and to the general OT protocol as *malicious OT*.

2.2 Black-Box Reductions.

The notion of a black-box reduction is somewhat confusing, as there are two different types commonly referred to as black box. We will call these types fully black-box reductions and semi black-box reductions.[5] A *semi black-box reduction* of a primitive P to a primitive Q is a construction of P out of Q that ignores the internal structure of Q (both in the construction itself and in its proof of correctness). Somewhat more formally, this is a construction of a polynomial time oracle machine M such that M^C implements the primitive P whenever C implements the primitive Q. Namely, Q is given in the construction as a black box rather than, say, as the description of an algorithm that implements it. In addition, the proof of correctness is also limited to a black-box access to Q. This means that for any polynomial-time oracle machine A_P,

[4] Sometimes OT is defined where s_0 and s_1 are bits rather than strings. Using the stronger definition (with strings) does not affect our results since in the honest but curious model an OT protocol for bits can easily be transformed into an OT protocol for strings (and the transformation does not increase the number of rounds).

[5] [IR89] refer to these as black-box reduction, and black-box construction, respectively. A fully black-box reduction is also known as a relativizing reduction.

there exists a polynomial-time oracle machine A_Q such that, if A_P with access to C breaks M^C (as an implementation of P), then A_Q with access to C breaks C itself (as an implementation of Q). Note that A_Q may use the internal structure of A_P (only C is given as a black box). A *fully black-box reduction* between P and Q is a semi black-box reduction, where in addition the proof is black box in a stronger sense: There is a polynomial time adversary A_Q which, given *oracle access* to any adversary A_P that breaks M^C (as an implementation of P), manages to break C (as an implementation of Q). Note that this kind of black-box argument is more restricted than the one allowed in semi black-box reductions. In this paper, by black-box reduction we refer to a fully black-box reduction.

The Oracle Separation Paradigm. Let P, Q be two cryptographic primitives. To separate P and Q with respect to black-box reductions it is enough to construct an oracle Γ such that relative to Γ the primitive Q exist whereas P does not. This in itself is enough to conclude that there are no fully black-box reductions from P to Q. In fact, Impagliazzo and Rudich [IR89] suggested a more powerful methodology which we follow in all of our separations. In [IR89], Γ is constructed as a combination of an oracle O and a PSPACE complete oracle. Relative to the PSPACE complete oracle, P=NP. Therefore in some sense, all of the computational hardness relative to Γ comes from the oracle O. The steps for separating P and Q are the following. First, prove that there is an implementation of the primitive Q using only O, which is secure with respect to Γ. That is, the implementation uses polynomial time and polynomial number of queries to O only, but it's security is proven with respect to polynomial time adversaries which have access both to O and to the PSPACE complete oracle in Γ. Second, prove that if $P = NP$ then relative to O there is no secure implementation of the primitive P. The proof should extend (given the implications of the PSPACE complete oracle) to showing that relative to Γ, there is no secure implementation of the primitive P. The implication of such a construction can be interpreted in two ways:

- As before, there are no fully black-box reductions from P to Q. But in addition:
- P=NP $\Rightarrow$ There are no semi black-box reduction from P to Q. Equivalently, providing such a semi black-box reduction is as hard as proving P$\neq$NP.

All our separation results follow this paradigm, and so the above two conclusions apply in all of them (we usually use the first formulation, but all our result can also be stated as the second). We will sometime loosely say that Q does not imply P, meaning that there is a separating under black-box reductions, as in the above paradigm.

3 Partial Implications

The main result of this paper is that PKE and OT are incomparable under black-box reductions. Nevertheless, we start by showing that PKE (resp. OT) with some special properties *does imply* the existence of (full fledged) OT (resp. PKE). In addition, we present a "pass-preserving" reduction from OT to KA. These partial implications serve two purposes. First, they show tightness of our separations (that are presented in Sections 4 and 5). Second, from a pedagogical standpoint, it is useful to understand the limitations of standard reductions between primitives as a first step in their separations.

3.1 Equivalence of PKE and 2-KA

We will now review known reductions between PKE and 2-KA that demonstrate their equivalence. This equivalence is used throughout the paper; when trying to construct PKE from some primitive, we might instead construct 2-KA, or vice versa.

Recall that 2-KA is a 2-pass KA protocol between two parties Alice and Bob. Let r be the random string of Alice and s the random string of Bob. Any 2-KA protocol has the following general structure:

Alice		Bob
r	$\xleftarrow{f_1(s)}$	s
$z = R'(r, f_1(s))$	$\xrightarrow{f_2(r, f_1(s))}$	$z = R(s, f_2(r, f_1(s)))$

In the above protocol, f_1, f_2, R, and R' are some efficient algorithms, and z is the key which Alice and Bob agree upon. Finally, recall that the protocol is secure if (no efficient eavesdropper) Eve that is given the communication $\langle f_1(s), f_2(r, f_1(s))\rangle$ as input, is able to distinguish between the key z and a uniformly distributed k-bit string, with non-negligible advantage.

Given this 2-KA protocol, we can define the three probabilistic algorithms of the PKE scheme (i.e. the generator (G), encryption (E), and decryption (D)), as follows:

- G: $pk = f_1(s)$, $sk = s$.
- E: $E_{pk}(m, r) = \langle f_2(r, pk), R'(r, pk) \oplus m\rangle$.
- D: $D_{sk}(c_1, c_2) = R(sk, c_1) \oplus c_2$.

The correctness of E and D, as encryption and decryption algorithms, follows from the fact that Alice and Bob agree on the key in the 2-KA protocol. The semantic security of the scheme is a simple consequence of the secrecy of the KA protocol.

