Towards a Separation of Semantic and CCA Security for Public Key Encryption

Yael Gertner[1], Tal Malkin[2], and Steven Myers[3]

[1] Department of Psychology, University of Illinois at Urbana-Champaign.*
(ygertner@cyrus.psych.uiuc.edu)
[2] Department of Computer Science, Columbia University.** (tal@cs.columbia.edu)
[3] School of Informatics, Indiana University.*** (samyers@indiana.edu)

Abstract. We address the question of whether or not semantically secure public-key encryption primitives imply the existence of chosen ciphertext attack (CCA) secure primitives. We show a black-box separation, following the methodology introduced by Impagliazzo and Rudich [23], for a large non-trivial class of constructions. In particular, we show that if the proposed CCA construction's decryption algorithm does not query the semantically secure primitive's encryption algorithm, then the proposed construction cannot be CCA secure.

1 Introduction

Public-key encryption primitives (PKEP) are used in numerous cryptographic protocols. Two frequently used definitions of security for PKEP in the cryptographic literature are semantic and chosen ciphertext attack security. Semantic security (SS) was introduced by Goldwasser and Micali [21] and guarantees that encrypted messages sent over a network are confidential to *passive adversaries* that are limited to eavesdropping (we provide formal definitions of this and the following notions in the next section). Unfortunately, in practice most adversaries are not limited to passive eavesdropping, and they can actively control and manipulate network traffic. This is especially true on the modern Internet, where it is particularly easy to manipulate traffic. Therefore, a strengthened security definition was needed. Naor and Yung [29] introduced Chosen Ciphertext Attack (CCA1) security, in which the adversary is allowed temporary access to a decryption oracle prior to the adversary's attempt to decrypt a message of interest. While this definition is substantially stronger than that of semantic security, it is still not strong enough for many network purposes. Therefore, an even stronger definition of CCA security was introduced by Rackoff and Simon [31] that gives the adversary continuous access to a deprecated decryption oracle that is restricted only in that it will not decrypt ciphertexts of direct interest to the adversary. This security is called CCA2 (or adaptive chosen ciphertext

* This work was done while at the CS Dept. at U. Penn.
** This work was partially done while at AT&T Labs.
*** This work was partially done while at the CS Dept. of the University of Toronto.

attack) security, and is the security standard that most PKEP need to meet in many of today's cryptographic protocols. The first CCA2 secure PKEP was given by Dolev, Dwork, and Naor [11], followed by a large body of research on developing such protocols and understanding the security notion (c.f. [35, 10, 27, 7, 12]).

There are many known constructions of SS PKEPs based on general cryptographic assumptions such as trapdoor predicates[21], trapdoor functions[20, 21], and trapdoor permutations[9]. In addition, these constructions are black-box and are relatively efficient. In contrast, *all known* constructions of CCA1 [29] and CCA2 [11, 27, 35] secure PKEPs from *general cryptographic assumptions* are based on only the existence of enhanced trapdoor permutations and are both non-black-box and inefficient due to their use of ZK or WI proofs.

In this paper we address the question of whether the weaker security requirement (semantic security) for public-key encryption, is in fact equivalent to the stronger requirement (chosen ciphertext attack security). That is, can any SS PKEP be used (without any further assumptions) to construct a CCA PKEP?

This is a natural question which is one of major open problems in cryptography in the last several years. To the best of our knowledge, the first explicit published posing of this as a problem is by Bellare et al. [4], while the most recent one is by Pass, shelat, and Vaikuntanathan [30]. In fact, the latter work addresses a similar problem, and establishes a reduction from any SS PKEP to non-malleable SS PKEP, without any further assumptions (and in a non-black box way). Non-malleable PKEP is a somewhat weaker security requirement than that of CCA2 (in particular, it is equivalent [8] to a definition where the adversary is allowed a single, parallel CCA2 query). As the authors of [30] discuss, their result does *not* generalize to a construction for general CCA security, which remains an interesting open question.

In sum, the current state of knowledge regarding the question we study, is that there is a construction of CCA PKEP from SS PKEP *with additional assumptions*, as well as a (non-black-box) construction of (the weaker) NM PKEP from SS PKEP without any further assumptions. It is not known whether there is an equivalence (whether through a black-box or a non-black-box construction) between SS PKEP and CCA PKEP.

As will be explained below, we show a black-box separation between semantic and CCA1 security for a large interesting class of constructions. This can be interpreted as evidence toward a *negative* answer to our question, or as guidance toward a *positive* answer (a reduction).

1.1 Black-Box Reductions and Separations

The existence of most modern cryptographic primitives implies $\mathcal{P} \neq \mathcal{NP}$, and thus is currently too difficult to prove unconditionally. Instead, cryptographers put a great deal of effort into constructing more complex primitives from simpler ones that are assumed to exist. In such constructions (reductions), if we assume primitives of type P exist and wish to show that a primitive of type Q exists, then we provide a construction C such that $C(M_P)$ is an implementation of Q

whenever M_P is an implementation of P. This is proved by showing that any supposed adversary A_Q breaking $C(M_P)$ as an implementation of Q, can be used for an adversary algorithm A_P breaking M_P as an implementation of P.

However, almost all constructions in modern cryptography are *black-box* (for example, the equivalence of one-way functions, weak one-way functions, PRNG's, PRFG's, PRPG's and digital signatures [19, 22, 26, 28, 33].) This means, intuitively,[4] that the construction C of Q uses the implementation M_P of P as a black box (or oracle), without using the algorithmic description (actual code) of the construction. Moreover, the proof constructs the adversary A_P which uses the adversary A_Q in a black-box manner (again, using it just as an oracle, without looking at its actual code).

While it is not clear how to prove a *negative* result, namely that there exist no reduction of primitive Q to primitive P, Impagliazzo and Rudich [23] initiated a methodology for proving that no *black-box* reductions exist. Specifically, their methodology involved proving that no *relativizing* reduction exists (note that black box reductions must relativize). This is done by exhibiting an oracle relative to which an implementation of P exists, while an implementation of Q does not.[5] Using this methodology, [23] proved a black-box separation between key agreement and one-way functions. A line of subsequent works used this methodology or new variants to show black-box separations among various other cryptographic primitives (c.f. [34, 6, 36, 24, 16, 17]), and to show that black-box constructions suffer from inherent efficiency limitations [25, 14, 15].

Non-Black-Box Constructions. We note that while the vast majority of constructions in cryptography are black-box, there are several results that are non-black-box (importantly, all known constructions of CCA secure PKEP from generic assumption are non-black-box). Many of these constructions are based on using Zero-Knowledge (ZK) or Witness Indistinguishable (WI) proofs (both interactive and non-interactive) in the construction.[6] These proofs are often used

---

[4] There are actually several subtleties and different types of black-box reductions of varying strengths, c.f. [32]. However, this intuitive description suffices for our presentation purposes here.

[5] Even here it's not immediately clear how to make this approach work, since the construction and its proof of security could always ignore the presence of the oracle and independently realize the primitive Q. To address this problem, Impagliazzo and Rudich [23] give a model in which one can prove separations modulo some major results in complexity theory. In their model they begin by assuming that $\mathcal{P} = \mathcal{NP}$, and adding an oracle O relative to which P exists and Q does not, implying that a black-box reduction would yield a proof that $\mathcal{P} \neq \mathcal{NP}$. Subsequent work, starting with Simon [36], used a stronger approach that embeds a PSPACE complete portion into the oracle O before proving that relative to O P exists but Q does not. This yields an unconditional proof that no relativizing (and thus no black-box) reductions exist. Other subsequent work (e.g. [17]) relaxed this approach to obtain a weaker black-box separation methodology.

[6] Perhaps the only exception is the works of Barak [2, 3] who has shown the existence of some protocols that are non-black-box, and that do not make use of ZK techniques.

to prove some property about the circuit description of a cryptographic primitive, and thus require the primitive to have a circuit description, and so are not black-box. Examples of such constructions include the development of PKEP that are secure against chosen ciphertext attacks [29, 35], assuming (enhanced) trapdoor permutations exist[7]. Unfortunately, the protocols that perform such proofs are invariably far too inefficient for practical deployment of the resulting cryptographic primitive (although, they are still polynomial time, they are of a degree that is too large to be practical), thus further justifying the quest for black-box constructions.

The Meaning of Black-Box Separations in Cryptography and Our Scenario In general, a black-box separation can be interpreted as evidence that a reduction of Q to P is unlikely using current techniques, or at least that it is unlikely to be efficient (as black-box reductions seem to be much more efficient than non-black-box ones). Such results may also be viewed as guiding which approaches to take when trying to actually prove a reduction exists. We refer the reader to the previous literature on black-box separations, e.g. [23, 32], for a more in-depth discussion of the meaning and importance of black-box separations in cryptography.

