On The Impossibility of Basing Identity Based Encryption on Trapdoor Permutations

Dan Boneh* Periklis A. Papakonstantinou† Charles Rackoff†
Yevgeniy Vahlis† Brent Waters‡

Abstract

We ask whether an Identity Based Encryption (IBE) system can be built from simpler public-key primitives. We show that there is no black-box construction of IBE from Trapdoor Permutations (TDP) or even from Chosen Ciphertext Secure Public Key Encryption (CCA-PKE). These black-box separation results are based on an essential property of IBE, namely that an IBE system is able to compress exponentially many public-keys into a short public parameters string.

1. Introduction

Identity-Based Encryption (IBE) is a public key system where any string is a valid public key. For example, user identities (e.g. names or email addresses) can be used as public keys. IBE systems have numerous applications in cryptography (see, e.g., [11, 4, 16, 37, 5, 30, 8, 2, 35] to name a few). Although the concept was introduced in 1984 [34] it took many years until the first practical constructions appeared in 2001 [6, 15]. To date most constructions of Identity Based Encryption are based on bilinear pairings, with the exceptions being quadratic residue based constructions [7, 15], and a lattice based construction [20]. However, none are built from generic cryptographic primitives such as a trapdoor permutation (TDP) and chosen ciphertext secure public key encryption (CCA-PKE).

In this paper we ask whether a secure IBE can be built from generic primitives using a black-box construction and without any further assumptions. We ask for a modest goal of constructing a semantically secure IBE system for encrypting a single bit message. Moreover, the IBE need only be weakly secure: we only require security against an adversary who non-adaptively requests private keys for identities of its choice before seeing the public parameters, and then tries to break semantic security for some other identity. We review the security model in more detail in the next section. Clearly, if one cannot construct an IBE scheme secure in this weak model, then one cannot construct a more powerful system that encrypts multiple bits and is secure under adaptive and chosen-ciphertext attacks.

The primitives that we consider are trapdoor permutations and chosen ciphertext secure public key encryption. Following the approach of Impagliazzo and Rudich [27] we describe an adversary that given access to some oracle, relative to which the above primitives exists, breaks every IBE with high probability.

Our results. We show that a Weakly Semantically Secure IBE (WSS-IBE) cannot be constructed from a TDP, or even a CCA-PKE using black-box constructions. More precisely, we use the methodology introduced by Impagliazzo and Rudich [27] to prove the following black-box separation results (stated informally):

- There exists no black-box reduction that from a given trapdoor permutation (g, π, ι) constructs a semantically secure IBE $\left(G^{(g,\pi,\iota)},\ K^{(g,\pi,\iota)},\ E^{(g,\pi,\iota)},\ D^{(g,\pi,\iota)}\right)$ for one bit messages.

- There exists no black-box reduction that from a given multi-bit CCA-secure public key encryption (g, e, d) constructs a semantically secure IBE $\left(G^{(g,e,d)},\ K^{(g,e,d)},\ E^{(g,e,d)},\ D^{(g,e,d)}\right)$ for one bit messages.

We refer to [32, 27, 23] for an excellent discussion of how to interpret such black-box separation results. In the language of [32], our results rule out *fully* black box constructions. Black-box separation has previously been used to bound the efficiency of constructing one primitive from another [28, 19, 18] as well as prove the infeasibility (efficient or not) of reductions [27, 36, 22, 24, 23].

*Stanford University. Supported by NSF and the Packard Foundation.
†University of Toronto. Supported by NSERC.
‡SRI International. Supported by NSF CNS-0749931, CNS-0524252, CNS-0716199; the U.S. Army Research Office under the CyberTA Grant No. W911NF-06-1-0316; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001.

Our separation proof builds an oracle $\mathcal{O}$ relative to which TDP and CCA-PKE exist. We then show that all candidate WSS-IBE systems $(G^{\mathcal{O}}, K^{\mathcal{O}}, E^{\mathcal{O}}, D^{\mathcal{O}})$ are insecure against a polynomial time adversary $\mathcal{A}$ with access to $\mathcal{O}$ and to a **PSPACE** oracle. The oracle $\mathcal{O}$ is very natural and directly implies the existence of trapdoor permutations and CCA-secure encryption. The difficult part of the proof is constructing the adversary $\mathcal{A}$ that breaks any given candidate SS-IBE system relative to this oracle.

Our proof brings to light the essential property of IBE systems: an IBE system creates an exponential number of public keys (identities) and compresses all of them into a short string called the public parameters. This ability to represent exponentially many public keys using a short string is not possible with a generic PKE or TDP. This exponential compression is achieved in existing IBE constructions using algebraic properties.

We briefly note that the results of Gertner et al. [22] can be adapted to prove that semi-honest oblivious transfer (OT) protocols cannot be black-box constructed from an WSS-IBE (and even SS-IBE). Since TDPs imply OT, we obtain the following picture:

$$\text{CCA-PKE} \quad \not\Leftarrow \quad \text{SS-IBE} \quad \not\Leftarrow \quad \text{OT}$$

Relation to other black-box separation efforts. Gertner et al. [23] recently made an important step toward a black-box separation between CCA-PKE and SS-PKE. Suppose the results of Gertner et al. will eventually be strengthened to a full black-box separation of CCA-PKE and SS-PKE. Consider also the result of Canetti et al. [12] that gives a black-box construction for CCA-PKE from any SS-IBE. Putting together the construction of [12] with the (yet unknown) separation of CCA-PKE and SS-PKE leads to a black-box separation between SS-PKE and SS-IBE. Our separation result, in comparison, is stronger since it separates CCA-PKE from SS-IBE. Hence, we obtain a stronger separation than what can be achieved by an eventual full separation of CCA-PKE and SS-PKE.