Constructing a 2-KA protocol from a PKE scheme is straightforward. In the first pass, Bob sends to Alice his public key, pk. In the next pass, Alice selects the key z and sends its encryption, $E_{pk}(z, r)$. Bob can then compute the secret using the decryption algorithm D_{sk}, where sk is his secret key. The correctness and secrecy of the protocol follow from that of the PKE scheme.

Finally, we note that for $k > 2$, there are no black-box constructions of PKE out of k-KA (since Rudich [Rud91] gave a black-box separation between k-KA and $(k-1)$-KA, and since PKE is equivalent to 2-KA).

3.2 Pass-preserving Reduction of KA to OT

That OT implies KA was part of folklore. However, the construction we explicitly present here is *pass-preserving*. Hence k-OT implies k-KA, and as a consequence 2-OT (or 1-round OT) implies PKE. In Sections 4 and 6, we show that it is not possible to reduce the number of passes when constructing KA from OT. More formally, there are no black-box constructions of $(k-1)$-KA out of k-OT. This result also shows that, in general, OT does not imply PKE.

It is intuitive that an OT may imply a KA. After all, an OT allows Alice to send one of her two secrets, s_0 or s_1 to Bob and such a secret can serve as the key. The protocol used below follows this intuition: Alice sends s_0 to Bob and they agree on it as their key. The correctness of the protocol is immediate. A more subtle point is to show that s_0 remains secret even when the OT protocol is performed with Bob's input fixed to zero.

Theorem 5 *For every k, k-OT for honest parties implies k-KA.*

Proof: The KA protocol between A and B is the following: A selects two random k-bit strings s_0 and s_1. A and B then simulate Alice and Bob, respectively, in an OT protocol. The inputs of Alice in the OT are s_0 and s_1 whereas the input b of Bob is set to 0. Finally, A outputs s_0 and B outputs the value Bob received in the OT protocol. By the correctness of the OT protocol, Alice and Bob agree with probability one. It is also immediate that the KA has exactly the same number of passes as the OT.

To complete the proof it is enough to show that if the KA is not secure, then neither is the OT. Assume that the KA protocol is indeed not secure. This means that there exists an efficient eavesdropper E that, given the communication between A and B, distinguishes s_0 from a uniformly distributed string. There are two possible scenarios: (1) E distinguishes s_0 from random *even when Bob's input, b, is set to* 1 (instead of 0). (2) When Bob's input is set to 1, E has at most a negligible advantage in distinguishing s_0 from random. In each one of the cases, E can be used to break the OT. More specifically, in each case one of the (honest but curious) parties can use E to learn more than they should. In the first case, the view of a honest Bob with input 1 gives s_1 (with probability one) but also gives information on s_0 (namely, the ability to distinguish it from random). In the second case, the view of a honest Alice gives a way to guess the value of b with non-negligible advantage over 1/2. The prediction essentially goes as follows: If E distinguishes s_0 from random (given the communication between Alice and Bob) guess "$b = 0$". Otherwise, guess "$b = 1$". Testing which is the case can be done given s_0 itself (which is also part of the view of the honest Alice). □

Note that, although Theorem 5 is stated for honest OT, it also applies to malicious OT (as an immediate consequence).

3.3 Special Cases of PKE that imply OT

In Section 3.2 we showed that k-OT implies k-KA. It is therefore natural to explore the converse direction, i.e., does k-KA imply k-OT? In Section 5, we show that the answer is no. In fact, even 2-KA does not imply OT (with any polynomial number of rounds). Yet, in this section, we show that if a KA has some special properties then it is possible to construct OT from it. We then translate the special properties of the KA to special properties of PKE that are sufficient for the construction of OT.

Our starting point is the constructions in [EGL85, Gol98] of OT based on PKE with additional properties (or trapdoor-pernutations). We offer constructions based on weaker properties. To this end, we first abstract and generalize their constructions, viewing them in terms of KA (rather than PKE).

Suppose Alice has a single secret s that she wants to send to Bob in the presence of an eavesdropper. Alice and Bob can perform a KA protocol. At the last pass Alice can also send $z \oplus s$, where z is the key they agree upon. This allows Bob to compute s while an eavesdropper learns nothing about s. In an OT protocol, however, Alice has two secrets s_0 and s_1, Bob wants to learn s_b and Alice should not learn b. Finally, Bob should not learn more than one secret. If we ignore this last requirement, then Alice can just send *both* s_0 *and* s_1 to Bob using two parallel executions of the KA protocol. Surprisingly, in some cases this simple (and silly) protocol can be transformed into an OT protocol *for honest parties*. The idea is simple, the only change that is needed is that in the execution of the KA that corresponds to s_{1-b} *Bob should "fake" his role*. Namely, it only acts as if he is trying to agree on a key, where in fact he learns nothing about the key that Alice outputs for that execution. More specifically, the properties needed from these fake runs are the following:

1. Alice should not be able to distinguish a fake run from a real run. Therefore, the view of Alice can be simulated by performing two real (or fake) runs of the KA protocol.

2. In a fake run, Bob cannot distinguish the key Alice outputs from random. Therefore, it is enough to simulate the view of Bob when instead of sending s_{1-b}, Alice sends a random string. Such a view can be easily simulated (given the string s_b that is available to the simulator).

We now define two properties for PKE, such that if a PKE satisfies one of these properties, then a corresponding KA protocol is fakeable. This will imply that honest-OT can be based on a PKE with one of these properties.