In the particular scenario of the black-box constructions of CCA secure PKEP from SS secure ones, we can view a separation as pointing to several possibilities:

- The need to develop some form of appropriate ZK or WI proofs based on semantic security (and such a direction is attempted in [30]), but such constructions are still likely to be inefficient.
- The need to develop more non-black-box techniques that are more efficient and applicable to the scenario of public-key encryption.
- In the failure of the latter two points, any construction of a CCA secure primitive derived solely from the hardness of a SS secure PKEP will be inefficient, or need to take into account specifics of the assumption that are not generic. For instance, any CCA cryptosystem based on SS PKEP proposed by Ajtai and Dwork [1], that results from the assumed hardness of a lattice problem, will either be too inefficient to be practically useful due to the need to use inefficient non-black-box techniques, or will require a unique construction whose proof of security relies on specific properties of the lattice assumption. This direction might include finding efficient ZK or WI proofs based on the specific hardness assumption under consideration.

1.2 Our Contributions

We prove the following:

[7] Both of these results actually only need the requirement that certain types of non-interactive zero-knowledge proofs exist, and these proofs are known to exist relative to enhanced trapdoor permutations

Theorem (informal statement): There exists no black box reduction that from a given SS PKEP (g, e, d) constructs a CCA1 secure scheme $(G^{g,e,d}, E^{g,e,d}, D^{g,d})$

We call such reductions (where the new primitive's decryption algorithm does not query the underlying primitive's encryption algorithm) *shielding reductions*. Our result, then, rules out any shielding black box reductions of CCA1 PKEP to SS PKEP. Consequently, the only possible constructions of a CCA1 (and thus also of CCA2) secure PKEP from a SS PKEP must either be non-black-box, or have its decryption algorithm use the encryption algorithm of the underlying scheme in an essential way.

Our Model and Proof Technique. The proof essentially follows the IR [23] methodology, showing that there is no (shielding) black-box reduction. This is done by introducing an oracle $\mathcal{O}$ relative to which there exists a SS PKEP $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$, but no CCA secure PKEP $(\mathbf{G}^{\mathbf{O}}, \mathbf{E}^{\mathbf{O}}, \mathbf{D}^{\mathbf{g},\mathbf{d}})$ exists relative to $\mathcal{O}$.[8] Our oracle $\mathcal{O}$ includes $(\mathbf{g}, \mathbf{e}, \mathbf{d})$, where $\mathbf{g}, \mathbf{e}$ are random functions. If there were no other parts to the oracle, the proof of semantic security would be immediate, but then $\mathbf{O}$ would in fact be CCA secure as well. Thus, we add more "weakening" components to $\mathcal{O}$, which make the proof of semantic security a little harder but still relatively simple, but make $\mathbf{O}$ and any other candidate scheme $(\mathbf{G}^{\mathbf{O}}, \mathbf{E}^{\mathbf{O}}, \mathbf{D}^{\mathbf{g},\mathbf{d}})$ vulnerable to CCA1 attacks. The latter is the technical heart of the proof, which is quite complex. We chose to expand on the intuition and main ideas of the proof. The full proof with all technical detail appears at [18].

For clarity of presentation, we start by thinking of all participants as being computationally unlimited, but restricted to making a polynomial number of polynomial sized oracle queries to the oracle $\mathcal{O}$. This already gives an interesting result, and encompasses all the main issues in the proof. Because the constructed adversary in the proof only uses more than a polynomial amount of time (i.e. its computationally unlimited powers) to search for and randomly choose efficiently verifiable strings, it is therefore possible to remove the requirement of computationally unlimited parties and replace it with the ability of randomly choosing $\mathcal{NP}$ witnesses. The proof can then be extended to support computationally bounded parties, by adding a PSPACE complete component to the oracle (or assuming $\mathcal{P} = \mathcal{NP}$), achieving the standard separation model of [23] and most subsequent work.

It may seem that if a construction of a CCA secure scheme $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ from any SS scheme $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ exists, it would be unnatural for $\mathbf{D}$ to call $\mathbf{e}$. After all, e is intended to be used by parties that do not require knowledge of any secret keys, thus using it in an essential way for a decryption algorithm seems counter intuitive.

However, we show that relative to our oracle $\mathcal{O}$, there *is* in fact a CCA2 secure scheme, where $\mathbf{D}$ uses $\mathbf{e}$ (namely a non-shielding black-box construction).

[8] This does not exactly follow the IR methodology, because it does not rule out any relativizing reductions (as the new primitive's $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ algorithms do not have access to the entire oracle $\mathcal{O}$, only to the underlying primitive's algorithms). Nonetheless, it rules out all (shielding) black box reductions.

The basic idea behind this scheme is the following (full details can be found in the full version [18]). To encrypt a message bit b with a random string r, first encrypt b using $\mathbf{e}$ with a public-key pk (and the randomness provided by the string r), and then encrypt all the individual bits of r as well using the same public-key, using new random strings derived *deterministically* from r (for example $r+1, r+2, \ldots$).

This CCA2 secure (relative to our oracle) primitive implies that the shielding limitation on the decryption algorithm in our theorem is inherent for our oracle (and not just a gap in our proof analysis). On the other hand, note that this scheme is *artificial*, and makes heavy use of the fact that $\mathbf{e}$ is a random function, by using new random strings deterministically derived from r (this technique is legitimate when the encryption function is *truly random*, but does not work in general). In fact, based on standard hardness assumptions, it is easy to show that there exist semantically secure PKEP relative to which the above construction does not achieve CCA2 security. Similarly, the construction of Fujisaki and Okamoto [13], which is a black-box construction of CCA PKEP from SS PKEP in the random oracle model, is also non-shielding, but this again heavily relies on the random oracle property.

This leaves open the possibility of using the weaker form of black-box separation of [17] to separate CCA1 security from semantic security without any restrictions on the black box reduction.[9]

We feel that closing this gap and answering whether a black box reduction where the CCA decryption algorithm does invoke the SS encryption algorithm exists, is a very interesting and non-trivial problem for future research. While our work does not completely answer the question of whether CCA secure PKEP can be constructed from SS ones without any further assumptions, we do make significant progress toward that direction.

ORGANIZATION. In Section 2 we formally define the notion of PKEP and the definitions of semantic, CCA1 and CCA2 security. This is followed in Section 3 by a description of our oracle construction, and a proof sketch that relative to such oracles with overwhelming probability there is an SS PKEP. In Section 4 we present our separation theorem and sketch its proof. In Section 5 we provide an example construction on which the various parts of the CCA1 attack are demonstrated, and in Section 6 we briefly discuss why our result transfers to the more tradition model that assumes $\mathcal{P} = \mathcal{NP}$ or that includes a PSPACE oracle.

[9] In fact, using this weaker separation model of [17], we can show that there are no black-box reductions of CCA1 to semantic security for another non-trivial class of constructions, which includes the artificial example mentioned above. Specifically, this is the class where $\mathbf{D}$ does invoke $\mathbf{e}$ in a certain way, where for every successful decryption query $\mathbf{d}(sk, c) \in \{0,1\}$ there is a corresponding invocation of $\mathbf{e}(\mathbf{g}(sk), *, *) = c$ (or very roughly, when D invokes e"in every possible opportunity"). The difficult case for which we do not know how to prove a separation, is the intermediate case where D (roughly) must invoke e in an essential way sometimes, but not other times.

2 Preliminaries and Definitions

2.1 Notation

Given a set S we use the notation $x \in_R S$ to denote the process of choosing x uniformly at random from S. Given a function $f : \mathbb{N} \to \mathbb{R}$, we say it is *negligible* if for all sufficiently large $n \in \mathbb{N}$ and for all $c \in \mathbb{N}$: $f(n) \leq n^{-c}$.

2.2 Definitions of PKEPs

Below we give the formal definitions of PKEPs and the notions of semantic, CCA1 and CCA2 security.

Definition 1 (PKEP). *A public-key encryption primitive is a triple of (G, E, D) of algorithms: G and E are probabilistic while D is deterministic. Let p_1 and p_2 be polynomials specified by the PKEP.*

- *for every n, for every $r \in \{0,1\}^n$ $G(r)$ outputs a pair of keys (sk, pk).*
- *for every $m \in \{0,1\}^{p_1(n)}$, each string $r' \in \{0,1\}^{p_2(n)}$ of coin tosses of E and pair (sk, pk) output by G on some input $r \in \{0,1\}^n$, it holds that $D(sk, E(pk, m, r')) = b$.*

We note that while this definition requires correct decryption our proof can be easily modified to allow for some error in decryption in any purported CCA1 construction.