The limits of black box separation results. Although most constructions of encryption schemes in cryptography are black box, there are several important constructions that use the underlying primitive in a non-black-box manner. A classic example is the transformation of any enhanced TDP to a CCA-PKE [17, 29]. Similarly, it is important to keep in mind that our seperation results do not rule out the existence of a non-black box construction of IBE from TDP.

2. Our oracles

We prove our theorem in a relativized model where all algorithms have access to three random oracles g, e, d that roughly correspond to key generation, encryption, and decryption oracles. We use $\mathcal{O} = (g, e, d)$ to refer to the triple of oracles. For every $\lambda \in \mathbb{N}$, the oracles g, e, d are sampled uniformly at random from the set of all functions satisfying the following conditions:

- $g : \{0,1\}^\lambda \to \{0,1\}^\lambda$. We view g as taking a secret key sk as input and outputting a public key.
- $e : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a function that on input $\text{pk} \in \{0,1\}^\lambda$ and $x \in \{0,1\}^\lambda$ outputs $e(\text{pk}, x) \in \{0,1\}^\lambda$. We require that for every $\text{pk} \in \{0,1\}^\lambda$ the function $e(\text{pk}, \cdot)$ be a permutation of $\{0,1\}^\lambda$.
- $d : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a function that on input $\text{sk} \in \{0,1\}^\lambda$ and $y \in \{0,1\}^\lambda$ outputs an $x \in \{0,1\}^\lambda$ that is the (unique) pre-image of y under the permutation defined by the function $e(g(\text{sk}), \cdot)$.

Partial oracles. Our attack algorithm will generate during its computation trapdoor permutation oracles that are defined only on a subset of the possible inputs. We call such functions partial trapdoor permutation oracles. The following definition describes when such a partial oracle is extendable to a full oracle:

Definition 2.1. Let $\mathcal{O}' = (g', e', d')$ be a function which is defined on part of the domain of trapdoor permutation oracles. $\mathcal{O}'$ is called a *consistent partial oracle* if it has all the following properties:

(i) For every $\text{pk} \in \{0,1\}^\lambda$, $e'(\text{pk}, \cdot)$ is 1-1; (ii) for every $\text{sk} \in \{0,1\}^\lambda$, $d'(\text{sk}, \cdot)$ is 1-1; (iii) for every sk such that $g'(\text{sk})$ is defined let $\text{pk} = g'(\text{sk})$. We require that for every $x \in \{0,1\}^\lambda$, $e'(\text{pk}, x) = \alpha$ is defined if and only if $d'(\text{sk}, \alpha) = x$ is defined.

Building Trapdoor Permutations and CCA Secure PKE. We begin by claiming that using the oracles above we can build both trapdoor permutations and CCA-secure encryption.

Claim 2.2. *There exists a trapdoor permutation scheme such that for all oracle adversary's $\mathcal{A}$ that make at most a polynomial number of queries to the oracle $\mathcal{O}$ we have that for sufficiently large λ the adversary's success in breaking the trapdoor permutation is negligible in λ, where λ is the security parameter of the scheme.*

Claim 2.3. *There exists an encryption scheme such that for all oracle adversary's $\mathcal{A}$ that make at most a polynomial number of queries to the oracle $\mathcal{O}$ we have that for sufficiently large λ the adversary's advantage in the chosen ciphertext security game is negligible in λ, where λ is the security parameter of the scheme.*

The construction of trapdoor permutations follows directly from the definitions of the oracles. While it is possible to build CCA-secure encryption from trapdoor permutations [17], current techniques do this in a non-black-box manner and therefore are not applicable in this setting. However, since we can employ the encryption function e as a random oracle for a given pk, we can build a CCA-secure encryption scheme using known techniques in the random oracle model [3][1]. Claims 2.2 and 2.3 remain true even if $\mathcal{O}$ includes a **PSPACE** complete oracle [19].

3. Identity Based Encryption in the Presence of Oracles

Recall that an IBE system consists of four algorithms [34, 6] (G, K, E, D) parameterized by a security parameter λ. In our case these algorithms have access to the oracles g, e, d described in Section 2. We define three important parameters associated with this system:

- We let $q = q(\lambda)$ be the maximum number of oracle calls that these algorithms make in a single execution.
- We let $\varepsilon \in (1/2, 1]$ be the IBE's level of correctness, that is, a ciphertext generated from the correct distribution will be successfully decrypted with probability $\geq \varepsilon$ (as defined in (1)).
- We let n be the size of various IBE parameters, as described below.

The functionality of each algorithm is as follows: algorithm $G(\text{MSK})$ takes as input a master secret key $\text{MSK} \in \{0,1\}^n$; it outputs public parameters $\text{PP} \in \{0,1\}^n$. Algorithm $K(\text{MSK}, \text{ID})$ is deterministic, and it takes as input the master secret key MSK, and an identity $\text{ID} \in \{0,1\}^n$. It outputs the private key $\text{SK}_{\text{ID}} \in \{0,1\}^n$ for identity ID. The encryption algorithm $E(\text{PP}, \text{ID}, b, r)$ encrypts a bit b for a given identity ID using public parameters PP and randomness $r \in \{0,1\}^n$; it outputs a ciphertext $C \in \{0,1\}^n$. The decryption algorithm $D(\text{SK}_{\text{ID}}, C)$ decrypts a ciphertext C using the private key SK_{ID} and is deterministic. Note that making the algorithm K deterministic does not restrict the type of IBE constructions that our results cover since any secure IBE with a randomized K can be converted into a secure IBE with a deterministic K by applying a pseudorandom function to the identity in order to obtain (pseudo) random bits for K.