A: One can efficiently select a string pk with a distribution which is indistinguishable from that of a public key generated by G, while preserving the semantic security of the encryption E_{pk}. Namely, even the algorithm that selects pk does not learn anything on a message m from $E_{pk}(m)$.[6]

B: For any public key pk, one can efficiently select a string c which is indistinguishable from a random encryption of a randomly chosen message m, without having information on the decryption $D_{sk}(c)$. (i.e., $D_{sk}(c)$ is pseudo-random even to the algorithm that selects c).

Proposition 12 *PKE having property A implies (2-pass) OT for honest parties.*

Proof: The fakeable KA protocol looks as follows:

Alice		Bob
	$\xleftarrow{pk}$	Run G to select(pk, sk)
Select z	$\xrightarrow{E_{pk}(z)}$	$z = D_{sk}(E_{pk}(z))$

[6] In particular, the semantic security requirement implies that the selecting algorithm cannot just run G, ignore sk and output pk. Similarly, for property B the selecting algorithm cannot simply encrypt a random message and output the resulting ciphertext.

To fake this protocol, Bob selects pk using the method guaranteed by property A (instead of running G). In this case, Bob learns nothing about the key z (which is sent encrypted), because of the semantic security in property A. Nevertheless, since pk has an indistinguishable distribution from that obtained by selecting it using G, we get that the fake run is indistinguishable from a real run (even to Alice).

By the reduction described above, we can obtain 1-round honest OT protocol from this fakeable KA protocol (using parallel executions of a real KA and a fake one). We note that this OT protocol was independently found by [Bei]. □

Proposition 13 *PKE with property B implies (3-pass) OT for honest parties.*

Proof: The fakeable KA protocol looks as follows:

Alice		*Bob*
$G : (pk, sk)$	$\xrightarrow{pk}$	
	$\xleftarrow{c=E_{pk}(r)}$	Select r
Select z	$\xrightarrow{\alpha=z\oplus D_{sk}(c)}$	$z = \alpha \oplus r$

To fake this protocol, Bob selects c using the method guaranteed by property B (instead of applying E_{pk} to r). In this case, Bob learns nothing since $D_{sk}(c)$ is indistinguishable to Bob from random. Nevertheless, since c is a pseudo-random ciphertext, the fake run is indistinguishable from a real run (even to Alice).

By the reduction described above, we can obtain 3-pass honest OT protocol from this fakeable KA protocol. □

Propositions 12 and 13, together, complete the proof of Theorem 3.

Discussion of the Properties. These properties are convenient abstractions of properties shared by well-known cryptosystems. For example, property A is true for El Gamal Encryption (corresponding to Diffie-Hellman KA): A public key p, g, y (where $y = g^x$ for the secret key x) can be sampled with identical distribution to that produced by the generating algorithm, since g^x is a permutation and therefore y can be selected at random. This is semantically secure if the PKE itself is semantically secure (i.e., under the decision DH assumption). Property B is true for many PKEs since in many cases a ciphertext is uniformly distributed on some easily sampalable set. One example is the general construction of PKE based on any trapdoor permutation [Yao82, GM84] (where a random ciphertext is a uniform string plus a uniform element from the domain of the permutation).

As mentioned above, the methodology of our constructions (using fakeable KA) is based on generalizing constructions of [EGL85]. There, OT was constructed based on PKE with the property that the distribution of ciphertexts is essentially the same for any public key. This is a special case of our property B (and thus a stronger restriction). Finally, we note that similar properties were recently defined by [DN00], for different purposes. They call these properties *oblivious public-key generation* and *oblivious ciphertext generation*.

4 OT Does Not Imply PKE

In this section we construct an oracle Γ_3 relative to which there is 3-OT, but no 2-KA (thus, no PKE). This can be extended (as we argue in Section 6) to show that for every $k \geq 3$ there is an oracle Γ_k relative to which there is k-OT but no $(k-1)$-KA. Note that this tightly matches our positive result of Section 3.2, where we showed that k-OT does imply k-KA.

The oracle Γ_3 is defined as follows.

- Three uniformly distributed, length-tripling functions $f_1(\cdot,\cdot)$, $f_2(\cdot,\cdot)$, and $f_3(\cdot,\cdot,\cdot)$. We restrict f_3 to be injective.
- A function R satisfying $R(w,\alpha) = z$ whenever $\alpha = f_3(z, r, f_2(w, f_1(z, r)))$ for some $|z| = |r| = |w|$ (we call such pairs (w,α) *valid*), and random everywhere else. That is, when (w,α) is not a valid pair, $R(w,\alpha)$ returns a random value of size $|w|$. Note that R is well defined, since f_3 is injective.

- a PSPACE-complete oracle.

Note that we can essentially ignore the restriction that f_3 is injective since, with probability 1, a random length tripling function is injective for sufficiently long inputs.