Next, we give the definitions of semantic, CCA1 and CCA2 security. The definitions are presented concurrently.

Definition 2. *Let $\mathcal{EP} = (G, E, D)$ be a PKEP. Let $A = (A_1, A_2)$ be a probabilistic adversary that is described in two parts, each of which has access to an oracle.*

The PKEP $\mathcal{EP}$ is atk-secure, where atk $\in$ {SS,CCA1,CCA2}, if there exists a negligible function μ such that for every adversary $A = (A_1, A_2)$ and for all sufficiently large $n \in \mathbb{N}$:

$$\Pr_{\substack{s \in_R \{0,1\}^n, (pk,sk) \leftarrow G(s) \\ (x_0, x_1, \sigma) \leftarrow A_1^{O_1}(pk) \\ b \in_R \{0,1\}; r \in_R \{0,1\}^{p_2(n)} c \leftarrow E(pk, x_b, r)}} [A_2^{O_2}(\sigma, c) = b] \leq \frac{1}{2} + \mu(n),$$

where σ represents state information communicated between the parts of the adversary, c represents a **challenge ciphertext** *and :*

- *if atk=SS then O_1 and O_2 are the null oracle: the oracles give the empty response, $\perp$, to all queries;*
- *if atk=CCA1 then $O_1(\cdot) = D(sk, \cdot)$, and O_2 is the null oracle;*
- *if atk=CCA2 then $O_1(\cdot) = D(sk, \cdot)$, and $O_2(\cdot) = D(sk, \cdot)$ but modified on the encryption challenge so that $O_2(c) = \perp$.*

In the case of SS and CCA1 security it is known that there are black-box reductions in both directions between PKEP that encrypt 1-bit messages and PKEP that encrypt n-bit messages (for the direction going from encrypting 1 to n bits, it is easy to see that the concatenation of independent encryptions works as a construction). We make use of this fact in our result, and focus on primitives that encrypt the message space of only one bit. Clearly the above definitions simplify slightly in this case (i.e. $x_0 = 0$ and $x_1 = 1$).

3 The Oracle

We define an experiment that produces an oracle that effectively implements a PKEP that is semantically secure but not CCA1 secure. We think of the oracle as consisting of 5 sub-oracles $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$, but this can easily be unified into one oracle by appropriate coding. This security of the oracle if achieved by effectively defining g, e to be appropriate random length increasing functions, and defining d appropriately, so that it can appropriately decrypt these function. This easily gives a secure PKEP, alas it is too secure (CCA2). Therefore, in order to weaken its security a fourth component of the oracle $\mathbf{w}$ is added which given a public-key pk for (g, e, d) will output an encrypted version of the secret-key. This is of no use to the adversary in the SS definition of security, but makes it trivial for a CCA adversary to break the primitive's security. Finally, a fifth sub-oracle $\mathbf{u}$ is added that gives the adversary the ability to determine the legitimacy of public-keys and ciphertexts (i.e., those that could legitimately be output by $\mathbf{g}$ and $\mathbf{e}$); this sub-oracle is not necessary for the result, but substantially simplifies an already technical proof.

Definition 3 (Oracle Distribution). *Let $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u}) \leftarrow \Upsilon$ denote an oracle that is chosen randomly according to the distribution described below. For each $n \in \mathbb{N}$ let:*

g: $\{0,1\}^n \rightarrow \{0,1\}^{3n}$ *be a random one-to-one function. ($\mathbf{g}$ generates public-keys given secret-keys.)*

e: $\{0,1\}^{3n} \times \{0,1\} \times \{0,1\}^n \rightarrow \{0,1\}^{3n}$ *where for every pk, the function $\mathbf{e}(pk, \cdot, \cdot)$ is a uniformly at random selected one-to-one function. ($\mathbf{e}$ takes a public-key, a message bit and a random string, and outputs a ciphertext.)*

d: $\{0,1\}^n \times \{0,1\}^{3n} \rightarrow \{0,1,\bot\}$ *where for every sk, c and b set $\mathbf{d}(sk, c) = b$ if there exists an r such that $\mathbf{e}(\mathbf{g}(sk), b, r) = c$; otherwise set $\mathbf{d}(sk, c) = \bot$. ($\mathbf{d}$ takes a secret-key and ciphertext and outputs the corresponding decryption.)*

w: $\{0,1\}^{3n} \times \{0,1\}^n \rightarrow \{0,1\}^{3n \times n}$ *where for each pk and j set $\mathbf{w}(pk, j) = \bot$ if $\mathbf{g}^{-1}(pk)$ is undefined; otherwise, if $\mathbf{g}^{-1}(pk) = sk \overset{defn}{=} (sk_1, ..., sk_n)$, set $\mathbf{w}(pk, j) = \mathbf{e}(pk, sk_1, r_{pk,1,j}), \ldots, \mathbf{e}(pk, sk_n, r_{pk,n,j})$, where for $1 \leq k \leq n$ let $r_{pk,k,j} \in_R \{0,1\}^n$. ($\mathbf{w}$ takes a public-key and an index as input, and outputs a bit-by-bit encryption of the public-key's corresponding secret-key.)*

u: $\{0,1\}^{3n} \times \{0,1\}^{3n} \rightarrow \{\top, \bot\}$ *where for each pk and c set $\mathbf{u}(pk, c) = \top$ if there exists an sk, b and r such that $\mathbf{g}(sk) = pk$ and $\mathbf{e}(pk, b, r) = c$; otherwise, set $\mathbf{u}(pk, c) = \bot$. ($\mathbf{u}$ takes a public-key and a string, and determines if the string corresponds to an encryption relative to the public-key.)*

NOTATION: In order to ease discussions of queries to an oracle $\mathcal{O}$, we briefly introduce some notation. Given an oracle $\mathcal{O}$ we often say that $\mathcal{O} = (\mathbf{O}, R)$ where $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ denotes the sub-oracles corresponding to the encryption primitive, and $R = (\mathbf{u}, \mathbf{w})$ corresponds to the security weakening sub-oracle $\mathbf{w}$ and the helper oracle $\mathbf{u}$. We denote by (o, q) the query q to the sub-oracle $o \in \{\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u}\}$ in $\mathcal{O}$. For example, we denote by $(\mathbf{g}, sk)$ the query $\mathbf{g}(sk)$. Similarly, we denote by the pair $(< o, q >, r)$ the response r to the query q made to the sub-oracle o. We call such a pair a query/response, and say a query/response $(< o, q >, r)$ is consistent with o if $o(q) = r$. In cases where a query $q = (v_1, .., v_i)$ is represented by several semantically different strings $v_1, .., v_i$ we denote by $(< o, v_1, .., v_{j-1}, *, v_{j+1}, ..v_i >, r)$ the fact that there exists a v_j such that the oracle query $\mathbf{o}(v_1, v_2, ..., v_i)$ was made and the response was r. For example $(< \mathbf{e}, (pk, *, r) >, c)$ represents the notion that there exists a bit $b \in \{0, 1\}$ such that $(< \mathbf{e}, (pk, b, r) >, c)$ represents a query/response consistent with the sub-oracle $\mathbf{e}$.

The following theorem states that this oracle provides semantic security for the PKEP $(\mathbf{g}, \mathbf{e}, \mathbf{d})$.

Theorem 1. *For every oracle adversary A limited to a polynomial number of oracle queries, there exists a negligible function μ such that for all sufficiently large n:*

$$\Pr_{\mathcal{O} \leftarrow \Upsilon} \left[\Pr[A^{\mathcal{O}}(pk, c) = b] \leq 1/2 + \mu(n) \right] \geq 1 - 1/2^{n/2}$$

where the interior probability is over the choice of $sk \in_R \{0,1\}^n, b \in_R \{0,1\}, r \in_R \{0,1\}^n$ and any coin flips performed by A. Further, $pk = \mathbf{g}(sk)$ and $c = \mathbf{e}(pk, b, r)$.