[1] Some care needs to be taken to deal with the fact that the oracle gives a permutation and not a function.

An IBE system must satisfy the following correctness property: for every $\text{ID} \in \{0,1\}^n$, $b \in \{0,1\}$

$$\Pr\left[\begin{array}{l} \text{PP} \leftarrow G(\text{MSK}), \\ \text{SK}_{\text{ID}} \leftarrow K(\text{MSK}, \text{ID}), \quad ; \; D(\text{SK}_{\text{ID}}, C) = b \\ C \leftarrow E(\text{PP}, \text{ID}, b) \end{array}\right] \geq \varepsilon \tag{1}$$

where the probability is over MSK, and the implicit randomness of the encryption algorithm.

One might notice that we treat the correctness parameter ε of an IBE algorithm as a constant. We might also consider IBE systems where the correctness parameter is a function of the security parameter; in particular the correctness guarantees might degrade with λ. However, by using standard amplification techniques we can transform any IBE system with correctness parameter that is non-negligibly greater than $\frac{1}{2}$ into one that is correct with probability ε for $\varepsilon < 1$.

Normal Form. We restrict our attention (without loss of generality) to IBEs that satisfy the following condition: each oracle query of the form $d(\text{sk}, \cdot)$ must be followed by the query $g(\text{sk})$. We use this condition to avoid situations in which the system uses a secret key without ever requesting the public key.

Definition of Security. For security we use a weakened version of the standard IBE semantic security defined in [6]. Weak semantic security is defined using a game in which the attacker starts by choosing a polynomially long sequence of identities for which it wishes to obtain private keys, along with the challenge identity that it is going to try and break. This is done even before the adversary is given the public parameters. The adversary is then given the public parameters, a sequence of private keys for the above identities (except the challenge identity), and an encryption of a random bit for the challenge identity using the public parameters.

More precisely, weak semantic IBE security is defined using a game between a challenger and an adversary $\mathcal{A}$ in which both parties are given a security parameter λ is input. The game proceeds as follows:

Setup: The adversary submits a challenge identity ID_*, and a sequence of identities $\text{ID}_1, \ldots, \text{ID}_l$ where $l = l(q)$ and $l(\cdot)$ is some polynomial.

Private Keys: The challenger chooses at random $\text{MSK} \in \{0,1\}^n$. It then computes $\text{PP} \leftarrow G(\text{MSK})$, and $\text{SK}_i \leftarrow K(1^\lambda, \text{MSK}, \text{ID}_i)$ for $1 \leq i \leq l$. The tuple $(\text{PP}, \text{SK}_1, \ldots, \text{SK}_l)$ is given to the adversary.

Challenge: The challenger chooses at random $r \in \{0,1\}^n$, $b \in \{0,1\}$. It then computes $C_* \leftarrow E(\text{PP}, \text{ID}, b; r)$, and C_* is given to the adversary.

Guess: The adversary outputs a guess $b' \in \{0,1\}$, and wins if $b' = b$.

We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}$ as

$$\mathrm{Adv}_{\mathcal{E},\mathcal{A}} \stackrel{def}{=} \left| \Pr[b = b'] - \frac{1}{2} \right|$$

The probability is over the random bits used by the challenger and the adversary.

4. Separating IBE from Trapdoor Permutations

In this section we describe an attack algorithm that breaks every IBE in the trapdoor permutation model. Our attack requires only a polynomial number of queries to the oracles, and can be implemented in polynomial time if we add a **PSPACE** complete oracle. We start with a high level overview of the attack.

The algorithm G in an IBE takes a random private MSK and outputs a public key PP. During its computation G may make use of its trapdoor permutation oracles, and use the results of the queries to construct the final public key. In our model any algorithm G that aims to be secure will have to make "essential" use of its oracles. Otherwise, an adversary could simply invert the function computed by G (using the **PSPACE** oracle), thus obtaining a valid private key. Looking ahead to our analysis, we show that G has to make use of the trapdoor generation oracle g by generating several public keys of trapdoor permutations, and embedding them in the public parameters of the IBE. After the IBE public parameters are fixed the algorithm K may include in each individual private key of an identity one or more trapdoors for the permutation public keys that were embedded in the IBE public parameters. We argue that using the permutation public keys that are embedded in the IBE public parameters is the only way to encrypt securely. Thus, if the adversary could collect all the important[2] trapdoors for the embedded public keys it would be able to decrypt any ciphertext.

It is precisely this task of discovering the important trapdoors that is made possible due to the fact that the public parameters of an IBE encode an exponential number of public keys, and at the same time contain only a polynomial number of embedded permutation public keys. Suppose that a certain permutation, for which the public key is embedded in the public parameters, is used in a significant way to encrypt a bit to some identity. Then, for the decryption to work the corresponding trapdoor must be embedded in that identity's private key.

Suppose there are q permutation public keys embedded in the public parameters. Now consider a set T of (identity,secret key) pairs for which each identity's secret key has a trapdoor embedded in it such that the trapdoor is not present in any other identity's secret key in T. It follows that T can contain at most q identities. Thus, for every set of identities that is significantly bigger than q only a small portion of the identities have private keys that reveal new important trapdoors. This immediately yields the following approach for an attack: let S be a sufficiently large set of identities (say of size q^c). We select a challenge identity at random from S and ask to see the private keys for all the other identities in S. With probability at least $1 - 1/q^{c-1}$ the private keys that we obtained contain all the important trapdoors that are needed to decrypt ciphertexts that were addressed to any identity in S, including the challenge identity.