In Section 4.1 we will first show that relative to Γ_3, 3-KA is possible. We will then observe that this KA is in fact "fakeable" in the sense discussed in Section 3.3. We will thus conclude that relative to this oracle, 3-pass honest OT is possible. Finally, we will show how this honest OT protocol may be modified to yield a 3-OT for malicious parties. In Section 4.2 we will show that relative to this oracle, 2-KA is not possible (thus PKE is not possible). The proof of impossibility follows the arguments of Rudich [Rud91], who showed another oracle relative to which 2-KA is not possible, but 3-KA is.[7]

4.1 3-OT Using the Oracle

3-KA protocol

The following is a 3-KA protocol using Γ_3:

Alice		*Bob*
Select z, r	$\alpha_1 = f_1(z,r) \longrightarrow$	
	$\alpha_2 = f_2(w,\alpha_1) \longleftarrow$	Select w
Output z	$\alpha_3 = f_3(z,r,\alpha_2) \longrightarrow$	Output $R(w,\alpha_3)$

The correctness of the protocol follows from the definition of R, which implies that Bob's output $R(w,\alpha_3) = R(w, f_3(z,r,f_2(w,f_1(z,r)))) = z$ is the same as Alice's output. To show the secrecy of the protocol, we need to show that given the conversation, z is indistinguishable from a uniformly chosen string. If R was not part of the oracle, proving this is standard using the fact that the functions are random. It can be shown that this remains true even in the presence of R, since, by the fact that the functions are length tripling, it is hard to sample elements from their range without actually applying the function, and so no "useful" application of R is possible based on the conversation alone. The argument is similar to ones used by [IR89, Rud91] and in our Section 5, and is omitted here.

3-OT for honest parties

Observe that the 3-KA protocol presented above is in fact fakeable: Bob in his turn can select a random α_2 of an appropriate length, and since f_2 is a random function, this fake α_2 is indistinguishable to Alice from one generated by applying $f_2(w,\alpha_1)$ for some w. On the other hand, without a w such that $\alpha_2 = f_2(w,\alpha_2)$, Bob cannot distinuish z from random (and indeed such a w not only is hard to find, but it is likely to not even exist).

Since the 3-KA is fakeable, a 3-OT for honest parties is possible, as described in Section 3.3, by executing in parallel a real and a fake run of KA, where Alice also sends her secrets concealed with keys agreed upon in the two executions.

General (malicious) 3-OT

The 3-OT described above is only secure for honest parties, since it relies on honest Bob faking one of the parallel KA executions (and not using two real runs). However, relying on the properties of our oracle functions, we are able to construct a malicious 3-OT protocol. In fact, this 3-OT will be even simpler than the one for honest parties.

The protocol is based on the same idea of executing two parallel runs of KA, but this time Bob (even when malicious) is forced to run at most one real KA run, while the other run must be fake. This is done by

[7] An alternative proof of our result that 3-OT does not imply 2-KA may use a very similar oracle to the one used by [Rud91], with a small (but essential) modification which allows for a 3-OT protocol (and not only 3-KA). We prefer the alternative presented here, since it seems to capture 3-KA in a very general and natural way, containing a function for each pass, and a function for the reconstruction.

letting Bob choose α_2 for only one of the runs, while for the other run α_2 will be such that Bob cannot find a corresponding w, and thus cannot distinguish Alice's secret from random, as argued above. To guarantee the latter, we use a simple idea: use the same α_2 for both runs. More specifically, the 3-OT protocol is as follows.

Alice		Bob
input : s_0, s_1		input : b
Select z_0, z_1, r_0, r_1;		
$\alpha_1^0 = f_1(s_0, r_0), \alpha_1^1 = f_1(s_1, r_1)$	$\xrightarrow{\alpha_1^0, \alpha_1^1}$	
	$\xleftarrow{\alpha_2 = f_2(w, \alpha_1^b)}$	Select w
$\alpha_3^0 = f_3(z_0, r_0, \alpha_2), \alpha_3^1 = f_3(z_1, r_1, \alpha_2),$		
$\beta_0 = z_0 \oplus s_0, \beta_1 = z_1 \oplus s_1$	$\xrightarrow{\alpha_3^0, \alpha_3^1, z_0 \oplus s_0, z_1 \oplus s_1}$	Output $R(w, \alpha_3^b) \oplus \beta_b$

The security of this protocol follows from the fact that it is hard to genreate α_2 that corresponds to some α_1 without actually applying the function $f_2(w, \alpha_1)$ for some w, and thus it is hard for Bob, even when malicious, to find one α_2 that corresponds to both α_1^0 and α_1^1 (moreover, such an α_2 most likely does not even exist). It follows that α_2 is "real" for at most one of the runs, whereas for the other run it is fake, namely Bob cannot distinguish the corresponding secret from random. Thus, Bob's view can be simulated by an efficient algorithm that may ask for a single value s_b. Simulating the view of a malicious Alice is standard, and therefore the OT protocol is secure.

4.2 No 2-KA Relative to the Oracle

The proof that no 2-KA protocol exists relative to Γ_3 is quite complicated, but very similar to the proof of Rudich in [Rud91], who showed that no 2-KA is possible relative to his own oracle. His arguments, in turn, rely on techniques developed by Impagliazzo and Rudich [IR89], some of which are sketched in the following Section 5. Since the proof of this part contains no new technical ideas beyond those of [Rud91], we only present a very informal intuition behind the proof. We note that some of these intuitions are further developed in Section 5.

First, it is proven in [IR89] that for an oracle containing a random function and a PSPACE-complete oracle, no KA is possible. Informally, they show that in that setting, for any protocol, an evesdropper Eve who is given the conversation can guess "everything both Alice and Bob know in common" with non-negligible probability. Thus, if Alice and Bob agree on a secret, Eve can also guess the secret. Following the same proof, relative to our oracle Γ_3 there is no KA protocol which never queries R (in any polynomial number of passes).

We now want to argue that R is not "useful" in any 2-pass protocol, and thus no 2-pass KA exists. It is intuitively clear that querying R on inputs (w, α) which are not of valid form is not useful, since R just outputs a random value, and we are again in a setting similar to that of [IR89]. Next, it can be shown that, if during a protocol R was queried on some valid input $(w, f_3(z, r, f_2(w, f_1(z, r))))$, with very high probability it must be the case that prior to this point f_1 was queried on (z, r), then f_2 was queried on $(w, f_1(z, r))$, and then f_3 was queried on $(z, r, f_2(w, f_1(z, r)))$.