Proof Sketch: If $\mathcal{O}$ consisted of only the sub-oracles $\mathbf{g}, \mathbf{e}$ and $\mathbf{d}$, then security would follow directly from their probabilistic construction (in a way which is by now standard, c.f. [23, 16]). To ensure that $\mathbf{w}$ and $\mathbf{u}$ do not destroy this security, it is shown that the adversary can effectively simulate the responses of these oracles. An adversary can simulate the response to a query $\mathbf{u}(pk, c)$ by outputting b if there has been a previous query/response $(< \mathbf{e}, pk, b, * >, c)$, and otherwise outputting $\perp$. When b is output the simulation is clearly correct, and when outputting $\perp$ the simulation is correct with high probability, as the ability of the adversary to find a value c such that $\mathbf{e}(pk, *, *)^{-1}(c) \neq \emptyset$ is negligible (in n) due to the random selection of $\mathbf{e}$ (again, following a standard argument). Similarly, $\mathbf{w}(pk, i)$ can be simulated if there has previously been a query/response of the form $(< \mathbf{g}, sk >, pk)$ by outputting a random encryption of sk, and otherwise outputting $\perp$.

It is not hard to verify that $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is not secure against a CCA1 attack: The adversary A_1 takes the input pk, queries $\mathbf{w}(pk, 0)$ to get an encrypted version of sk, and then uses its CCA1 access to the decryption oracle to decrypt sk. sk is then passed to A_2, which uses it to evaluate and output $\mathbf{d}(sk, c)$. In the next section we show that in fact any shielding construction is vulnerable to a (possibly much more complex) CCA1 attack.

4 The Separation

4.1 A Large Class of Constructions: Shielding Reductions

In order to state a proper theorem that provably restricts the class of black-box constructions capable of being CCA1 secure, this class needs to be formally defined. Let $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ be a semantically secure PKEP. We will consider constructions $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^O})$ of PKEPs that are purportedly CCA1 secure. We require that there exist constants ρ_0, ρ_1, ρ_2 and ρ_3 such that for all sufficiently large $n \in \mathbb{N}$ we have:

- $\mathbf{G^O} : \{0,1\}^n \to \{0,1\}^{n^{\rho_0}} \times \{0,1\}^{n^{\rho_1}}$. $(\mathbf{G^O}(S) = (SK, PK))$
- $\mathbf{E^O} : \{0,1\}^{n^{\rho_1}} \times \{0,1\} \times \{0,1\}^{n^{\rho_2}} \to \{0,1\}^{n^{\rho_3}}$ $(\mathbf{E^O}(PK, M, R) = C)$
- $\mathbf{D^O} : \{0,1\}^{n^{\rho_0}} \times \{0,1\}^{n^{\rho_3}} \to \{0,1\} \cup \{\perp\}$. $(\mathbf{D^O}(SK, C) = M)$

In the above definition we consider n the security parameter for the PKEP. We make several assumptions without loss of generality: each of the algorithms on inputs corresponding to security parameter n make exactly n^q queries to $\mathbf{O}$ of size at most n^s, that no duplicate queries are made, that $\mathbf{G}$ never queries $\mathbf{d}$ (it can predict the responses itself), and that the triple satisfies the PKEP correctness property so long as $\mathbf{O}$ does (i.e., all ciphertexts decrypt properly, but again this assumption can be weakened so that random encryptions decrypt properly with some probability greater than 1/2.).

The important assumption we make is that $\mathbf{D}$ does not query $\mathbf{e}$; this is formally what we mean by a shielding construction. This assumption does result in loss of generality and is what is responsible for the restriction in our separation of CCA1 and Semantic Security. This assumption is required in order for latter hybridization experiments to go through. Further, as discussed in the introduction, using the oracle given in this paper, it is possible to construct a non-shielding CCA2 secure PKEP, implying that this assumption is necessary for our oracle.

4.2 Separation Theorem

From this point on, fix an arbitrary PKEP construction $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ that satisfies all of the assumptions of Section 4.1 (in particular, it is shielding).

Theorem 2. *There exists a CCA1 adversary $A = (A_1, A_2)$ for which it's the case that for all sufficiently large n:*

$$\Pr_{\substack{\mathcal{O}=(\mathbf{O},R)\leftarrow\Upsilon \\ S\in_R\{0,1\}^n, M\in_R\{0,1\}, R\in_R\{0,1\}^{n^{\rho_2(n)}} \\ (PK,SK)\leftarrow G^{\mathbf{O}}(S), C\leftarrow\mathbf{E^O}(PK,M,R)}} \left[A_1^{\mathbf{D^O}(SK,\cdot),\mathcal{O}}(PK) \to \sigma; A_2^{\mathcal{O}}(\sigma, C) = M\right] \geq 1 - 1/n.$$

A simple averaging argument then shows that for almost every selection of $\mathcal{O}$, the adversary breaks the $CCA1$ security of the PKEP. Combining this with a simple counting argument shows that there exists a specific oracle relative to which $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ is semantically secure, but where $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ is not.

The main idea behind our oracle separation is as follows. since we want to construct a CCA1 attack, where the adversary only has access to the decryption oracle before it receives the challenge ciphertext, this access cannot be used to learn something specific about the challenge. Therefore, the goal of our adversary will be to use the decryption oracle access to learn enough information on the secret key, that will allow it to later decrypt the challenge ciphertext with good probability. This will be done by reconstructing a secret-key SK', corresponding to its public-key, which will later be used with the decryption algorithm $\mathbf{D}$ to decrypt the challenge ciphertext.

In our black-box model, where parties are computationally unlimited but limited in the number of oracle queries they can make, all security of the constructed primitive $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ must stem from the oracle PKEP $(\mathbf{g}, \mathbf{e}, \mathbf{d})$. Therefore, it seems intuitive that the only secret and usable information that an execution of $G^{\mathbf{O}}(S) \rightarrow (PK, SK)$ embeds in SK are the strings sk for which the corresponding strings $pk = \mathbf{g}(sk)$ have been embedded in PK (It is known by the work of Impagliazzo and Rudich [23] that the construction needs to use the 'trapdoorness' of the oracle if it hopes to be secure, as a random-oracle —such as that provided simply by using only the sub-oracles $\mathbf{g}$ and $\mathbf{e}$— is insufficient to achieve even semantic security). Therefore, our adversary's goal will be to retrieve such sk strings by using the decryption oracle. Clearly, the adversary will additionally have to make use of the sub-oracle $\mathbf{w}$, for without the presence of this oracle, the scheme $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is CCA1 secure. Once such embedded sk are retrieved, the adversary must learn how to use them to actually construct an appropriate SK' and decrypt the challenge ciphertext. Unfortunately, most of these steps are non-trivial, and the adversary is not able to generate a key SK' that can decrypt every ciphertext. Instead, we focus on the ability of finding an SK' that can be used to decrypt the average ciphertext generated by an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen M and R, as this is exactly the distribution from which the adversary's challenge ciphertext will come. Below we give a very high-level description of the steps an adversary must perform to decrypt a challenge ciphertext for the given PKEP.

The large probabilistic experiment (CCA1 attack) that the adversary will perform is broken to the following three parts (given in more detail below).

- In the first part, the adversary uses its input PK to learn the relevant public keys pk and ciphertexts c that are embedded in it.
- In the second part, the adversary's goal is to learn secret keys sk corresponding to the keys pk recovered in the first stage. Note that for each pk, access to $\mathbf{w}$ gives the adversary an encrypted version of sk (encrypted with respect to $\mathbf{e}(pk, *, *)$). The adversary also has access to a decryption oracle $\mathbf{D}(SK, *, *)$. Thus, the goal in this stage will be to 'embed' the encryptions of sk into useful ciphertexts C that can be fed to the decryption oracle, and whose decryptions can be translated back to decryptions of sk. We do not necessarily achieve this goal for all the sk that correspond to pk collected in

the first part. However, we achieve it for enough such *sk* that make the third part go through.
– In the third part, the adversary uses the keys *sk* constructed in the second part, in order to construct an SK' such that $\mathbf{D}(SK', C)$ decrypts correctly with high probability for randomly chosen ciphertexts C.

The technical heart of the proof is in the second and third parts, where enough *sk* should be retrieved to enable a good construction of SK'. Below we provide more technical detail on each of these parts, sketching some of the obstacles encountered and their solutions. The experiment is then described in Section 5 for a specific PKEP construction example that demonstrates several different cases in the proof.

A Caveat We point out that the description here assumes that certain highly unlikely probabilistic events never occur. Examples of such events are the adversary making queries of the form $\mathbf{d}(sk, c) \neq \perp$ when there has never previously been a query of the form $\mathbf{g}(sk) = pk$ or a query $\mathbf{e}(pk, *, *) = c$; or that estimations of specific values retrieved through sampling deviate substantially from the actual value they estimate. In the full version, these bad events are specified, and their possibility of occurring is taken into account in the analysis. To simplify presentation here, it is simply assumed they do not occur.

