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

**Extended Abstract**

Yael Gertner[*] Tal Malkin[†] Steven Myers [‡]

**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, using the methodology introduced by Impagliazzo and Rudich [20], 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.*

[*]Department of Psychology, University of Illinois at Urbana-Champaign: ygertner@cyrus.psych.uiuc.edu. (Work was done while at the CS Dept. at U. Penn)

[†]Department of Computer Science, Columbia University: tal@cs.columbia.edu. (This work was partially done while at AT&T Labs)

[‡]School of Informatics, Indiana University, Bloomington: samyers@indiana.edu. (This work was partially done while at the CS Dept. of the University of Toronto)

## 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 [18] and guarantees that encrypted messages sent over a network are confidential to *passive adversaries* that are limited to eavesdropping. 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 and cheap to manipulate traffic. Therefore, a strengthened security definition was needed. Naor and Yung [29] introduced Chosen Ciphertext Attack (CCA#1) security, in which it is assumed that the adversary temporarily has access to a decryption oracle in order to simulate the adversary's ability to manipulate traffic. In particular, the adversary has access to the oracle until a time where it wishes to attempt to decrypt a message of interest to it. 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 CCA#2 (or adaptive chosen ciphertext attack) security, and is the security standard that most PKEP need to meet in many of today's cryptographic protocols. The first CCA#2 secure PKEP was given by Dolev, Dwork, and Naor [10], followed by a large body of research on developing such protocols and understanding the security notion (c.f. [36, 9, 25, 6, 11]).

There are many known constructions of SS PKEPs based on general cryptographic assumptions such as trapdoor predicates[18], trapdoor functions[17, 18], and trapdoor permutations[8]. In addition, these constructions are black-box and are relatively efficient. In contrast, *all known* constructions of CCA#1 [29] and CCA#2 [10, 25, 36] 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 CCA#1 (in particular, it is equivalent [7] to a single CCA#1 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 CCA#1 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 [16, 19, 24, 26, 34].) This means, intuitively,[1] 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 [20] initiated a methodology for proving that no *black-box* reductions exist. Specifically, their methodology involved proving that no *relativizing* reduction exists (since 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.[2] Using this methodology, [20] 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. [35, 27, 37, 21, 14, 15]), and to show that black-box constructions suffer from inherent efficiency limitations [23, 12, 13].

**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.[3] These proofs are often used to prove some property about the circuit description of a cryptographic primitive, and thus require the primitive to have a circuit description, and thus are not black-box. Examples of such constructions include the development of PKEP that are secure against chosen ciphertext attacks [29, 36], assuming (enhanced) trapdoor permutations exist[4]. 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. [20, 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.

[1]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.

[2]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 [20] 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 [37], 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. [15]) relaxed this approach to obtain a weaker black-box separation methodology.

[3]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.

[4]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

- 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.

## 1.2 Our Contributions

We prove the following:
**Theorem (informal statement):** There exists no black box reduction that from a given SS PKEP $(g, e, d)$ constructs a CCA#1 secure scheme $(G^{g,e,d}, E^{g,e,d}, D^{g,d})$

Consequently, the only possible constructions of a CCA#1 (and thus also of CCA#2) 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 follows the IR [20] methodology, showing (the stronger condition) that there is no relativizing 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 $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^{g,d}})$ exists relative to $\mathcal{O}$. Our oracle $\mathcal{O}$ in fact includes $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ as 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 $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^{g,d}})$ vulnerable to CCA#1 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 in the body of the paper, including the full proof with all technical detail in the appendix.

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 [20] and most subsequent work.

One may argue 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 seems unnatural for $D$ to call $\mathbf{e}$. After all, $e$ is intended to be used by any party and does not require the knowledge of any secret keys, and 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 CCA#2 secure scheme, where $D$ uses $e$ (while our results show that without $D$ using $e$, no scheme can achieve even CCA#1 security). The basic idea behind this scheme (which is presented in detail in Appendix H.1) is the following. To encrypt a message bit $b$ with a random string $r$, first encrypt $b$ using $e$ 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 using $r + 1, r + 2, \ldots$ as the random strings to encrypt the individual bits of $r$).

This CCA#2 secure (relative to our oracle) primitive implies that the 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 $e$ 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 CCA#2 security.

This leaves open the possibility of using the weaker form of black-box separation of [15] to separate CCA#1 security from semantic security without any restrictions. (The separation model of [15] does not prove that no relativizing reductions exist, but rather shows that for any candidate construction of $Q$ from $P$, there exists an oracle relative to which the implementation of $P$ remains secure, but the proposed construction fails to implement $Q$.)[5]

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 the next section we formally define the notion of PKEP and the definitions of semantic, CCA#1 and CCA#2 security. This is followed by a description of the random process used to generates the oracles $\mathcal{O}$ used in our separation, and a proof sketch that relative to such oracles it is both highly likely that there is a provably semantically secure PKEP that does not achieve CCA#1 security. This is followed by a section that formalizes the exact class of black-box constructions of PKEP to which our separation result applies, followed by a sketch of the proof of the separation theorem. Finally, we briefly discuss why our result transfers from the computationally unlimited adversarial model to the more tradition model that assumes $\mathcal{P} = \mathcal{NP}$ or that includes a PSPACE oracle.