Now, consider any *2-pass* protcol, and consider the first "useful" application of R. As explained above, we may assume that with non-negligible probability, Eve knows everything that Alice and Bob know in common before R is applied. Assume (w.l.o.g.) that Alice is the party about to apply R on some valid input (w, α). Since the protocol is 2-pass and f_1, f_2, f_3 should have been applied sequentially, it follows that one of the parties applied both f_1, f_2 or both f_2, f_3 on the corresponding inputs. In either case, at that point that party knew all of w, z, r. If this party is Alice, we may eliminate her application of R without changing the correctness or secrecy of the protocol, since she already knows the output z; thus, in this case the application of R is not useful. On the other hand, if this party is Bob, Bob knows w, z, r, and by accessing f_1, f_2 and f_3 he can also find α; thus (w, α) is already known to both Alice and Bob, and therefore it is known to Eve with non-negligible probability, in which case Eve herself can apply $R(s, \alpha)$.

5 PKE Does Not Imply OT

In this section we construct an oracle Γ_{PKE} relative to which there is PKE, but no OT (not even honest-OT) in any polynomial number of rounds, thus proving Theorem 4. We use a natural oracle containing the functions f_1, f_2, R to be used for key generation, encryption, and decryption, respectively. We must, however, take some care to define these functions such that the special properties of Section 3.3, under which PKE implies OT, do not hold. Towards this end, making f_1 and f_2 length expanding and random will guarantee that it is hard to generate a valid public key or a valid encryption without actually applying f_1 (on the *secret key*) or applying f_2 (encrypting some *message*), respectively. Furthermore, providing ways to *test* whether a given string is a valid public key and whether a given string is a valid encryption, will guarantee that it is even hard to come up with strings that "look" valid.[8] In Section 5.2 we show that these modifications of f_1, f_2 and R are (not only necessary but also) sufficient in order to guarantee the impossibility of OT. The oracle Γ_{PKE} is thus defined as follows.

- f_1 is a uniformly distributed, length-tripling function.
- f_2 is an injective, uniformly distributed, length tripling function on the set of its *valid* inputs. An input $\langle m, r, y\rangle$ is valid for f_2 if for some x, both $y = f_1(x)$ and $|x| = |m| = |r|$. On any invalid input f_2 outputs $\perp$.
- R satisfying $R(c, x) = m$ whenever $c = f_2(m, r, f_1(x))$ and $|m| = |r|$. Otherwise, $R(c, x) = \perp$. R is well defined since f_2 is injective.
- A $PSPACE$-complete oracle.

We will call an input to R *valid* if the output on this input is not $\perp$ (this is consistent with the definition of valid inputs to f_2).

Note that we can essentially ignore the restriction that f_2 is injective. This is because, with probability 1, a uniformly distributed length-tripling function is one-to-one on all sufficiently long inputs. Also note that f_2 provides a way to test whether a given y is in the range of f_1 or not, e.g., by calling $f_2(\vec{0}, \vec{0}, y)$ and checking whether $\perp$ is returned. Similarly, R provides a way to test whether a given c is valid with respect to x, namely wheter $c = f_2(m, r, f_1(x))$ for some m, r.

5.1 PKE Using the Oracle

PKE under this oracle is straightforward: The key generation PPTM $G(1^l)$ chooses a random $s \in_R \{0,1\}^l$, and sets $sk = s$, $pk = f_1(s)$; The encryption PPTM chooses a random string $r \in_R \{0,1\}^l$ and sets $E_{pk}(m) = f_2(m, r, pk)$; The decryption PPTM simply calls R, namely $D_{sk}(c) = R(c, SK)$.

The correctness of this PKE follows directly from the definition of the oracle, and security can be proved in a standard way from the fact that f_1 and f_2 are random functions.

Note that this PKE protocol does not use the fact that the functions f_1, f_2 are "testable": The same protocol would work even if f_2 and R returned a random string instead of $\perp$ when their input is not valid. The testability property will be necessary for guaranteeing that no OT exists, as we show next.

5.2 No OT Exists Relative to This Oracle

We prove that no OT protocol exists relative to Γ_{PKE}. Since our proof is technically involved, we start by providing a high level overview of the main ideas and challenges, concentrating on the intuition behind the proof. Some additional technical details follow. A complete proof is deffered to the full version.

[8] Without such a test, choosing a random string of the appropriate length will look valid. This would mean that Properties A and B of Section 3.3 hold for the PKE and therefore a honest OT protocol is still possible.

High Level Overview

Informally, we prove that there is no honest-OT by proving that for any protocol between Alice and Bob, honest-but-curious Alice can find out "everything that Bob can learn about her". This will imply that, in particular, she can find out which of her secrets Bob can learn, thus implying that the protocol cannot be a secure OT (since either Bob can learn both secrets, or he cannot learn any secret, or Alice can find out which secret he was after).

Towards making this intuition more formal, we start by modeling the concept of "knowledge", as in [IR89, Rud91], via the queries made to the oracle (and its answers), and modeling what two parties know in common as their intersection queries, namely queries that they both made to the oracle. Further, we require that any protocol for OT take a normal form, which includes the requirement that Alice queries the oracle on her two secrets, and Bob queries the oracle on the secret he recovered. Now, we are left with proving that Alice can indeed find out all the intersection queries, and thus no OT exists, as explained above.