The Environment & the First Part of the Experiment: Learning about *PK* Define the environment that the adversary is operating in to consist of the oracle $\mathcal{O} = (\mathbf{O}, R)$ that was chosen by Υ in the probabilistic statement of the theorem, as well as the seed S selected to generate the public- and secret-key pair $(PK, SK) = G^{\mathbf{O}}(S)$, where PK is given to the adversary, and access to the decryption oracle $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ is initially given to the adversary. These are fixed for the remainder of the description of all three parts.

The first part of the experiment learns some basic facts about the semantically secure PKEP $\mathbf{O}$, and it learns which $pk \in \mathbf{g}(\{0,1\}^*)$ are 'embedded' in the public-key PK. The determination of these *pk* is done by sampling a large number of executions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen M and R and looking for queries of the form $(\mathbf{e}, pk, *, *)$. If such queries are made, then it is reasonable to assume that *pk might* be embedded into PK. Note there are two issues that immediately arise here: first, there might be values of *pk* retrieved that have been arrived at during the execution of $\mathbf{E}$ by the response to some query $\mathbf{g}(sk)$ (rather than being embedded in PK). However, such values can easily by filtered out by monitoring queries to $\mathbf{g}$. The other issue is that there might very well be *pk* embedded in PK that are never retrieved by this sampling process, but we can safely ignore them, as the fact that they do not show up in this sampling suggests that they are not used during most encryptions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen M and R. Let KS be the set of public-keys *pk* retrieved in the first part of the experiment.

The final thing done in this part of the experiment is that a set $\mathcal{E}$ of specific encryptions output by $\mathbf{e}$ during the executions of $\mathbf{E}$ is created. This is done because some specific encryption c output by $\mathbf{e}$ may be consistently embedded into encryptions C produced by $\mathbf{E}$ (i.e, this information is encoded into PK). Later, the decryption algorithm $\mathbf{D}(SK, \cdot)$ may check for the presence of the embedding of c in C, and refuse to decrypt C if c is missing. Knowledge of such $c \in \mathcal{E}$ will be necessary in the second and third parts of the experiment.

To summarize, at the end of the first stage the adversary has a list KS of public keys pk and a list $\mathcal{E}$ of ciphertexts c (with respect to the system $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$), that were encountered during a large number of random executions of the encryption protocol $\mathbf{E}^{\mathbf{O}}(PK, *, *)$. Intuitively, KS corresponds to the public keys pk embedded into PK, and $\mathcal{E}$ include ciphertexts embedded into PK.

The Second Part of the Experiment: retrieving *sk* embedded in *SK*
In the second part of the experiment the adversary attempts to retrieve a subset of $\mathbf{g}^{-1}(KS)$ to be used to later construct the alternate secret-key SK'. Again, the intuition is that the values in $\mathbf{g}^{-1}(KS)$ that are embedded in SK must be responsible for the purported security of the primitive $(\mathbf{G}, \mathbf{E}, \mathbf{D})$.

Note that for any $pk \in KS$, the adversary can query $\mathbf{w}(pk, 0) = (e_1, ..., e_n)$, where the response $(e_1, \ldots, e_n)$ represents the bit-wise encryption of sk (with respect to pk). Thus, the adversary's goal is to embed these bit encryptions e_i into ciphertexts C whose decryption (obtained from the decryption oracle) helps decrypt e_i to obtain sk. This is done using the following idea (presented in an over simplified form).

Imagine that during the execution of a random encryption of the message M made by $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ there is a query made to $\mathbf{e}(pk, b, r)$ in order to encrypt a bit b for a $pk \in KS$, but which has the property that when one replaces the query's response with a random encryption $\mathbf{e}(pk, 1-b, *)$ of the bit $1-b$, the resulting ciphertext C' output by $\mathbf{E}$ will decrypt to something other than M (we say that it decrypts *improperly* since M is not output); but when one replace the query's response with a random encryption $\mathbf{e}(pk, b, *)$ the resulting ciphertext C'' decrypts to M (respectively, we say it decrypts *properly*). Call such a query $\mathbf{e}(pk, b, r)$ *decisive* with respect to pk. If we can find such decisive queries, then the adversary can use the decryption oracle in conjunction with the encryptions $(e_1, \ldots, e_n)$ obtained from $\mathbf{w}$, to retrieve $sk = \mathbf{g}^{-1}(pk)$. This is done by re-executing $\mathbf{E}(PK, M, R)$ n times, where in the i^{th} iteration it replaces the response to the query $\mathbf{e}(pk, b, r)$ with e_i. In the i^{th} case call the output of $\mathbf{E}$ C_i. If C_i decrypts to M (as discovered with the adversary's decryption oracle), then the adversary knows that the i^{th} bit of sk is b and otherwise it is $1-b$. Therefore, it can retrieve $sk = \mathbf{g}^{-1}(pk)$.

The question is how does the adversary find such decisive queries. There are actually two issues here: how does the adversary know which pk have decisive queries, and assuming it knows that a pk has decisive queries, which query $\mathbf{e}(pk, *, *)$ made during a random encryption $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ is decisive. Assume for the moment that we know that with high probability over the choice of M and R that there is (on average) a decisive query with respect to pk made during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. The adversary can perform n^q (the largest number of queries made by $\mathbf{E}$) hybridization experiments, where in the i^{th} experiment a large number of encryptions $\mathbf{E}(PK, M', R')$ are performed (for randomly chosen M' and R') but in each of these the first i responses to queries of the form $\mathbf{e}(pk, b, *)$ are replaced with random encryptions of bits $\mathbf{e}(pk, b', r')$ (b' and r' randomly chosen), and the responses to the remainder of the queries $\mathbf{e}(pk, *, *)$ are left unaltered. Since we have assumed that such a decisive query must exist, then there will be an $i < n^q$ such that there is a significant increase in the fraction of improper decryptions in the i^{th} and the $(i+1)^{\text{th}}$ experiments. In this case, we can think of the i^{th} query of the form $\mathbf{e}(pk, *, *)$ as being decisive in an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. Of course this is only true on average, so we cannot deduce the value of any bit of $\mathbf{g}^{-1}(pk)$ with a single call to the decryption oracle using the decisive encryption. However, for each bit of sk, we can perform a sampling experiment to retrieve it.

Note that it is the creation of these hybrid ciphertexts that requires us to introduce the shielding restriction. If the construction were not shielding, there is no guarantee that any of the hybrid ciphertexts would decrypt, and therefore the experiment would be useless for retrieving keys. The reason is that, roughly speaking, the reduction being shielding means that in the process of encrypting $\mathbf{E}(PK, M, R)$, replacing one random encryption of $\mathbf{e}(pk, b, r)$ with another random encryption $\mathbf{e}(pk, b, r')$ of the same bit b, should not be noticeable to the decryption algorithm $\mathbf{D}$, and thus still result in a proper decryption (while replacing with an encryption of another bit may result in an improper decryption).

The above explanation assumes that the adversary already knows that a particular $pk \in KS$ will have (on average) a *decisive* query during a random execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. But this presents a serious challenge: how does the adversary even know whether any pk has a decisive query, let alone figure out which pk has one? And what to do about pk that do not have decisive queries? In fact, it is possible that no individual pk has any decisive queries. Consider for example a scheme $\mathbf{E}$ where the message is encrypted three times with respect to three different public keys pk_1, pk_2, pk_3, and the decryption algorithm $\mathbf{D}$ calls $\mathbf{d}$ on each of them, and outputs the majority.[10] Then, it is clear that for any pk_i, even if we replace all encryptions $\mathbf{e}(pk_i, b, *)$ with arbitrary encryptions, the final ciphertext will decrypt correctly, as long as encryptions with respect to the other two public keys are unaltered.

We solve the above problems by adding a layer of an hybridization experiment performed on the set of pk, where at each stage encryptions with respect to all public keys up to the current one are replaced with random encryptions, while other encryptions are executed correctly. Specifically, we consider two sets of keys: a bad key set BKS and a good key set GKS. BKS contains pk that are embedded in PK, but for which $\mathbf{g}^{-1}(pk)$ is unknown. Initially, this is set to be the set KS. GKS contains those pk that were initially in BKS, but for which $sk = \mathbf{g}^{-1}(pk)$ has been previously retrieved by the adversary. Initially, $GKS = \emptyset$. Given BKS we perform the following hybridization experiments over

[10] A more detailed example, including this and several other parts demonstrating various aspects of the overall experiment, is presented in Section 5.

keys in BKS to find a decisive key pk, and then using the methodology described earlier to retrieve $sk = \mathbf{g}^{-1}(pk)$. We can then remove pk from BKS and insert it in GKS. The hybridization experiment over BKS is then repeated until enough secret-keys corresponding to decisive pk embedded into PK have been retrieved.