While the above argument gives an intuitive sketch for an attack algorithm, the actual algorithm and proof contain several challenging subtleties. The most interesting challenges arise from the fact that the public parameters and IBE private keys may be encoded in an arbitrary manner. In particular, an attack algorithm that simply "looks" at an IBE private key will not necessarily be able to tell which oracle secret keys are embedded in it. We resolve this issue by encrypting a random bit to that identity, and then decrypting the resulting ciphertext. We record all trapdoors that are actually used in queries to the permutation inverse oracle d during the computation of the encryption and decryption algorithms.

The fact that we discover important trapdoors by observing a sample computation of the encryption and decryption algorithms for each identity, rather than "reverse engineering" the private keys, leads to the main technical complication in our attack. Since we discover trapdoors only experimentally, it is crucial that all the values that appear during the decryption of the challenge ciphertext are distributed in a manner as they would be if the correct private key and the actual oracle were used for the decryption. Otherwise, if the decryption process for the challenge identity looks significantly different from the decryption for the other identities then new trapdoors may be required to decrypt the challenge ciphertext.

We are now ready to state our main theorem:

Theorem 4.1. *Let $1/2 < \delta < \varepsilon \leq 1$. Then, there exists a constant c such that for every $q \in \mathbb{N}$, and every IBE which makes at most q queries to the oracle and has correctness parameter $\geq \varepsilon$, there exists an adversary $\mathcal{A}$ that makes at most q^c queries to the oracle and decrypts the challenge ciphertext successfully with probability $\geq \delta$.*

[2]Some trapdoor public keys may be generated by G and embedded in PP but never used (in a significant way) by the encryption algorithm. For these public keys there may be no way of recovering the trapdoors with a small number of queries. However, it is also unnecessary as the adversary will never need these trapdoors to decrypt.

The theorem shows that we can break any IBE in the trapdoor permutation model with probability which is as close to the correctness guarantee ε as we wish. For notational simplicity we will occasionally refer to the algorithms without using the $\mathcal{O}$ superscripts.

4.1. The Attack Algorithm

Let $1/2 < \delta < \varepsilon \leq 1$, and $q, n \in \mathbb{N}$. We now describe the attack algorithm $\mathcal{A}$, which plays the Identity-Based Encryption security game with a challenger $\mathcal{C}$ for a given IBE system $IBE = (G, K, E, D)$ which makes at most q queries, has inputs of length n, and has correctness guarantee $\geq \varepsilon$. $\mathcal{A}$ wins with probability $\geq \delta$.

For here on we let e denote the basis of the natural logarithm. The adversary starts by choosing five constants c_1, c_2, c_3, c_4, c_5 such that $c_4 \geq c_1 + 2$ and

$$\frac{2}{q^{c_3-c_1-1}} + \frac{3}{eq^{c_2-c_4}} + \frac{2}{q^{c_1-1}} + \frac{10}{q^{c_5-(c_1+c_2+c_3+c_4+2)}} \leq \frac{\varepsilon-\delta}{2}$$

4.1.1 Degenerate case

We note that it is okay to treat ε as a constant (see Section 3). Furthermore, we can assume that $q \geq 2$ and use this to choose constants that will be guaranteed to satisfy the conditions.

Our general adversary does not deal well with the following IBEs due to the small size of the probability spaces in question, therefore, we break such IBEs separately.

If $2^\lambda \leq q^{c_5}$ then the adversary proceeds as follows:

1. The adversary queries the oracles g, e, d on all 2^λ inputs. After this step the adversary has completely learned the oracle $\mathcal{O}$. Thus, from this point on the adversary will not make any additional oracle queries.

2. The adversary then chooses an arbitrary challenge identity $\text{ID}_* = \bar{0}$ and asks to obtain an encryption C_* of a random bit to ID_*. The adversary is given PP and C_*.

3. The adversary randomly chooses MSK' such that $G^{\mathcal{O}}(\text{MSK}') = \text{PP}$, and computes $\text{SK}_* = K^{\mathcal{O}}(\text{MSK}', \text{ID}_*)$.

4. Finally, the adversary computes and outputs $D^{\mathcal{O}}(\text{SK}_*, C_*)$.

Clearly in the degenerate case the adversary successfully decrypts the ciphertext with probability ε.

4.1.2 The general case

If $2^\lambda > q^{c_5}$ our adversary performs the following steps.

Initialization. The adversary starts by initializing a table L, which will be used to store information that the adversary learns about the oracle $\mathcal{O}$. More precisely, L is a set of tuples of the form (α, β) where α specifies an oracle from $\{g, e, d\}$ and the query to that oracle, and β is the reply that was given to that query.

From this point on, *every query* that the adversary makes to the oracle $\mathcal{O}$, and the answer to that query are recorded in L.

Setup. Let $S \subseteq \{0,1\}^*$ be the set containing the binary encodings of the integers $1, \ldots, q^{c_1}$. The adversary picks uniformly at random $\text{ID}_* \in_R S$, submits ID_* as the challenge identity and requests to see private keys for all $\text{ID} \neq \text{ID}_*$, $\text{ID} \in S$.

Challenge. The adversary is given a tuple $(\text{PP}, \text{SK}_1, \ldots, \text{SK}_{\text{ID}_*-1}, \text{SK}_{\text{ID}_*+1}, \ldots, \text{SK}_{q^{c_1}}, C_*)$. From now on the adversary will make use of the public parameters PP and the private keys SK_i, but the challenge ciphertext C_* will only be used in the last stage of the attack.

Step 1: Discovering important trapdoors. For each $\text{ID} \neq \text{ID}_*$, $\text{ID} \in S$, the adversary chooses at random $r \in_R \{0,1\}^n$, $b \in_R \{0,1\}$. Then, the adversary computes $C \leftarrow E^{\mathcal{O}}(\text{PP}, \text{ID}, b; r)$ and $D^{\mathcal{O}}(\text{SK}_{\text{ID}}, C)$. Notice that during this step the adversary will add to L, among other things, mappings of the form $g(\text{sk}) = \text{pk}$. Whenever such a mapping is added to L we can invert the permutation pk successfully.