To see how we go about proving this, let us start by taking a closer look at the way our oracle Γ_{PKE} was constructed. First, the oracle contains random functions f_1, f_2 and a $PSPACE$-complete oracle. Following the proof of [IR89], if these were the only parts of the oracle, KA would not be possible. Specifically, [IR89] prove that for such an oracle there exists an "eavesdropper" PPTM Eve which with high probability, given the conversation alone, can find a polynomial length list containing all the intersection queries (and thus containing the agreed key).[9] However, by adding R to our oracle, KA becomes possible (as shown in Section 5.1). Moreover, as discussed above, with f_2 or R defined as random everywhere, OT is also possible. Therefore, a crucial part of our oracle is that f_2 and R are defined on valid inputs only, thus providing a way to test whether a string is valid. Also crucial is that valid inputs are hard to guess (since f_1 and f_2 are length expanding and random). We will argue that these properties guarantee that OT is impossible (while still maintaining the possiblity of KA, as shown by the protocol of Section 5.1).

What makes the validity tests so powerfull that they prevent OT? The main idea is the following: whenever a new intersection query is created *the validity tests help detecting and verifying it.* To see that, let us consider how intersection queries are created. One kind of intersection queries are those created without the use of R. In some sense, these are not really interesting since both parties (and even an eavesdropper) are aware of such common knowledge. Therefore, we only need to care about intersection queries that are created by R. A typical example is the one obtained by the PKE of Section 5.1: One party selects x and sends $y = f_1(x)$ to the second party which selects m and r and sends $c = f_2(m, r, y)$ back to the first party. The knowledge they both have in common now contains m (since the first party can compute $m = R(c, x)$). How can Alice detect such intersection queries? Intuitively, all she needs to do is handle two cases: (1) A new intersection query c is created. Alice knows x such that $m = R(c, x) \neq \perp$. If Alice did not obtain c herself (by a previous query to f_2) she can deduce that Bob did (since the only way to obtain a valid output of f_2 is essentially by *querying* f_2). Alice can therefore conclude that m is an intersection query. (2) A new intersection query $c \neq \perp$ is created. Alice knows m, r and y such that $c = f_2(m, r, y)$. If Alice did not obtain y herself (by a previous query to f_1) she can deduce that Bob did (since the only way to obtain a valid output of f_1 is essentially by *querying* f_1) and he therefore knows x such that $y = f_1(x)$. Alice can therefore conclude that m is an intersection query (since Bob can compute $m = R(c, x)$). The formal proof is of course by far more subtle. We now give some details on the proof technique.

Overview of Proof Technique

To carry out the intuitive arguments above, we would like to extend the proof techniques of [IR89] to work for our (more complex) oracle. However, it is clear that in our setting there is no Eve which can find the intersection queries from the conversation alone, since otherwise KA would not be possible. Nevertheless, we are able to use similar techniques to prove a claim appropriate for our setting. As explained above, this claim is roughly saying that (curious) Alice can find the intersection queries from her view. One subtle and important difference between our claim and the one used by [IR89], is that for us it is not enough to merely find a polynomial size list containing the intersection queries (indeed, this would be trivial for Alice, by

[9] We note that [IR89] work with an oracle containing a single random function, but the same proof can be extended to an oracle with two functions f_1, f_2 where f_1 is random, and f_2 is random on valid inputs of the form $(m, r, f_1(x))$, and is $\perp$ otherwise.

simply outputting all her queries). Instead, for our rationale to go through and imply that no OT exists, the list output by Alice should also have the property that (curious) Bob *can* indeed find all the queries on the list. This is exactly where we will use the power to test for validity of a string (from the range of f_1 or f_2): Alice will use this power to make sure she does not add to her list queries that Bob cannot find.

Putting the above intuition together, the main lemma we prove is the following.

MAIN LEMMA (INFORMAL STATEMENT): For every OT protocol (Alice, Bob) there are PPTMs E_A, E_B (which can be thought of as the "curious" parts of Alice and Bob, respectively) such that E_A gets as input Alice's view and outputs a list L_A, E_B gets as input Bob's view and outputs a list L_B, and the following is satisfied (with good probability):

- L_A contains all intersection queries.[10]
- $L_A \subseteq L_B$.

That is, intuitively, anything Bob learned about Alice is on her list L_A, and anything he cannot learn (even when behaving curiously) is not on her list. Thus, if Bob learns one of her secrets but cannot learn the other one, Alice knows which is the secret he learned, and OT is not possible.

Some More Details — Repeated Sampling Paradigm

Repeated sampling is an important method, used in the definition of E_A and E_B in the proof of the main lemma. This method is adapted from [IR89]. We now give a very high level description of repeated sampling in the context of [IR89] and in the context of our proof.

Recall that the oracle world of [IR89] contains a random function f and a $PSPACE$-complete oracle. Let Eve be the evesdropper that breaks the KA protocol in [IR89]. Eve maintains at each stage a polynomial size list L containing all intersection queries between the two parties upto this point. To guess the next intersection query (if one is made in the next round), Eve repeats polynomialy many times the following process:

Simulation Phase Sample an (almost) uniformly distributed run of the protocol that is consistent with the communication so far and the partial knowledge of Eve on f. We stress that any simulated query to f that was not determined so far gets a simulated answer (therefore, the probability distribution of the simulated run is also over random f's consistent with Eve's partial knowledge). Based on a result of [JVV86] it is shown in [IR89] (see their Corollary 3.2), that this sampling phase can be performed efficiently (using the $PSPACE$-complete oracle).