Suppose $BKS = \{pk_1, ..., pk_\ell\}$, then l hybridization experiments are performed where in the i^{th} experiment we sample the percentage of times a modified execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ produces a ciphertext that *decrypts properly*, when all queries of the form $\mathbf{e}(pk_k, b, r)$ for $(k \leq i)$ are replaced with queries of random encryptions $\mathbf{e}(pk_k, b', r')$ for randomly chosen b' and r'. Clearly in the zeroth experiment, by the correctness property of the PKEP, all encryptions will properly decrypt, and we expect that as we go through the experiments there will be some experiment i, where the percentage of encryptions that decrypt properly drops substantially. This is because we expect that some bits that $\mathbf{E}$ is using to encode M are encoded in encryption $\mathbf{e}(pk, *, *)$ for $pk \in BKS$. If there is no such substantial drop in the percentage of proper decryptions by the final hybridization experiment, then this intuitively corresponds to the case where enough sk that are embedded in SK have been retrieved that are sufficient to construct an alternate decryption key SK'. Note that this does not mean that all of the embedded sk have been retrieved, only that those that have will suffice to construct an SK' that can be used to decrypt an average ciphertext generated by PK.

To illustrate this, consider again the example mentioned above, where the decryption algorithm outputs the majority of decryptions with respect to three keys pk_1, pk_2, pk_3. Here, the first hybridization experiment will identify pk_2 as having some decisive query j, and find $sk_2 = \mathbf{g}^{-1}(pk_2)$ (by replacing encryptions with respect to pk_1, as well as the first $j-1$ encryptions with respect to pk_2, with random encryptions; maintaining encryptions with respect to pk_3, as well as encryptions $j+1$ and on with respect to pk_2 correctly, and replacing the j'th encryption with respect to pk_2 with the encrypted bit of sk_2). It will then move pk_2 to GKS and perform the hybridization experiment again on $BKS = \{pk_1, pk_3\}$ (where encryptions with respect to pk_2 are now always done correctly). It will identify pk_3 as having a decisive query and find sk_3 in a similar manner, ending up with $BKS = \{pk_1\}$, for which it will not succeed to retrieve a corresponding sk_1 (because once encryptions with respect to pk_2, pk_3 are performed correctly, there is no decisive query for pk_1). Nonetheless, having sk_2, sk_3 is sufficient to decrypt the challenge ciphertexts.

Finally, we note that the hybridization experiments described above must take into account the lists obtained in the first stage. In that stage the adversary constructed a set of encryptions $\mathcal{E}$ that had the property that they might be embedded into encryptions $\mathbf{E}(PK, M', R')$ (for random M' and R'), and the decryption algorithm $\mathbf{D}(SK, \cdot)$ checked for the presence of these embeddings. Because of this, when performing the hybridization experiments that were previously described, it is essential that the response to a query $\mathbf{e}(pk, b, r)$ is replace only if $\mathbf{e}(pk, b, r) \notin \mathcal{E}$.

A More Formal Look. We now give a slightly more formal presentation of the second part of the experiment, to give an idea of technicalities involved. In Figure 1 pseudo-code is given for the second part of the experiment and for a helper function $\widehat{\mathbf{E}}$, which is a modified version of the encryption algorithm $\mathbf{E}$. This encryption algorithm takes as additional arguments a set BKS of bad-keys for which responses to encryption queries can be modified, a set $\mathcal{E}$ of encryption queries that cannot be modified, and a series of $2n^q$ ciphertexts for each $pk \in BKS$, where $c_i^{pk,b}$ is used to answer the ith oracle query of the form $\mathbf{e}(pk, b, *)$ made by $\mathbf{E}$, when $pk \in BKS$ and $(\mathbf{e}, pk, b, r) \notin \mathcal{E}$. By properly modifying the input values to this series, hybridization experiments can easily be performed. Recall that n^q is, by assumption, the largest number of queries that $\mathbf{E}$ makes. To aid future discussion, we say that the input $c_\ell^{pk_j,b}$ corresponds to a *correct encrypted bit* if $c_\ell^{pk_j,b} = \mathbf{e}(pk_j, b, *)$, and otherwise it corresponds an *incorrect encrypted bit*. The algorithm $\widehat{\mathbf{E}}$ is not called directly by the second part of the experiment, rather a function called $ApproxErrorRate(t, PK, ES, BKS)$ is introduced to repeatedly call $\widehat{\mathbf{E}}$ on a distribution of inputs—which will be specified momentarily— and returns an estimate of the probability that ciphertexts from that distribution decrypt properly. This estimate is achieved by using the decryption oracle to determine the fraction of ciphertexts output by $\widehat{\mathbf{E}}$ that decrypt properly. The distribution of inputs to $\widehat{\mathbf{E}}$ is specified by calling $\widehat{\mathbf{E}}(PK, M, R, BKS, \mathcal{E}, c'_1, ..., c'_t, c_{t+1}, ..., c_{2|BKS|n^q})$ where M and R are chosen randomly, $c'_1, ..., c'_t$ are encryptions of random-bits under the appropriate pk's, and $c_{t+1}...c_{2|BKS|n^q}$ are encryptions of *correct encrypted bits*. Therefore, by calling *ApproxErrorRate* with an increasing value of t we can perform a hybridization experiment in the second part of the experiment.

To summarize, at the end of the second stage the adversary has a list GKS of public keys (which is a subset of the list KS from the first stage), together with a corresponding $sk = \mathbf{g}^{-1}(pk)$ for each pk in GKS. Intuitively, these $\mathbf{g}^{-1}(GKS)$ are the 'essential' secret keys sk (with respect to the system $\mathbf{O} = (g, e, d)$) which are embedded into the secret key corresponding to PK and are used for proper decryption (in the system $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^O})$).

The Third Part of the Experiment: Constructing SK' Next, we specify how to use the secret-keys in $\mathbf{g}^{-1}(GKS)$ in order to construct a secret-key SK'. Given a specific example of a PKEP, this can often be a trivial task, but we require a uniform procedure that is guaranteed to work for all possible constructions that are considered by the statement of the theorem. Further, there is no guarantee that $GKS = KS$, so there may very well be a secret-key sk embedded into SK, for which $\mathbf{g}(sk) \notin GKS$. From the second part of the experiment we know that $\mathbf{g}^{-1}(GKS)$ contains enough secret-keys embedded into SK to decrypt properly, but not necessarily those that are necessary to reconstruct SK. For an example of the difficulty of constructing a uniform protocol for constructing SK', consider two PKEP that completely ignore the oracle $\mathcal{O}$, and therefore fall into the theorem's specification of acceptable constructions: an RSA based and a Quadratic Residuosity based PKEP. In both cases there would be no sk

$\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{E}, c_1^{pk_1,0}, \ldots, c_{n^q}^{pk_1,0}, c_1^{pk_1,1}, \ldots, c_{n^q}^{pk_1,1}, \cdots,, c_1^{pk_{|BKS|},1}, \ldots, c_{n^q}^{pk_{|BKS|},1})$

$\forall\ pk' \in BKS,\ b' \in \{0,1\}$ set $\delta_{pk',b'} \leftarrow 0$

Simulate Execution $\mathbf{E}^{\mathbf{O}}(PK, M, R)$

  On query $(\mathbf{g}, sk)$ reply with $\mathbf{g}(sk)$.

  On query $(\mathbf{d}, sk, c)$ reply with $\mathbf{d}(sk, c)$.

  On query $(\mathbf{e}, pk, b, r)$

    If $pk \notin KS$ or $(\mathbf{e}, pk, b, r) \in \mathcal{E}$ reply with $\mathbf{e}(pk, b, r)$.

    otherwise

      $\delta_{pk,b} \leftarrow \delta_{pk,b} + 1$

      reply with $c_{\delta_{pk,b}}^{pk,b}$

Output result of simulation

$\textsc{Exp}_2(PK, KS, \mathcal{E})$

  Let $BKS \leftarrow KS$

  Let $GKS \leftarrow \emptyset$

  Repeat $|KS|$ times

    $w \leftarrow 2|BKS|n^q$

    for $t = 0$ **to** w

      Let $p_t = \mathit{ApproxErrorRate}(t, PK, \mathcal{E}, BKS)$

    $\Delta \leftarrow \{j |\ 1 \leq j \leq w$ and $|p_j - p_{j-1}| > \frac{1}{n^{\alpha_6}}\}$ (computed gaps)

    If $\Delta = \emptyset$ then Output (BKS, GKS) and FINISH EXPERIMENT

    $\ell \leftarrow \min \Delta$

    Let $\mathbf{e}(pk_j, b, *)$ correspond to the correct encrypted bit of the ℓth ciphertext input to $\widehat{\mathbf{E}}$.