Step 2: Discovering frequent queries. The adversary repeats the following q^{c_2} times: chooses at random $r \in_R \{0,1\}^n$, $b \in_R \{0,1\}$, and computes $E^{\mathcal{O}}(\text{PP}, \text{ID}_*, b; r)$.

Step 3: Discovering secret queries and decrypting the challenge. The adversary chooses at random $1 \leq k \leq q^{c_3}$ and repeats the following k times.

1. **(Offline phase)** The adversary chooses uniformly at random a global secret key MSK', and a partial oracle $\mathcal{O}' = (g', e', d')$ that satisfy the following properties: $\mathcal{O}' \setminus L$ is of size[3] at most q^{c_4}; $L \subseteq \mathcal{O}'$, that is, $\mathcal{O}'$ contains the partial information that the adversary has learned about $\mathcal{O}$; $G^{\mathcal{O}'}(\text{MSK}') = \text{PP}$; For every $\text{ID} \neq \text{ID}_*$, $\text{ID} \in S$ it holds that $K^{\mathcal{O}'}(\text{MSK}', \text{ID}) = \text{SK}_{\text{ID}}$; and $K^{\mathcal{O}'}(\text{MSK}', \text{ID}_*)$ is defined[4].

   We also require that $\mathcal{O}'$ is a *consistent partial oracle* according to Definition 2.1. Note that if were to implement our adversary as a PPT algorithm we would

[3] That is, the number of mappings that were invented by the adversary, and did not appear in L, is at most q^{c_4}.

[4] By defined we mean that when K is run on global secret key MSK', $\mathcal{O}'$ is sufficient to answer all of K's queries. Note that c_4 is large enough to allow $\mathcal{O}'$ to contain all the offline values that are needed to satisfy this condition.

use the **PSPACE** complete oracle to implement the offline phase. In fact, this is the only place where the **PSPACE** oracle is needed.

2. Using the partial oracle $\mathcal{O}'$ the adversary computes $\text{SK}'_* \leftarrow K^{\mathcal{O}'}(\text{MSK}', \text{ID}_*)$.

3. **(Online phase)** The adversary chooses at random $r \in_R \{0,1\}^n$, $b \in_R \{0,1\}$, and computes $C \leftarrow E^{\mathcal{O}}(\text{PP}, \text{ID}_*, b; r)$. Notice that here we are using the actual oracle $\mathcal{O}$.

4. The adversary combines the partial oracle $\mathcal{O}'$ and the actual oracle $\mathcal{O}$ into a new oracle $\mathcal{O}''$. The precise description of $\mathcal{O}''$ is given in the next paragraph. The adversary then simulates $D^{\mathcal{O}''}(\text{SK}'_*, C)$. To decrypt the challenge ciphertext the adversary replaces C with C_* at the last repetition of this step. That is, at repetition k the adversary simulates $D^{\mathcal{O}''}(\text{SK}'_*, C_*)$ and obtains a bit b'.

Guess. The adversary outputs the bit b' obtained in the last repetition of the previous stage.

4.1.3 The oracle $\mathcal{O}''$

The oracle $\mathcal{O}''$ is constructed by the adversary to create a single oracle which combines the offline generated values of $\mathcal{O}'$ with the actual oracle $\mathcal{O}$. Note that we cannot avoid using the oracle $\mathcal{O}$ in the decryption since it was used to encrypt the challenge ciphertext.

To describe $\mathcal{O}''$ we need to treat specially a certain type of permutation. We call a trapdoor permutation which is generated by K for some ID, but not by G, an "internal trapdoor permutation". Intuitively, the internal permutations are not accessible to the encryptor and therefore we do not have to use the actual oracle to answer queries about them. Formally,

Definition 4.2. Let $\mathcal{O}'$ be a partial oracle, and $\text{MSK}' \in \{0,1\}^n$. We define

1. U to be the set of pk such that $g(\text{sk})$ is asked during $G^{\mathcal{O}'}(\text{MSK}')$ and received answer pk.

2. V be the set of pk such that $g(\text{sk})$ is asked during $K^{\mathcal{O}'}(\text{MSK}', \text{ID})$ for some $\text{ID} \in S$ and received answer pk.

3. L_g be the set of pk such that there exists a mapping of the form[5] $[g(\cdot) = \text{pk}]$ in L.

The set of internal trapdoor permutations is $W = V \setminus U$.

[5] We write $[g(\cdot) = \text{pk}] \in L$ to denote that there exists an sk such that $[g(\text{sk}) = \text{pk}] \in L$.

The oracle. The oracle $\mathcal{O}''$ works on a query α as follows:

1. If $\mathcal{O}'(\alpha)$ is defined then reply with $\mathcal{O}'(\alpha)$.

2. Else if α is a query of the form $e(\text{pk}, \cdot)$ (or $d(\text{sk}, \cdot)$) where $\text{pk} \in W \setminus L_g$ (or there exists $\text{pk} \in W \setminus L_g$ such that $[g(\text{sk}) = \text{pk}] \in \mathcal{O}'$) then $\mathcal{O}''$ generates the answer randomly and records it in $\mathcal{O}'$. More precisely, if α is of the form $e''(\text{pk}, x)$ where pk is an internal trapdoor permutation, or of the form $d''(\text{sk}, \alpha)$ where $g'(\text{sk})$ is an internal trapdoor permutation, then $\mathcal{O}''$ emulates a random permutation independent of the actual oracle $\mathcal{O}$. Namely, it responds randomly (and consistently), ensuring that $e''(\text{pk}, \cdot)$ is a permutation and $d''(\text{sk}, \cdot)$ is its inverse, where $g'(\text{sk}) = \text{pk}$. By consistently we mean that any new generated value α (say $e''(\text{pk}, x) = \alpha$) is recorded by adding the entries $e'(\text{pk}, x) = \alpha$ and $d'(\text{sk}, \alpha) = x$ to $\mathcal{O}'$.