## 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, CCA#1 and CCA#2 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 $r \in_R \{0,1\}^n$ $G(r)$ 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 $D(sk, E(pk, m, r')) = b$.*

Next, we give the definitions of semantic, CCA#1 and CCA#2 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.*

[5]In fact, using this weaker separation model (that was introduced in [15]), we can show that there are no black-box reductions of CCA#1 to semantic security for another non-trivial class of constructions, which includes the artificial example mentioned above. Specifically, this is the class where $D$ does invoke $e$ 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.

*The PKEP $\mathcal{EP}$ is atk-secure, where atk $\in$ {SS,CCA#1,CCA#2}, if there exists a negligible function $\mu$ such that for every adversary $A = (A_1, A_2)$ and for all sufficiently large $n \in \mathbb{N}$:*

$$\Pr_{\substack{s \in_R \{0,1\}^n, (pk,sk) \leftarrow G(s) \\ (x_0,x_1,\sigma) \leftarrow A_1^{O_1}(pk) \\ b \in_R \{0,1\}; r \in_R \{0,1\}^{p_5(n)} c \leftarrow E(pk,x_b,r)}} [A_2^{O_2}(\sigma, c) = b] \leq \frac{1}{2} + \mu(n),$$

*where $\sigma$ 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=CCA#1 then $O_1(\cdot) = D(sk, \cdot)$, and $O_2$ is the null oracle;*
- *if atk=CCA#2 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 CCA#1 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 CCA#1 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, unfortunately it is too secure (CCA#2). 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. ($\mathbf{g}$ as 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:** $\{0,1\}^n \times \{0,1\}^{3n} \rightarrow \{0, 1, \perp\}$ *where for every $sk, c$ and $b$ set $\mathbf{d}(sk, c) = b$ if there exists an $r$ such that $\mathbf{e}(\mathbf{g}(sk), b, r) = c$; and otherwise set $\mathbf{d}(sk, c) = \perp$. ($\mathbf{d}$ 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 $\mathbf{w}(pk, j) = \perp$ 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:** $\{0,1\}^{3n} \times \{0,1\}^{3n} \rightarrow \{\top, \perp\}$ *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 $\mathbf{u}(pk, c) = \perp$.*
*($\mathbf{u}$ 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 *q*uery/response, and say a query/response $(< o, q >, r)$ is *c*onsistent 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 4.** *For every oracle adversary $A$ limited to a polynomial number of oracle queries, there exists a negligible function $\mu$ 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 $c = \mathbf{e}(pk, b, r)$ and $\mu$ is a negligible function.*

**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. [20, 14]). 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 $\perp$.

Next, we briefly sketch why $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is not secure against CCA#1.

**Theorem 5.** *There exists an adversary $A = (A_1, A_2)$ limited to a polynomial number of oracle queries such that for all sufficiently large $n$:*

$$\Pr_{\mathcal{O} = (\mathbf{O}, R) \leftarrow \Upsilon} \left[ \Pr[A_2^{\mathcal{O}}(\sigma, c) = b] = 1 \right] = 1,$$

*where the interior probability is over the following experiment: $sk \in_R \{0,1\}^n; pk \leftarrow \mathbf{g}(sk); (0, 1, \sigma) \leftarrow A_1^{\mathbf{d}(sk,\cdot), \mathcal{O}}(pk); b \in_R \{0,1\}; r \in_R \{0,1\}^n; c \leftarrow \mathbf{e}(pk, b, r)$. Therefore, the PKEP defined by $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ in the oracle is not CCA#1 secure*

**Proof Sketch:**The adversary $A_1$ takes the input $pk$, queries $\mathbf{w}(pk, 0)$ and decrypts the response using the decryption oracle to retrieve $sk = \mathbf{g}^{-1}(pk)$. $sk$ is then passed to $A_2$, which uses it to evaluate and output $\mathbf{d}(sk, c)$.