Updating Phase All the queries of the simulation phase are now asked from the *actual* oracle. They are subsequently added to L.

Why does this work? Assume that in the next round Bob has a non-negligible chance of making a new intersection query. In this case, a non-negligible fraction of possible Alice's already made this query, and thus by sampling them Eve has a chance to find it too. Another way of looking at this intuition is that when sampling Alice, in each execution of the process, during the simulation phase all "made-up" oracle answers are either: (1) Consistent with Bob's view of the oracle (i.e., they are not intersection queries). In this case the simulated Alice view is a possible one from Bob's point of view (thus there is nothing "hidden" that Alice and Bob both know which Eve does not); or (2) The answers are not consistent with Bob's view. In this case the simulation phase is useless, but during the update phase a new intersection query will be found, making progress towards having a "good" simulation (as in (1)).

The repeated sampling technique of [IR89] can be partially carried out to our setting. In our case E_A and E_B will use repeated sampling to maintain the lists L_A and L_B that satisfy the desired properties of the main lemma (up to the current stage). Nevertheless, the use of repeated sampling in our context is more subtle. The main difference is that letting E_A and E_B simulate runs of the protocol that are consistent with the conversation alone is not good enough. The place where the previous intuition fails with an oracle like ours

[10] To be a bit more accurate, L_A will contain all queries that are intersections with probability larger than some fixed threshold ε. Nevertheles, such a list L_A is good enough for our arguments.

(which has R as well), is that now it is possible that a made-up simulated oracle is both inconsistent with Bob's view and does not yield "progress" via an intersection query. For example, assume Alice chooses x and sends $y = f_1(x)$ to Bob, then receives $c = f_2(m, r, y)$ from him, and applies $R(x, c)$. Then in the simulation phase, a wrong $\hat{x}$ will be sampled ("pretending" that $f_1(\hat{x}) = y$), but asking for the real value of $R(\hat{x}, c)$ in the update phase provides useless information. This can happen in every repetition for polynomially many times, without revealing any new intersection queries. The solution comes from the fact that E_A has access to all of (real) Alice's view, and E_B has aceess to (real) Bob's view. Therefore, E_A and E_B sample runs of the protocol that are consistent with *both the conversation and this additional information*. In the example above, Bob's view already contains m, and Alice's view contains the real x such that $f_1(x) = y$, thus after invoking $R(x, c)$ she obtains the real m (and R is never invoked on the useless $(\hat{x}, c)$).

We conclude with briefly mentioning two additional subtelties of our proof:

- The generation of almost uniform runs of a protocol (that are consistent with some partial information) is more complex in our oracle (though still doable).
- While creating the list L_A, we cannot be as rackless as Eve is in the descryption above. Namely, E_A cannot add all the update queries to L_A since L_B may not contain them. Therefore, E_A should be very conservative in creating L_A. This is where she uses the validity tests described above. On the othere hand, E_B can be much more liberal and put all his knowledge into L_B (which creates the asymmetry between L_A and L_B).

6 Relationships with KA and with Trapdoor Functions

Our results extend beyond the relationship between PKE and OT, to show interesting relationships to KA, trapdoor functions, and trapdoor permutations. In this section we sketch our additional results (some of which follow as direct corollaries of our previous theorems, and some require extensions of the proofs).

6.1 Relationship between OT and KA

Recall that OT is known to imply KA, and that in fact KA can be constructed from OT in a pass-preserving way, as we showed in Section 3.2. That is, k-OT implies k-KA. However, we now show that a construction that reduces the passes is not possible.

k-OT does not imply $(k-1)$-KA, for any $k \geq 2$. In Section 4 we have constructed an oracle Γ_3 with the desired properties for $k = 3$. This construction can be generalized to any k in a straightforward way, by using Γ_k containing (in addition to a PSPACE-complete oracle) random length tripling functions $f_1, \ldots, f_k$, and a reconstruction function R such that $R(w, \alpha) = z$ whenever α is of the appropriate form for k sequential applications of $f_1, \ldots, f_k$ alternating between applications of the form $f_i(z, r, \alpha_{i-1})$ and of the form $f_i(w, \alpha_{i-1})$. The oracle is slightly different (syntactically) in the case that k is odd and the case k is even, depending on whether the first function f_1 is defined on (z, r) (when k is odd), or on w (when k is even). Proving that a k-OT (malicious) protocol exists relative to Γ_k, as well as proving that no $(k-1)$-KA protocol exists, is very similar to the corresponding proofs for $k = 3$.

Note that the above result clearly implies a separation between k-KA and $(k-1)$-KA (which was proven in [Rud91]), and a separation between k-OT and $(k-1)$-OT (both of which are weaker).

KA does not imply OT. Since we have proved that PKE (or 2-KA) does not imply OT (in any number of rounds), this immediately yields the corollary that, while OT implies KA, the converse is not true. That is, KA does not imply OT (under black-box reductions).

6.2 Relationships with Trapdoor Functions

Trapdoor permutations vs. PKE. While trapdoor permutations are known to imply PKE, our results imply that the converse is not true, namely PKE does not imply trapdoor permutations. This is so

since trapdoor permutations imply OT, and thus relative to our oracle Γ_{PKE}, PKE exists but trapdoor permutations do not.