3. Else do the following:

   (a) If α is a query of the form $d(\text{sk}, y)$ such that there exists pk such that $[g(\text{sk}) = \text{pk}] \in \mathcal{O}'$ and there exists sk' such that $[g(\text{sk}') = \text{pk}] \in L$ then rewrite α as $d(\text{sk}', y)$. This is to ensure that if we have a correct trapdoor for pk we use it, and not some other incorrect trapdoor that we invented during the offline phase.

   (b) Get the answer from the actual oracle: $\beta \leftarrow \mathcal{O}(\alpha)$, and return β (recall that $[\mathcal{O}(\alpha) = \beta]$ is added to L).

This concludes the description of our adversary. The following lemma directly implies our theorem:

Lemma 4.3. *When $2^\lambda > q^{c_5}$ our adversary correctly decrypts the challenge ciphertext with probability at least $\varepsilon - \left(\frac{2}{q^{c_3-c_1-1}} + \frac{3}{eq^{c_2-c_4}} + \frac{2}{q^{c_1-1}} + \frac{10q^{c_1+c_2+c_3+c_4+2}}{2^\lambda}\right)$.*

Due to lack of space the proof of Lemma 4.3 appears in the full version. In the following section we give an overview of the technical difficulties that each of the adversary's steps deals with.

5. Proof Overview

In this section we give a high level overview of the proof of Lemma 4.3, and highlight the main issues that cause the complex description of our adversary. Below we describe three main challengess that our adversary has to deal with in order to successfully simulate the decryption of the challenge ciphertext. Each of the three steps of the adversary deals with one of the problems described below.

Problem 1. The first and somewhat isolated issue is the fact that the challenge ciphertext is computed using the actual oracle while the adversary simulates the decryption using a hybrid between the actual oracle and the small partial oracle $\mathcal{O}'$ which was "invented" offline.

Suppose that for some query α which is asked during the computation of the challenge ciphertext we have that *both* $\mathcal{O}'(\alpha)$ is defined and $\mathcal{O}'(\alpha) \neq \mathcal{O}(\alpha)$. Then, the simulated decryption algorithm may discover this fact, and it will refuse to decrypt. This type of inconsistency occurs with small probability due to step 2 "discovering frequent queries" of the adversary. During this step the adversary repeatedly encrypts a random bit to the challenge identity, and records the oracle queries and answers that appear during this computation. Since this is repeated many times the adversary discovers the portions of the oracle that are frequently accessed during such encryptions. The oracle $\mathcal{O}'$ is then chosen in a manner that is consistent with all the information that the adversary has learned about the actual oracle, therefore $\mathcal{O}'$ will be consistent with all the frequent queries and answers that may appear during the computation of the challenge. $\mathcal{O}'$ may contain entries which are not consistent with the actual oracle. However, these queries appear infrequently during encryptions, and so are unlikely to appear during the computation of the challenge ciphertext.

Problem 2. The second issue is that the adversary may be unlucky in its of choice of the challenge identity ID_*. For example, consider a degenerate IBE which securely encrypts messages to one fixed identity and sends the plaintext for every other identity. If the adversary accidentally picks that identity to be challenged on, it will not succeed in decrypting the challenge ciphertext. More generally, our adversary may select an ID_* for which there is a trapdoor that is needed with high probability during decryption, and that trapdoor appears only with low (or zero) probability when encrypting and decrypting ciphertexts for the other identities. In this case, this crucial trapdoor will not be discovered during step 1 ("discovering important trapdoors"), and the adversary will fail to decrypt the challenge ciphertext. adversary simulates the decryption

However, the adversary will choose an unlucky challenge identity with only a small probability. If we look at the computation of an encryption and decryption of a random bit for q^{c_1} identities then at most q of these computations may reveal new trapdoors that are for trapdoor permutations that are embedded in the public parameters and that are not present in any other secret key. If an identity is chosen at random then with probability $1 - q/q^{c_1}$ its computation will not contain any trapdoors which do not also appear in the computations of the other identities.

Problem 3 (main technical difficulty). The final, and most complex issue is the accuracy of the adversary's simulation of the decryption of the challenge ciphertext. The key technical difficulty is that when a new trapdoor is not needed, the adversary still does not perfectly simulate decryption. However, we will show that the adversary can do a very good simulation, which is sufficient for our purpose. The decryption of the challenge ciphertext is simulated using the hybrid oracle $\mathcal{O}''$ which is a combination of the partial oracle $\mathcal{O}'$ that was chosen offline, and the actual oracle $\mathcal{O}$. The reason that the adversary's simulation is not perfect is that the partial oracle $\mathcal{O}'$ is chosen to be consistent with the adversary's knowledge of the actual oracle *before* it starts simulating the decryption. New points of the actual oracle that are revealed during the simulation may reveal that the oracle $\mathcal{O}'$ was not chosen from the right distribution, and cause the decryption algorithm to misbehave. This is a very subtle point in the analysis, and deserves more discussion. Consider all the oracle queries that are asked before the adversary chooses its final partial oracle $\mathcal{O}'$. Most of these queries were asked by the adversary, and therefore appear in L. However, some of the queries were asked by the challenger while it computed the private keys for identities $\mathrm{ID} \neq \mathrm{ID}_*$ and the public parameters. Although the private keys and the public parameters are given to the adversary, the queries that were asked while computing them are not. Thus, the adversary obtains some partial information (encoded in the public parameters and private keys) about the queries that were asked.