## 4 The Separation

### 4.1 A Large Class of Constructions

In order to state a proper theorem that provably restricts the class of black-box constructions capable of being CCA#1 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 $(\mathbf{G}^{\mathbf{O}}, \mathbf{E}^{\mathbf{O}}, \mathbf{D}^{\mathbf{O}})$ of PKEPs that are purportedly CCA#1 secure. We require that there exist constants $\rho_0$, $\rho_1$, $\rho_2$ and $\rho_3$ such that for all sufficiently large $n \in \mathbb{N}$ we have:

- $\mathbf{G}^{\mathbf{O}} : \{0,1\}^n \to \{0,1\}^{n^{\rho_0}} \times \{0,1\}^{n^{\rho_1}}$. $(\mathbf{G}^{\mathbf{O}}(S) = (SK, PK))$

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

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).

The important assumption is that we assume $\mathbf{D}$ does not query $\mathbf{e}$. This assumption does result in loss of generality and is what is responsible for the restriction in our separation of CCA#1 and Semantic Security. This assumption is required in order for latter hybridization experiments to go through. Further, using the oracle given in this paper, it is possible to construct a CCA#2 secure PKEP if we ignore this assumption, therefore in some sense it is necessary. We point out that this CCA#2 secure construction does not permit a proof of security when replaced with a computationally secure PKEP (as opposed to the PKEP given by the random selected oracle). This construction is was sketched in Section 1.2 and is discussed in full detail in the Appendix H.1.

## 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.

**Theorem 6.** *There exists a CCA#1 adversary $A = (A_1, A_2)$ for which it's the case that for all sufficiently large $n$:*

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

A simple averaging argument then shows that for almost every selection of $\mathcal{O}$, the adversary breaks the *CCA*#1 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 $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^O})$ is not.

The main idea behind our oracle separation is as follows. If a PKEP is to be CCA#1 insecure, but semantically secure, then queries that the adversary makes to the decryption oracle must leak information about a secret-key $SK$ corresponding to the public-key $PK$ given to the adversary. Such queries cannot be used to learn about the challenge ciphertext, because in this (CCA#1) security definition the adversary only has access to the decryption oracle before it receives the ciphertext. Failure to make use of the decryption oracle implies the adversary will make no progress, as it is the only distinction between the definitions of security. Therefore, the goal of our adversary will be to reconstruct a secret-key $SK'$, corresponding to its public-key. 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 $G^{\mathbf{O}}(S) \to (PK, SK)$ 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 [20] 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 CCA#1 secure. Once such embedded $sk$ are retrieved, the adversary must learn how to use them to actually decrypt the challenge ciphertext. In order to do this, the adversary reassembles them into a usable, alternate secret-key $SK'$ that functions with the algorithm $\mathbf{D}$. 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 $\mathbf{E^O}(PK, M, R)$ 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 that the adversary will perform is broken into three parts to help with presentation. The experiment is then described for a specific PKEP construction example that demonstrates several different cases in the proof. The Appendix contains the full technical description of the experiment discussed here, and its proof of correctness.

**A Caveat** Before the explanation of the experiment is given, we point out that this high-level experiment 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 Appendix, 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.

### 4.2.1 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 $\Upsilon$ 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 of the adversary's experiment.

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: firstly, 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 $\mathbf{E}$ 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 $\mathbf{O} = (g, e, d)$), 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 $PK$.

### 4.2.2 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 $(\mathbf{G}, \mathbf{E}, \mathbf{D})$.

In order to retrieve the secret keys in $\mathbf{g}^{-1}(KS)$ we use the following idea (presented in a 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 sub-oracle $\mathbf{w}$ of $\mathcal{O}$ to retrieve $sk = \mathbf{g}^{-1}(pk)$. This is done by querying $\mathbf{w}(pk, 0) = (e_1, ..., e_n)$, where $(e_1, ..., e_n)$ represent the bit-wise encryption of $sk$. The adversary can now re-execute $\mathbf{E}(PK, M, R)$ $n$ times, where in the $i^{\text{th}}$ iteration it replaces the response to the query $\mathbf{e}(pk, b, r)$ with $e_i$. In the $i^{\text{th}}$ 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 $i^{\text{th}}$ 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 $i^{\text{th}}$ 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 semantically secure encryptions of random 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 $i^{\text{th}}$ and the $(i+1)^{\text{th}}$ experiments. In this case, we can think of the $i^{\text{th}}$ 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. Without loss of generality, assume we're retrieving the first bit of $sk$, $sk_1$. We perform random encryptions $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ replacing the first $i - 1$ queries $\mathbf{e}(pk, *, *)$ with random encryptions, the $i^{\text{th}}$ query with an encryption of $sk_1$ (where the encryption is provided by the sub-oracle $\mathbf{w}$) and the remaining queries are left unaltered. The ciphertexts output by these modified executions of $\mathbf{E}$ can then be sent through the decryption oracle, and based on how frequently the ciphertexts decrypt properly, we can determine with high probability the value of the bit of $sk$. This process is then iterated to retrieve the remaining bits of $sk$.