Trapdoor functions vs. OT. By the result of Bellare al. [BHSV98], trapdoor functions with polynomial pre-image size imply PKE. Thus, we conclude that OT does not imply trapdoor functions with polynomial pre-image size (obviously, it follows that OT does not imply injective trapdoor functions, and in particular it does not imply trapdoor permutations).[11]

On the other hand, we may extend our results to prove that injective trapdoor functions do not imply OT. For this purpose we may use our oracle Γ_{PKE}, for which we already know that no OT exists. We may now consider the family $\{T_y(m,r) = (f_2(m,r,y),r)\}$ where the key generation algorithm first chooses a random t as the trapdoor information, and sets the key to $y = f_1(t)$. It is easy to see that T_y is easy to compute, and easy to invert given the trapdoor (and access to R). The one-wayness of T_y (without the trapdoor information) follows from the fact that f_2 is random (thus it is hard to invert without access to R), and the fact that f_1 is random (thus it is hard to use R in a meaningful way, given y and the output of T_y). T_y is injective since f_2 is.

We may conclude that OT and injective trapdoor functions are *incomparable* (and they are both implied by trapdoor permutations). The same construction also implies a separation between trapdoor permutations (i.e. trapdoor functions which are injective and onto), and trapdoor functions which are injective.

Acknowledgments

We thank Shafi Goldwaser for mentioning this open problem and inspiring us to continue working on it. We are grateful to Amos Beimel for many insightful remarks and useful discussions on previous works as well as our own. Finally, we would like to thank anonymous referees for comments on an earlier version of the paper.

References

[AIR00] W. Aiello, Y. Ishai, and O. Reingold. Oblivious commerce: How to sell digital goods. Manuscript in Preperation, 2000.

[BCR86] G. Brassard, C. Crépeau, and J.M. Robert. Information theoretic reductions among disclosure problems. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 168–173, 1986.

[Bei] Amos Beimel. Personal communication.

[BHSV98] M. Bellare, S. Halevi, A. Sahai, and S. Vadhan. Many-to-one trapdoor functions and their relations to public-key cryptosystems. In *Advances in Cryptology – Crypto '98 Proceedings*, Lecture Notes in Computer Science, 1998.

[Blu83] M. Blum. How to exchange (secret) keys. *ACM Transactions of Computer Systems*, 1(2):175–193, 1983. Preliminary version in the Proceedings the ACM Symposium on the Theory of Computing, pages 440–447, 1983.

[BMM99] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *Advances in Cryptology – Crypto '99 Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 80 – 97, 1999.

[CK88] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 42–52, 1988.

[11] As for trapdoor functions with super-polynomial pre-image size, these are implied by (standard) one-way functions [BHSV98], and thus by OT, as well as PKE, and any other assumption that implies OWF.

[Cré87] C. Crépeau. Equivalence between two flavours of oblivious transfers. In *Advances in Cryptology – Crypto '87 Proceedings*, pages 350–354, 1987.

[CS91] C. Crépeau and M. Santha. On the reversibility of oblivious transfer. In *Proceedings of EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 106–113, 1991.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions in Information Theory*, 22(6):644–654, 1976.

[DN00] I. Damgård and J.B. Nielsen. Improved non-commiting encryption schemes based on a general complexity assumption. In *Advances in Cryptology – Crypto '00 Proceedings*, pages 432–450, 2000.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986. Preliminary version in the Proceedings of the IEEE Symposium on the Foundations of Computer Science, pages 464–479, 1984.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer Security*, 28:270–299, 1984. Preliminary version in the Proceedings of the ACM Symposium on the Theory of Computing, pages 365–377, 1982.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game, or: A completness theorem for protocols with honest majority. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proofs. *Journal of the ACM*, 38(3):691–729, July 1991. Preliminary version in the Proceedings of the IEEE Symposium on the Foundations of Computer Science, pages 174–187, 1986.

[Gol97] S. Goldwasser. New directions in cryptography: Twenty some years later. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 314–325, 1997.

[Gol98] O. Goldreich. Secure multi-party computation (working draft). `http://www.wisdom.weizmann.ac.il/~oded/foc.html`, 1998.

[GT00] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, 2000. To Appear.

[HILL99] J. Håstad, R. Impagliazzo, L.A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essencial for complexity based cryptography. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 230–235, 1989.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 44–61, 1989.

[IR00] R. Impagliazzo and S. Rudich. Personal communication.

[JVV86] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[Kil91] J. Kilian. A general completeness theorem for two-party games. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 553–560, 1991.

[Kil00] J. Kilian. More general completeness theorems for secure two-party computation. 2000.

[KKMO00] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM Journal on Computing*, 29(4):1189–1208, 2000.

[KSS00] J. Kahn, M. Saks, and C. 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*, 2000.

[KST99] J.H. Kim, D. Simon, and P. Tetali. Limits on the efficiency of one-way permuation-based hash functions. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 535–542, 1999.

[LR88] M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudo-random functions. *SIAM Journal on Computing*, 17(2):373–386, April 1988. Preliminary version in Proceedings of the ACM Symposium on Theory of Computing, 1986.

[Nao91] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in Advances in Cryptology – Crypto '89 Proceedings, pages 128–136, 1989.

[Nao00] M. Naor. Presentation in DIMACS workshop on Cryptography and Intractibility, March 2000.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[Rab81] M.O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, 1981.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[Rud91] S. Rudich. The use of interaction in public cryptosysytems. In *Advances in Cryptology – Crypto '91 Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 242–251, 1991.

[Sim98] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions. In *Proceedings of EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, 1998.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 80–91, 1982.

[Yao86] A.C. Yao. How to generate and exchange secrets. In *Proceedings of the IEEE Symposium on the Foundations of Computer Science*, pages 162–167, 1986.