    $\psi \leftarrow |pk|/3$

    For each k $(1 \leq k \leq \psi)$ set $\mathcal{D}_k \leftarrow \emptyset$

    for $z = 1$ **to** $n^{2\alpha_4}$

      $(d_1, \ldots, d_\psi) \leftarrow \mathbf{w}(pk, z)$ (we consider z a binary string)

      For each k $(1 \leq k \leq \psi)$ set $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{d_k\}$

    for $k = 1$ **to** ψ

      $\pi_k \leftarrow \mathit{ApproxErrorRate}'(\ell, PK, \mathcal{E}, BKS, \mathcal{D}_k)$

      If $|\pi_k - p_{\ell-1}| \geq 1/n^{\alpha_5}$ then $\overline{sk}_k \leftarrow 1 - b$ otherwise $\overline{sk}_k \leftarrow b$

    Let $\overline{sk} = (\overline{sk}_1, \ldots, \overline{sk}_\psi)$

    $GKS \leftarrow GKS \cup \{(pk, \overline{sk})\}$

    $BKS \leftarrow BKS \setminus \{pk\}$

Fig. 1.

embedded in the secret-keys of either PKEP and so this should in theory be an easy case, but based on the public-keys of each respective PKEP the adversary must generate corresponding secret-keys. To solve this problem, in order to find corresponding secret-keys a massive search is used.

We make use of the unlimited computational power of the adversary and have it enumerate all possible pairs of oracles $\mathcal{O}^*$ generated by Υ and seeds S^* that are consistent with our knowledge of $\mathcal{O}$ and SK, and create a set of *Valid Environments*. Note that this step does not actually require the adversary to query the oracle $\mathcal{O}$, for it is simply enumerating all possible environments and checking to see which are consistent. An oracle $\mathcal{O}^*$ and seed S^* are consistent if $\mathbf{G}^{\mathcal{O}^*}(S^*) = (PK, SK^*)$ for some SK^*, this execution of $\mathbf{G}$ queries $\mathbf{g}^*(sk') = pk$ for each $pk \in KS$, and $sk' = \mathbf{g}^{-1}(pk)$ for each $pk \in GKS$. Further, $\mathcal{O}^*$ is consistent with any queries and responses that have been made to $\mathcal{O}$ by the adversary, and $\mathbf{D}^{\mathcal{O}^*}(SK^*, \cdot)$ is consistent with any queries that have been made to the decryption oracle $\mathbf{D}^{\mathcal{O}}(SK, \cdot)$.

Because of the random process Υ by which $\mathcal{O}$ was selected and the random selection of S, each pair $(\mathcal{O}^*, S^*)$ in the set of *Valid Environment* is equally likely to be the environment $(\mathcal{O}, S)$ that the adversary is actually in. Therefore, the adversary uniformly at random selects one such pair $(\mathcal{O}', S')$, and lets SK' be the reconstructed secret-key where $\mathbf{G}^{\mathcal{O}'}(S') = (PK, SK')$. At this point SK' contains the secret-keys in $\mathbf{g}^{-1}(GKS)$, but while $\mathcal{O}$ and $\mathcal{O}'$ agree on all of the queries that have previously been made by the adversary, they probably agree on little else. Therefore, we consolidate the oracles $\mathcal{O}'$ and $\mathcal{O}$ into a new oracle $\widehat{\mathcal{O}}$. This is done so that $\mathcal{O}$ and $\widehat{\mathcal{O}}$ agree on nearly all queries (and in particular any queries that are likely to be made during calls to $C = \mathbf{E}^{\mathbf{O}}(PK, M, R)$ and $\mathbf{D}^{\mathbf{O}}(SK, C)$), but relative to which it is still the case that $\mathbf{G}^{\widehat{\mathbf{O}}}(S') = (SK', PK)$. This is achieved by taking $\mathbf{O}$ and modifying so that it is consistent with any queries that would have been made during the execution of $\mathbf{G}^{\mathbf{O}'}(S') = (SK', PK)$ and $\mathbf{D}^{\mathbf{O}'}(SK', C)$ for every decryption C made by the adversary so far to the decryption oracle $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$.

Since $\mathcal{O}$ and $\widehat{\mathcal{O}}$ agree on nearly all queries, with high probability $\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R) = C$ and therefore $M = \mathbf{D}^{\widehat{\mathbf{O}}}(SK', C) = \mathbf{D}^{\mathbf{O}}(SK, C)$. Therefore, if the adversary could execute $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', \cdot)$ we'd be done, and the adversary could break the CCA1 security of the PKEP with high probability, by simply decrypting the challenge ciphertext. Unfortunately, the adversary cannot construct the oracle $\widehat{\mathcal{O}}$ with a polynomial number of queries to $\mathcal{O}$. It will instead simulate access to $\widehat{\mathbf{O}}$ using $\mathbf{O}$ and $\mathbf{u}$. The largest problem in simulating $\widehat{\mathbf{O}}$ during an execution of $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C)$ is in simulating queries $\widehat{\mathbf{d}}(sk, c)$ for $\widehat{\mathbf{g}}(sk) = pk \in BKS$, because $\mathbf{e}(pk, b, r) = \widehat{\mathbf{e}}(pk, b, r) = c$ for most b and r, but most likely $sk \neq \mathbf{g}^{-1}(pk)$, and therefore $\widehat{\mathbf{d}}(sk, c) = b$ but $\mathbf{d}(sk, c) = \perp$. However, it is exactly such queries whose responses were found not to be necessary for the decryption algorithm, because $pk \in BKS$. Therefore, on such queries $\widehat{\mathbf{d}}(sk, c)$ the adversary simply flips a coin and outputs the result as the response to the query. This is where we use $\mathbf{u}$ to make sure that indeed a bit in $\{0, 1\}$ should be output, as opposed to $\perp$. Using this simulation, $\mathbf{D}^{\widehat{\mathbf{O}}}(SK, C)$ is likely to decrypt properly for an encryption $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen M and R, and thus the adversary can decrypt its challenge ciphertext.

5 An Example

We consider an example of a simple (and artificial) PKEP construction to help ground and clarify the different parts of the experiment. Fix $n \in \mathbb{N}$. Define:

- $\mathfrak{G}^{\mathbf{O}}(S)$: let $S = (S_0, ..., ..., S_8)$, where each $S_i \in \{0,1\}^n$. Query $\mathbf{g}(S_i) = pk_i$ for each i, $0 \leq i \leq 6$. Compute $k_1 = \mathbf{e}(pk_6, 0, S_8)$, and outputs $PK = (pk_0, .., pk_5, pk_6, S_8)$ and $SK = (sk_0 = S_0, ..., sk_5 = S_5, sk_6 = S_6, k_1)$.
- $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$: let PK be as noted, $M \in \{0,1\}$ and $R = (R_0, ..., R_6)$ where each $R_i \in \{0,1\}^n$. Compute $c_i = \mathbf{e}(pk_i, M, R_i)$ for each $1 \leq i \leq 5$. Compute $k_1 = \mathbf{e}(pk_6, 0, S_8)$. If R_6 is the bit-string of all zeros, then query $\mathbf{e}(pk_0, M, R_0) = c_0$ and output $C = (0, k_1, 0^{3n}, 0^{3n}, 0^{3n}, 0^{3n}, c_0)$; otherwise, output $C = (1, k_1, c_1, c_2, \ldots, c_5)$.
- $\mathfrak{D}^{\mathbf{O}}(SK, C)$: Let $C = (b, k_1', c_1, c_2, \ldots, c_5)$ where $b \in \{0,1\}$, $k_1' \in \{0,1\}^n$ and each $c_i \in \{0,1\}^n$. Let SK be as noted. If $k_1' \neq k_1$ output $\perp$. Otherwise, if $\mathbf{d}(sk_6, k_1') \neq 0$ output $\perp$. Otherwise, If $b = 0$, then output $\mathbf{d}(sk_0, c_0)$. Otherwise, let $M_i = \mathbf{d}(sk_i, c_i)$ for each $i \leq 5$, and output $\mathit{Majority}(M_1, ..., M_5)$,

Now consider a (PK, SK) generated by $\mathfrak{G}^{\mathbf{O}}(S)$ as described above, and a CCA adversary attempting break the security of the scheme $(\mathfrak{G}, \mathfrak{E}, \mathfrak{D})$ as prescribed by our experiments.