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 $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. 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 (using the above method by determining that decisive queries were made with respect to $pk$). Initially, $GKS = \emptyset$. Given $BKS$ we perform the following hybridization experiments over keys in $BKS$ to find a decisive key $pk$, and then using the above methodology 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 $i^{\text{th}}$ 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 all the $sk$ 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 in to construct an $SK'$.

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 semantically secure 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 $\mathbf{e}(pk,b,r) \notin \mathcal{E}$.

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 $(\mathbf{G}^{\mathbf{O}}, \mathbf{E}^{\mathbf{O}}, \mathbf{D}^{\mathbf{O}})$).

### 4.2.3 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 $sk$ 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 $\Upsilon$ 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 that $\mathbf{D}^{\mathcal{O}^*}(SK',\cdot)$ is consistent with any queries that have been made to the decryption oracle $\mathbf{D}^{\mathcal{O}}(SK,\cdot)$.

Because of the random process $\Upsilon$ 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, 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 CCA#1 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 $\mu$. 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. 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.

## 4.3 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.

## 4.4 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[20], and state the theorem in the traditional computational model, based on the assumption that $\mathcal{P} = \mathcal{NP}$. Alternatively, following the lead of Simon [37], we can further embed a *PSPACE* oracle into our final oracle $\mathcal{O}$. Since $\mathcal{P}^{PSPACE} = \mathcal{NP}^{PSPACE}$ we get a non-relativizing 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 and Phillip Rogaway. Optimal asymmetric encryption. *Lecture Notes in Computer Science*, 950:92–111, 1995.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] E. Elkind and A. Sahai. A unified methodology for constructing publickey encryption schemes secure against adaptive chosen-ciphertext attack, 2004.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[17] 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.

[18] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[19] J. Hastad, R. Impagliazzo, L.A. Levin, and M. Luby. Construction of pseudorandom generator from any one-way function. *Accepted to the SIAM Journal of Computing*, 28(4):1364–1396, 1998.

[20] 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.

[21] 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.

[22] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory.* The MIT Press, 1994.

[23] 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.

[24] 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.

[25] Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. In *EUROCRYPT: Advances in Cryptology: Proceedings of EUROCRYPT*, 2003.

[26] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17:373–386, 1988.

[27] Amit Sahai Mihir Bellare, Shai Halevi 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/`.

[28] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[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] A. shelat R. Pass 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] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120, February 1978. The basics of trap-door functions and the famous RSA public key cryptosystem are presented in this paper.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

## A Appendix

**A Note to the Reader** This appendix really just constitutes the last chapter of the PhD dissertation of the third author. It is provided, as it contains the full proofs of all of the theorems in the paper. Additionally, it includes more background and uses the example of Section 4.3 as a running example demonstrating how the adversary would break this particular proposed construction that concurrently exemplifies many of the problems encountered in the proof. We point this out, so that the numerous reference to the 'current chapter' do not confuse the reader, nor do the one or two references to previous chapters, which are unimportant in the current context. Nonetheless, if the reader would prefer the entire copy of the dissertation itis available electronically at `http://www.informatics.indiana.edu/samyers/phdDis.pdf`.

**Beginning of the Chapter** In this chapter we will consider different types of public-key encryption primitives (PKEP). These primitives permit two parties that have never met to exchange secret messages. The primitives consist of three algorithms, denoted $(G, E, D)$. A user Alice will use the algorithm $G$ to randomly generate a secret-/public-key pair $(sk, pk)$. Alice will publish $pk$ so that everyone is aware that her public-key is $pk$. Any other user, say Bob, can use the probabilistic encryption algorithm $E$ along with $pk$ to send an encryption $c$ of a secret message $m$ to Alice. Alice can then use the secret-key along with the decryption algorithm to recover the secret message $m$ from its encryption $c$.

In the cryptographic community there are several notions of security for PKEPs. The three most common are semantic, chosen-ciphertext attack #1 (CCA#1), and chosen-ciphertext attack #2 (CCA #2) security. We briefly describe these three notions[6]:

**Semantic security** guarantees that an adversary that is permitted to view random encryptions of messages of its choice will not learn anything when presented with a random encryption of a message randomly selected from a list of messages provided by the adversary.

**CCA#1 security** strengthens the adversary by permitting it to temporarily request and receive the decryptions of as many ciphertexts as it chooses: in essence the adversary has access to an oracle that decrypts ciphertexts. The ability of the adversary to request decryptions is then revoked. A PKEP is considered CCA#1 secure if, after having had temporary access to a decryption oracle, an adversary is unable to learn anything when presented with a random encryption of a message randomly selected from a list of messages provided by the adversary.