Consider the following simplified scenario: suppose that the public parameters PP contain the answer to one of two queries. That is, PP contains either $O(\alpha)$ or $O(\alpha')$, for some $\alpha \neq \alpha'$, each with probability $1/2$. Now, suppose that in a given run of the experiment PP contains $O(\alpha) = \beta$ but the adversary guesses that it contains $O(\alpha')$, and sets $\mathcal{O}'(\alpha') = \beta$. This is fine as long as once $\mathcal{O}'$ is fixed, the query $O(\alpha)$ will not be asked. Otherwise, if $O(\alpha)$ is asked then in the adversary's simulated world we get $O(\alpha) = O(\alpha') = \beta$. This is an unlikely event that can be used by D to abort decryption when run relative to $\mathcal{O}''$. Moreover, the adversary will guess that query input wrongly half the time.

To avoid the situation described above we have the repetitions of step 3. Due to step 3, once the query $O(\alpha)$ is asked, and the mapping $O(\alpha) = \beta$ is added to L, the adversary is very unlikely to set $O(\alpha') = \beta$. The total number of queries that are asked by the challenger and not seen by the adversary (and embedded in some meaningful way) is at most q^{c_1+1} (q queries for each private key, and for the public parameters), thus if we repeat step 3 sufficiently many times before decrypting the challenge ciphertext we can be assured that the above situation will not arise. Unfortunately, we do not know when one of the hidden queries is discov-

ered. We solve this by repeating step 3 a random number of times, and at the last repetition we decrypt the challenge ciphertext. This can be thought of as repeating step 3 q^{c_3} times, and plugging the challenge ciphertext at a random repetition. There can be at most q^{c_1+1} repetitions during which we discover new hidden queries, and so the probability that we try to decrypt the challenge in one of these repetitions is at most q^{c_1+1}/q^{c_3}.

We have described three problems that may cause our adversary to fail, and our way of dealing with each of the problems. This concludes the intuitive overview of our proof. Formally, we describe four experiments where the first experiment is the IBE security game played with our adversary, and the last experiment does not involve an adversary, and uses the correct oracle and private key to decrypt the message. We show that the distributions on the transcripts of the experiments are close to each other (in some sense), and thus conclude that the adversary decrypts the challenge ciphertext with probability which is close to the correctness guarantee ε of the IBE.

6. Extensions and Open Directions

We discuss possible extensions to these Identity-Based Encryption separation result. In addition, we look at new directions in separating different primitives that belong in a class of functionality that we call *functional encryption*.

6.1. Extending Our IBE Separation Results

A simple way to extend our results is to look for other oracles that provide straightforward capabilities to realize other schemes, but that still do not provide a compression mechanism for public keys and therefore allow for an analogous separation argument. For example, one might try to extend our results to separate IBE from doubly homomorphic encryption, a rather powerful (and yet unrealized) primitive. This separation rules out black-box IBE constructions from number-theoretic encryption systems, such as the Paillier [31] system.

We also remark that our separation results provide separation from an even weaker form of IBE, known as selectively secure IBE [11]. In a selectively secure system an attacker commits to the identity, ID^*, to attack before he sees the system parameters. Our attacker $\mathcal{A}$ chooses his attack identity independent of the IBE public parameters, so it is straightforward to argue that such an attack would also work in the selective-ID game.

6.2. Building Primitives from IBE

In this paper we focused on the (im)possibility of building IBE in a black-box manner from other primitives. However, it is also interesting to think about the opposite direction in terms of what primitives can be built from IBE. At first glance, IBE might appear to be a very strong primitive. For instance, Canetti, Halevi, and Katz [13] showed how to build a CCA-secure cryptosystem from any IBE scheme.

At the same time, Identity-Based Encryption cannot be used to build Oblivious Transfer in a black-box manner. It is relatively straightforward to adapt the techniques of Gertner et. al. [22] to build an oracle that yields an IBE scheme, but for which no OT scheme exists. Their result follows from the fact that any OT sender (using their oracles) can make a list of the intersection of knowledge between the sender and receiver. The main addition necessary for separation from IBE is that the sender will need to include in the intersection list the identity/public parameter pairs for which he knows the master key, but the receiver has the private key.

6.3. Relationships Among Primitives in Functional Encryption

Since the introduction of the first Identity-Based Encryption constructions, the cryptographic community has introduced many variants of IBE, such an anonymous IBE and hierarchical IBE. One interesting future direction is to explore the relationships among these primitives.

We begin by observing that Identity-Based Encryption can serve as an articulation point for a much richer concept of encryption that we call *functional encryption* systems. Ideally, in a functional encryption system an authority can create a pair of public parameters PP and master secret key MSK. Any encryptor in the system can create an encryption of a string x as $\mathrm{CT} = E_{\mathrm{PP}}(x)$. The authority with the master secret key will be able to create a private functionality SK_f for any (polytime-computable) function f. If a user applies SK_f to a ciphertext $E_{\mathrm{PP}}(x)$ he will learn the value of $f(x)$ and nothing more.

We emphasize two points about functional encryption systems. The first is that we make no attempt to hide the description of f from a user that possesses SK_f— otherwise this might imply something closer to obfuscation for which impossibility results are known. Second, unlike secure multi-party computation or other interactive protocols, there are no known general constructions to realize functional encryption systems for any polynomial time functions. However, we argue that introducing the concept is useful both as a context to view existing work and as an ideal goal on which to base future work. In Figure 1 we show known relationships between existing primitives that represent progress in the direction of functional encryption systems. In particular, the diagram shows known black-box reductions between Identity-Based Encryption (IBE), Attribute-Based Encryption [33, 25, 14], Hierarchical Identity-Based Encryption [26, 21], (Hierarchical)

Anonymous Identity-Based Encryption [5, 10, 1], and primitives for searching on encrypted data [9, 35].