In the first part of our experiment the adversary will perform a large number of encryptions $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen M and R, and will observe queries of the form $\mathbf{e}(pk_i, *, *)$ made during such executions for $1 \leq i \leq 6$, but it is unlikely that queries $\mathbf{e}(pk_0, *, *)$ are observed. Thus it is unlikely the adversary will need pk_0 to decrypt the challenge ciphertext and it can be ignored. The adversary also will observe the query $\mathbf{e}(pk_6, 0, S_8)$ with response k_1, and note that it will have to ensure the key it later constructs is consistent with this query/response.

In the second part of the experiment the adversary will attempt to determine sk_i for $1 \leq i \leq 6$. This will be done by encrypting random messages by executing $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$, but replacing responses of queries of the form $\mathbf{e}(pk_i, b, r)$ with responses to $\mathbf{e}(pk_i, b', r')$ where b' and r' are chosen randomly in a hybridization experiment. In this case the hybridization is over the pk_i. In such an experiment, the resulting ciphertexts C' will either decrypt to the appropriate message M that was originally encrypted or it will not (note the adversary uses the decryption oracle to check this).

In our toy example, randomizing only the responses to all queries $\mathbf{e}_{pk_i}$, $i \in \{1, 2\}$, will result in proper decryptions, as the Majority function in $\mathfrak{D}$ acts as a form of error-correcting code. However, when responses to all queries of the form $\mathbf{e}_{pk_i}$, $i \in \{1, 2, 3\}$, are randomized, the result is occasional improper decryptions. The occasional improper decryption allows the adversary to determine sk_3. This is because the oracle $\mathbf{w}$ will provide a number of random encryptions of sk_3 that can injected into modified executions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ as in the hybrid experiment. By determining if the ciphertexts produced by these executions of $\mathbf{E}$ decrypt properly the bits of sk_3 can be retrieved. By the end of the second part of the experiment the adversary will have retrieved sk_i for $3 \leq i \leq 6$. Note that sk_i, $0 \leq i \leq 2$ will not be retrieved because of the error-correcting properties of the Majority function in $\mathfrak{D}$. Still, this is sufficient to decrypt on average and thus all the adversary will ask.

In the third part of the experiment the adversary must reconstruct the secret-key. Since it does not know $sk_0, ..., sk_2$ it cannot reconstruct SK, but it can construct an SK' that is satisfactory to decrypt the challenge ciphertext. From observation it is clear that a secret-key of the form $SK' = (sk'_0, sk'_1, sk'_2, sk_3, ..., sk_6, k_1)$ will decrypt the challenge ciphertext with high probability, only possibly failing in the unlikely event that the first bit of the challenge ciphertext is 0. The issue is automating the above construction. In order to do so the adversary essentially searches through all oracle/seed pairs $(\widehat{\mathcal{O}}, \widehat{S})$ in which the oracles are consistent with everything the adversary knows about $\mathcal{O}$ (i.e. $\mathbf{g}(sk_i) = pk_i$ for $3 \leq i \leq 6$ and $\mathbf{e}(pk_6, 0, S_8) = k_1$) and that $\mathbf{G}^{\widehat{\mathbf{O}}}(\widehat{S}) = (\widehat{SK}, PK)$. Such a $\widehat{SK}$ is then used by the adversary to decrypt its challenge ciphertext.

6 The Complexity Theoretic Statements

A quick review of the experiment the adversary performs shows that the only situation in which the adversary uses more than a polynomial amount of computation is when it must select uniformly at random an oracle and seed pair $(\mathcal{O}', S')$ from the set of *Valid Environments*. It selects such oracles and seeds based on them satisfying a polynomial number of local consistency constraints that are efficiently verifiable. Further, once this is done almost all of the oracle $\mathcal{O}'$ is thrown out when the adversary consolidates $\mathcal{O}$ with $\mathcal{O}'$. Therefore, the process of randomly selecting an oracle and seed could alternately be thought of as selecting an oracle 'stub' with corresponding seeds, where the oracle stub only specifies the oracle's values on those queries that are necessary to satisfy the constraints mentioned. Once such a stub had been selected, the oracle can be randomly extended to a full oracle if needed without changing the distribution. However, choosing such stubs can be thought of as uniformly at random selecting an $\mathcal{NP}$ witness. Bellare, Goldreich and Petrank [5] show that if $\mathcal{P} = \mathcal{NP}$ then one can efficiently and uniformly at random select $\mathcal{NP}$-Witnesses. Therefore, we can consider this result in the more traditional model of Impagliazzo and Rudich[23], and state the theorem in the traditional computational model, based on the assumption that $\mathcal{P} = \mathcal{NP}$. Alternatively, following the lead of Simon [36], we can further embed a *PSPACE* oracle into our final oracle $\mathcal{O}$. Since $\mathcal{P}^{PSPACE} = \mathcal{NP}^{PSPACE}$ we get a restriction on black-box result in the standard computational model.

Acknowledgments We would like to thank Bill Aiello for posing the problem, and participating in the initial stages of this research. The third author would like to thank Charles W. Rackoff and Toniann Pitassi for many useful discussions.

References

1. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In ACM, editor, *Proceedings of the twenty-ninth annual ACM Symposium on the Theory of Computing: El Paso, Texas, May 4–6, 1997*, pages 284–293, 1997. ACM order no. 508970.
2. B. Barak. How to go beyond the black-box simulation barrier. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 106–115, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 2001. IEEE Computer Society Press.
3. Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, pages 345–355. IEEE Computer Society, 2002.
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO: Proceedings of Crypto*, 1998.
5. Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of np-witnesses using an np-oracle. *Electronic Colloquium on Computational Complexity (ECCC)*, 5(32), 1998.

6. Mihir Bellare, Shai Halevi, Amit Sahai, and Salil Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. Cryptology ePrint Archive, Report 1998/019, 1998. http://eprint.iacr.org/.
7. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. *Lecture Notes in Computer Science*, 950:92–111, 1995.
8. Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. *Lecture Notes in Computer Science*, 1666:519–536, 1999.
9. Manuel Blum and Shafi Goldwasser. An *efficient* probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–299. Springer-Verlag, 1985, 19–22 August 1984.
10. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 13–25, London, UK, 1998. Springer-Verlag.
11. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, 6–8 May 1991.
12. E. Elkind and A. Sahai. A unified methodology for constructing publickey encryption schemes secure against adaptive chosen-ciphertext attack, 2004.
13. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999.
14. R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science*, pages 305–313. IEEE Computer Society Press, 2000.
15. Rosario Gennaro, Yael Gertner, and Jonathan Katz. Lower bounds on the efficiency of encryption and digital signature schemes. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing*, pages 417–425. ACM Press, 2003.
16. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In IEEE, editor, *41st Annual Symposium on Foundations of Computer Science*, pages 325–335. IEEE Computer Society Press, 2000.
17. Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In IEEE, editor, *42nd IEEE Symposium on Foundations of Computer Science*, pages 126–135. IEEE Computer Society Press, 2001.
18. Yael Gertner, Tal Malkin, and Steve Myers. Towards a separation of semantic and cca security for public key encryption. Cryptology ePrint Archive, 2006. http://eprint.iacr.org/.
19. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.
20. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.
21. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

22. J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudo-random generator from any one-way function. *Accepted to the SIAM Journal of Computing*, 28(4):1364–1396, 1998.
23. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61. ACM Press, 1989.
24. Jeff Kahn, Michael Saks, and Cliff Smyth. A dual version of reimer's inequality and a proof of rudich's conjecture. In *COCO '00: Proceedings of the 15th Annual IEEE Conference on Computational Complexity*, page 98. IEEE Computer Society, 2000.
25. Jeong Han Kim, D. R. Simon, and P. Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 535–542. IEEE Computer Society Press, 1999.
26. L. A. Levin. One-way functions and pseudorandom generators. In *ACM Symposium on Theory of Computing (STOC '85)*, pages 363–365, Baltimore, USA, May 1985. ACM Press.
27. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 2003.
28. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17:373–386, 1988.
29. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Baruch Awerbuch, editor, *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 427–437, Baltimore, MY, May 1990. ACM Press.
30. R. Pass, a. shelat, and V. Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO '06: Proceedings of the 26th Annual International Cryptography Conference on Advances in Cryptology*, 2006.
31. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 433–444, London, UK, 1992. Springer-Verlag.
32. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.
33. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In Baruch Awerbuch, editor, *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 387–394, Baltimore, MY, May 1990. ACM Press.
34. Steven Rudich. The use of interaction in public cryptosystems (extended abstract). In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 242–251. Springer, 1991.
35. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, 1999.
36. D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT 98*, pages 334–345, 1998.