**CCA#2 security** further strengthens the CCA#1 adversary by allowing it almost complete access to the decryption oracle. Specifically, a PKEP is considered CCA#2 secure if, after having had access to a

[6]The formal definitions will be provided later in the chapter

decryption oracle, an adversary is unable to learn anything when presented with a random encryption, $c$, of a message randomly selected from a list of messages provided by the adversary. However, in this case the adversary is permitted to continue accessing the decryption oracle, so long as it does not attempt to decrypt the challenge encryption $c$.

There are many known constructions of semantically secure PKEPs based on general cryptographic assumptions such as trapdoor predicates[18], trapdoor functions[17, 18], and trapdoor permutations[8]. In addition, these constructions are black-box and are relatively efficient. In contrast, all known constructions of CCA#1 [29] and CCA#2 [10, 25, 36] secure PKEPs from general cryptographic assumptions are based on only the existence of trapdoor permutations and are both non-black-box and inefficient due to their use of ZK or WI proofs.

The lack of efficient, black-box constructions of CCA#1 and CCA#2 secure PKEPs is frustrating, as most cryptographic protocols that make use of PKEPs require them to have security properties that are strictly stronger than those given by CCA#1 security, although often security weaker than CCA#2 security suffices. Therefore, limitations on the efficiency of the constructions of such PKEPs place limitations on the efficiency of many cryptographic protocols.

## B The Main Result

The question that we consider in this chapter is whether or not there are black-box constructions of CCA#1 PKEPs from semantically secure PKEP. This question was first posed by Bellare et al. [4]. While we are unable to completely answer the question, we are able to make significant progress and bridge the gap in understanding the requirements for constructing a CCA#1 PKEP from a semantically secure PKEP.

We consider a model where all parties are computationally-unlimited Turing machines with oracle access. Further, these machines are restricted so that they may make at most a polynomial number of polynomial sized oracle queries (where the polynomial is with respect to the size of the input of the machine). We show that in such a model there exists an oracle that provides a semantically secure PKEP, but relative to which there is a large natural class of black-box PKEP constructions that are not CCA#1 secure.

We will construct an oracle $\mathcal{O}$ that has five distinctive types of queries. It will be useful to think of the oracle as being comprised of five sub-oracles $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$. The sub-oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ will naturally implement a *CCA*#2 secure PKEP[7]. Therefore, we will introduce the sub-oracle $\mathbf{w}$ to weaken the security of $\mathbf{O}$. Relative to $\mathbf{w}$ the PKEP given by $\mathbf{O}$ is semantically but not CCA#1 secure. Intuitively, the sub-oracle $\mathbf{w}$ will provide encryptions of the secret-key of interest to an adversary, and therefore a CCA#1 or CCA#2 adversary that has access to a decryption oracle will be able to use it in conjunction with $\mathbf{w}$ to retrieve the secret-key. In contrast, a semantically secure adversary that does not have access to the decryption oracle finds access to $\mathbf{w}$ useless. The sub-oracle $\mathbf{u}$ provides a method for distinguishing valid encryptions from arbitrary strings, and thus provides a functionality that is both useful in our proof and is equivalent to a property that appears in several real PKEPs (i.e. the systems presented in [10, 25, 29, 36])

We will show that relative to $\mathcal{O}$ no scheme $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{g},\mathbf{d}})$ can be CCA#1 secure. That is, if a proposed CCA#1 PKEP has a decryption algorithm that does not query the encryption sub-oracle $\mathbf{e}$, then there is no black-box proof that the scheme is CCA#1 secure. This is an interesting class of PKEPs, as the notion of having a CCA#1 secure PKEP's decryption algorithm calling the semantically secure PKEP's encryption algorithm is arguably neither a natural nor an obvious requirement.

We will also give a construction relative to our sub-oracle $\mathbf{O}$ that is CCA#2 secure relative to $\mathcal{O}$, and therefore relative to our oracle it is not possible to remove the limitation on the class of constructions that do not permit black-box proofs of CCA#1 security in our proof. Unfortunately, it is easy to see that our CCA#2 construction is very artificial and makes use of artificially strong security properties of the oracle $\mathcal{O}$. In fact, based on standard hardness assumptions, it is easy to show that there exist semantically secure PKEP relative to which our construction does not achieve CCA#2 security. We observe that these results leave open the possibility of using the weaker form of black-box separation of Gertner et al. [15] to separate

[7]We will think of the sub-oracle $\mathbf{g}$ as implementing the key-pair generation algorithm, the sub-oracle $\mathbf{e}$ as implementing the encryption algorithm and the sub-oracle $\mathbf{d}$ as implementing the decryption algorithm.

CCA#1 security from semantic security without any restrictions on the types of black-box constructions that are considered.

The main result in this chapter is based on joint work with Yael Gertner and Tal Malkin, who had preliminary discussions with Bill Aiello.