It is an open problem to provide black-box separations between the boxes in Figure 1.

7. Conclusions

We showed that there cannot exist a (fully) black-box reduction from Identity-Based Encryption (IBE) to Trapdoor Permutations (TDPs) or even to Chosen Ciphertext Secure Public Key Encryption. Our separation proof constructs an oracle relative to which TDPs and CCA-secure encryption exist, yet there exists an attack on any IBE scheme. Furthermore this attack can be implemented in probabilistic polynomial time when given access to a $\mathbf{PSPACE}$ oracle. Our methods thus capture the intuition that it is difficult to generically realize the public-key compression property of IBE systems.

Acknowledgements. We thank Chris Peikert for valuable comments and suggestions. We thank Ali Juma, and Hamed Hatami for their insightful suggestions, and Siavosh Benabbas for helping review an earlier version of this manuscript.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBExs, and extensions. In *CRYPTO*, pages 205–222, 2005.

[2] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[4] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. of Computing (SICOMP)*, 36(5):915–942, 2006.

[5] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Eurocrypt '04*, 2004.

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. extended abstract in Crypto 2001.

[7] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In *FOCS'2007*, 2007.

[8] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO '05*, pages 258–275, 2005.

[9] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, 2007.

[10] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Crypto '06*, 2006.

[11] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Eurocrypt 2003*, volume 2656 of *LNCS*. Springer, 2003.

[12] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[13] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Eurocrypt 2004*, LNCS, 2004. `http://eprint.iacr.org/2003/182/`.

[14] M. Chase. Multi-authority attribute-based encryption. In *TCC 2007*, 2007.

[15] C. Cocks. An identity based encryption scheme based on quadratic residues. In *The 8th IMA International Conference on Cryptography and Coding*, pages 26–8, 2001.

[16] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[17] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Review*, 45(4):727–784, 2003.

[18] R. Gennaro, Y. Gertner, and J. Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *STOC '03*, pages 417–425, 2003.

[19] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *FOCS '00*, 2000.

[20] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08*, 2008. `http://eprint.iacr.org/`.

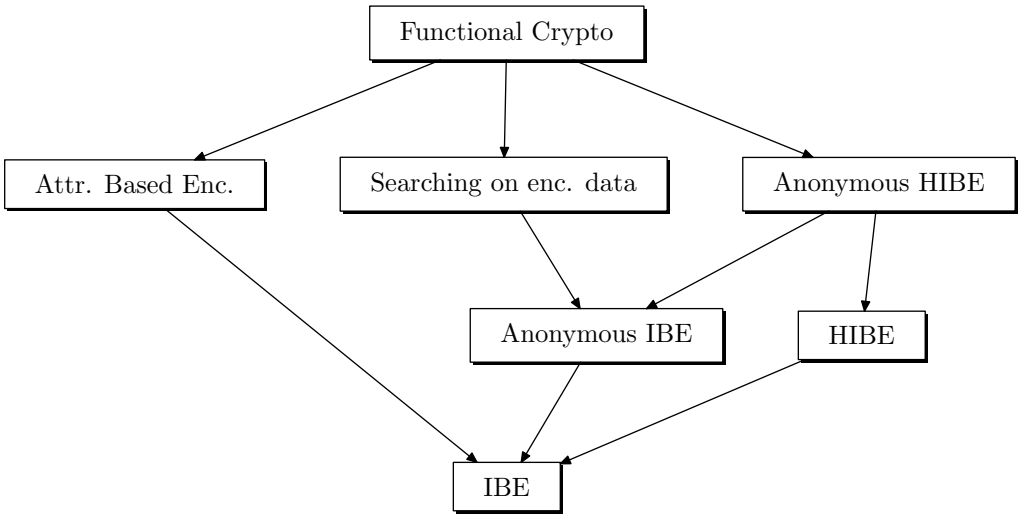

Figure 1. Relations among primitives related to functional encryption. Arrows indicate known black-box reductions. No black-box separations are known between the boxes in the figure.

[21] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Asiacrypt 2002*, 2002.

[22] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pages 325–335, 2000.

[23] Y. Gertner, T. Malkin, and S. Myers. Towards a separation of semantic and CCA security for public key encryption. In *TCC 2007*, 2007.

[24] Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS*, pages 126–135, 2001.

[25] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute Based Encryption for Fine-Grained Access Conrol of Encrypted Data. In *ACM CCS*, 2006.

[26] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In *Eurocrypt 2002*, pages 466–481, 2002.

[27] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC '89*, 1989.

[28] J. H. Kim, D. Simon, and P. Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *FOCS '99*, 1999.

[29] Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *JCRYPTOL: Journal of Cryptology*, 19, 2006.

[30] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Crypto '02*, 2002.

[31] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[32] O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC '04*, LNCS, pages 1–20, 2004.

[33] A. Sahai and B. Waters. Fuzzy Identity Based Encryption. In *Advances in Cryptology – Eurocrypt*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[34] A. Shamir. Identity-based cryptosystems and signature schemes. In *Crypto '84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.

[35] E. Shi, J. Bethencourt, T.H. Hubert Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Conference on Security and Privacy*, 2007.

[36] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions. In *Eurocrypt '98*, 1998.

[37] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Birgit Pfitzmann, editor, *ACM Conference on Computer and Communications Security 2004*, pages 354–63, 2004.