## C Preliminaries & Notation

**Notation 7.** *Let $S$ be a finite set, and let $x \in_R S$ denote the act of choosing an element $x$ uniformly at random from $S$.*

**Notation 8.** *Let $\prec$ denote a lexicographic ordering over $\{0,1\}^*$, and let $\mathcal{S} = \{s_1, .., s_m\} \subset \{0,1\}^*$ be an arbitrary finite set where it is the case that for $1 < i \leq m$ that $s_{i-1} \prec s_i$. For $1 \leq i \leq m$ we define $Index(i, S)$ to be $s_i$.*

**Definition 9.** *We say that function $\mu : \mathbb{N} \rightarrow \mathbb{R}$* **is negligible** *if for all sufficiently large $n \in \mathbb{N}$ and all constants $c > 0$, it is the case that $\mu(n) \leq 1/n^c$.*

A well known tail-bound on the sum of identically distributed Bernoulli random-variables is the Chernoff-Hoeffding bound. This bound will be used many times in this chapter, and is presented below. Refer to [28], [22] or any other standard book on probabilistic computation for a proof of this theorem.

**Theorem 10 (Chernoff-Hoeffding Bound).** *Let $x_1, x_2, x_3, \ldots$ be identical, independently distributed random variables over 0 and 1 with probability $p$ and $1-p$, respectively. Let $X_{n^t} = \frac{1}{n^t}\sum_{i=1}^{n^t} x_i$. For any $n, k, t > 0$:*

$$\Pr\left[|X_{n^t} - p| \geq \frac{1}{n^k}\right] \leq 2^{-2n^{t-2k}}.$$

## D Standard Definitions

Below we give the formal definitions of PKEPs and Semantic, CCA#1 and CCA#2 security.

**Definition 11 (Public-Key Encryption primitive (PKEP)).** *A public-key encryption primitive is a triple $(G, E, D)$ of algorithms. Algorithms $G$ and $E$ are probabilistic while $D$ is deterministic. Let $p_1, p_2, p_3, p_4$ and $p_5$ be polynomials that will specify the asymptotic lengths of the inputs and outputs for the algorithms.*

$G$ **(Key Generation Algorithm):** *for every $n \in \mathbb{N}$ this algorithm takes as input a random $n$-bit string, where $n$ defines the security parameter of the system. It generates two strings, $sk$ and $pk$ which are respectively called the the secret and public keys, and where $|sk| = p_1(n)$ and $|pk| = p_2(n)$.*

$E$ **(Encryption Algorithm):** *for every $n \in \mathbb{N}$ this algorithm takes as input a public-key $pk$ of length $p_2(n)$, a message $m \in \{0,1\}^{p_4(n)}$ that is referred to as the plain-text and a random string of bits of length $p_5(n)$. The output is a string of $p_3(n)$ bits, and is referred to as the ciphertext.*

$D$ **(Decryption Algorithm):** *for every $n \in \mathbb{N}$ this algorithm takes as input a key $sk$ of length $p_1(n)$ and a supposed ciphertext $c$ of length $p_3(n)$. It outputs a string of length $p_4(n)$ (the decryption of $c$) or the symbol $\perp$ indicating that $c$ is not a valid ciphertext.*

**Correctness Requirement:** *For every $n$; for every pair $(sk, pk)$ that is generated by $G(s)$ for $s \in \{0,1\}^n$; and for every $m \in \{0,1\}^{p_4(n)}$: $D(sk, E(pk, m)) = m$.*

Next, we give the definitions of semantic, CCA#1 and CCA#2 security. The definitions are presented concurrently to prevent redundancy and to make it easier for the reader to observe the similarities and differences in the different versions of the definitions.

**Definition 12.** *Let $\mathcal{EP} = (G, E, D)$ be an encryption primitive with associated polynomials $p_1, ..., p_5$ as previously described in Defn. 11.*

*Let $A = (A_1, A_2)$ be a probabilistic adversary that is described in two parts, each of which has access to an oracle which will either represent a decryption oracle, or a null oracle. Each part of the adversary is restricted to making at most a polynomial number of queries to these oracles, where the polynomial is specified in terms* *of the length of the input of the respective part adversary. The first part of the adversary, $A_1$, will be given access to a randomly chosen public-key of length $p_2(n)$, and will output two messages $x_0, x_1 \in \{0,1\}^{p_4(n)}$, as well as any pertinent state information $\sigma$ it would like to pass along to the second part of the adversary $A_2$. The second part of the adversary, $A_2$, will be given an encrypted message $c \in \{0,1\}^{p_3(n)}$, some state information $\sigma$ and outputs a bit.*

*The PKEP $\mathcal{EP}$ is atk-secure, where atk $\in$ {semantic,CCA#1,CCA#2}, if there exists a negligible function $\mu$ such that for every adversary $A = (A_1, A_2)$ and for all sufficiently large $n \in \mathbb{N}$:*

$$\Pr_{\substack{s \in_R \{0,1\}^n, (pk,sk) \leftarrow G(s) \\ (x_0,x_1,\sigma) \leftarrow A_1^{O_1}(pk) \\ b \in_R \{0,1\}; r \in_R \{0,1\}^{p_5(n)} c \leftarrow E(pk,x_b,r)}} [A_2^{O_2}(\sigma, c) = b] \leq \frac{1}{2} + \mu(n),$$

*where $\sigma$ represents state information and :*

- *if atk=semantic then $O_1$ and $O_2$ are the empty oracle: the oracles give the empty response, $\perp$, to all queries;*
- *if atk=CCA#1 then $O_1(\cdot) = D(sk, \cdot)$, and $O_2$ is the empty oracle;*
- *if atk=CCA#2 then $O_1(\cdot) = D(sk, \cdot)$, and $O_2(\cdot) = D(sk, \cdot)$ but modified on the encryption challenge so that $O_2(c) = \perp$.*

We note that in Defn. 12, in the case that the message space for PKEP is $\{0,1\}$ (i.e. $p_4$ is the constant function valued at 1), then given the adversary $A = (A_1, A_2)$, the only values that make sense for $A_1$ to output are 0 and 1, and therefore in the case of CCA#1 security, we can assume that these are the outputs of $A_1$, and so there is no need for them to be output. In the case of CCA#2 security, $A_1$ can be removed from the definition with no loss to the adversary's power, as $A_2$ can simulate any queries $A_1$ would have made, so long as $PK$ is included in the state information $\sigma$. We point these modifications out, as they will be assumed for the remainder of the chapter.

From Defn. 12 it is fairly easy to see that the following theorem holds, and it and its proof can found in numerous sources in the literature, for example [4]

**Theorem 13.** *CCA#2 security implies CCA#1 security and CCA#1 security implies semantic security.*

## E Organization of the Chapter

We provide a brief sketch of the layout of this chapter. We will begin by describing a random process for constructing oracles that we will use to prove many of the results in this chapter. The oracles constructed by this process contain a natural sub-oracle that effectively implements a PKEP. We will argue that the PKEP that is defined by an oracle, chosen by the described random process, is semantically secure with probability close to 1. Next, we will formalize the exact class of PKEP constructions that we will show are insufficient for proving CCA#1 security follows from black-box use of a semantically secure PKEP. The pertinent restriction on this class of constructions will be that the decryption algorithm will not be permitted to make use of the semantically secure PKEP's encryption algorithm. We will show that this restriction is necessary relative to the oracles our random process creates, by demonstrating a PKEP construction that achieves CCA#2 security from the semantically secure PKEP given by the oracle; but this construction has the property that the proposed decryption algorithm calls the encryption algorithm of the semantically secure PKEP.

Finally, we will begin the task of describing how, given a PKEP construction that belongs in the class described, to construct a CCA#1 adversary that will be able to break the proposed PKEP's CCA#1 security. In order to simplify the description of this adversary, we break it down into three experiments that it must run consecutively. We then describe these three experiments, and show why they're successful at breaking the CCA#1 security of the proposed PKEP. The details of the three parts of this experiment, as well as other aspects of the constructed adversary, will be given in more detail later when the reader has been made aware of more context. We conclude with a discussion of the implications of this adversary, and the strengths and weaknesses of the results presented in this chapter.

## F The Distribution for the Oracle $\mathcal{O}$

We describe a large probabilistic experiment that generates a distribution on oracles that we will be interested in. We will use the probabilistic method to show that there exists an oracle in this distribution that has the security properties that we are interested in. The experiment that generates the distribution on oracles is described below:

**Definition 14 (Oracle Distribution).** *We define an experiment that produces an oracle that effectively implements a 1-bit semantically secure public-key encryption primitive.*

*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.*

**g** *For each $n \in \mathbb{N}$ let $\mathbf{g} : \{0,1\}^n \rightarrow \{0,1\}^{3n}$ be a random one-to-one function.*
*(Think of $\mathbf{g}$ as generating public-keys from secret-keys)*

**e** *For each $n \in \mathbb{N}$ let $\mathbf{e} : \{0,1\}^{3n} \times \{0,1\} \times \{0,1\}^n \rightarrow \{0,1\}^{3n}$ be a function such that for every $pk \in \{0,1\}^{3n}$ we let $\mathbf{e}(pk, \cdot, \cdot)$ be a random one-to-one function.*
*(Think of $\mathbf{e}$ as taking a public-key, a bit to be encrypted and a random string, and outputting a ciphertext.)*

**d** *For each $n \in \mathbb{N}$ let $\mathbf{d} : \{0,1\}^n \times \{0,1\}^{3n} \rightarrow \{0, 1, \perp\}$ be defined such that for every $sk \in \{0,1\}^n, c \in \{0,1\}^{3n}$ and $b \in \{0,1\}$ we define $\mathbf{d}(sk, c) = b$ if there exists an $r \in \{0,1\}^n$ such that $\mathbf{e}(\mathbf{g}(sk), b, r) = c$; and otherwise we define $\mathbf{d}(sk, c) = \perp$.*
*(Think of $\mathbf{d}$ as taking a secret-key and ciphertext and outputting the corresponding decryption.)*

**w** *For each $n \in \mathbb{N}$ let $\mathbf{w} : \{0,1\}^{3n} \times \{0,1\}^n \rightarrow \{0,1\}^{3n \times n}$ be defined as follows. For each $pk \in \{0,1\}^{3n}$ and $j \in \{0,1\}^n$ we set $\mathbf{w}(pk, j) = \perp$ if $\mathbf{g}^{-1}(pk)$ is undefined; otherwise, let $\mathbf{g}^{-1}(pk) = sk$, which we will represent using a bitwise notation as $(sk_1, ..., sk_n)$, and we 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 $1 \leq k \leq n$ we randomly chose $r_{pk,k,j} \in \{0,1\}^n$.*
*(Think of $\mathbf{w}$ as taking a public-key and an index, and outputting an encrypted version of the public-key's corresponding secret-key. The encryptions are under the given public-key, and done bit-by-bit.)*

**u** *Garbage Oracle: For each $n \in \mathbb{N}$ let $\mathbf{u} : \{0,1\}^{3n} \times \{0,1\}^{3n} \rightarrow \{\top, \perp\}$ be defined as follows. For each $pk \in \{0,1\}^{3n}$ and $c \in \{0,1\}^{3n}$ we define $\mathbf{u}(pk, c) = \top$ if there exists an $sk \in \{0,1\}^n, b \in \{0,1\}$ and $r \in \{0,1\}^n$ such that $\mathbf{g}(sk) = pk$ and $\mathbf{e}(pk, b, r) = c$; otherwise, we define $\mathbf{u}(pk, c) = \perp$.*
*(Think of $\mathbf{u}$ as taking a public-key and a string, and deciding whether or not the string corresponds to an encryption relative to the given public-key.)*

For purposes of presentation, we will often refer to an oracle $\mathcal{O}$ as a pair of sub-oracles $(\mathbf{O}, R)$ where $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $R = (\mathbf{w}, \mathbf{u})$. As was mentioned previously, the intuition for the oracle $\mathcal{O} = (\mathbf{O}, R)$ is as follows: the sub-oracle $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ defines a traditional public-key encryption primitive that is CCA#2 secure; therefore, we cannot hope to prove the separations that we're interested in. Thus, we include the oracle $\mathbf{w}$ and it weakens the security properties of $\mathbf{O}$. The weakening is accomplished by having $\mathbf{w}$ provide random encryptions under $\mathbf{e}$ of a secret-key, $sk = \mathbf{g}^{-1}(pk)$, that correspond to the public-key, $pk$, that is given to $\mathbf{w}$ as input. This makes the PKEP defined by $\mathbf{O}$ easily attackable by CCA#1 and CCA#2 adversaries: given a public-key $pk$ the adversaries use $\mathbf{w}$ in conjunction with their decryption oracles to retrieve a corresponding secret-key $sk$. We will show that, not surprisingly, with this weakening the semantic security of primitive defined by $\mathbf{O}$ is preserved. Finally, we provide a sub-oracle $\mathbf{u}$ that discerns *valid* encryptions (i.e. those with a corresponding decryption relative to an appropriate key-pair) from arbitrary strings that are not encryptions.

We introduce some notation that is valuable for discussing queries and replies to an oracle $\mathcal{O}$.

**Notation 15.** *Given an oracle $\mathcal{O}$, 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 $o(q) = r$.*

**Definition 16.** *We say that the* size *of a query $(o, q)$ to the sub-oracle $o \in \{\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u}\}$ in $\mathcal{O}$ is $|q|$. Thus, we will ignore the number of bits that would actually be necessary to encode the sub-oracle in the query.*

We also introduce the following slight abuse of notation that permits us to easily discuss broad classes of queries and responses from oracles.

**Notation 17.** *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 $\mathbf{e}$.*

## G The Semantically Secure PKEP Embedded in the Oracle $\mathcal{O}$

We formalize the sense in which a random oracle $\mathcal{O}$ provides a semantically secure PKEP. We first point out how we syntactically get a PKEP from an oracle $\mathcal{O} = (\mathbf{O}, R)$ produced by the random process $\Upsilon$ that was described in Section F. As alluded to earlier, we note that the oracle $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ syntactically represents a PKEP. This can be seen by noting that for every $n \in \mathbb{N}$ we can generate a set of public- and secret-keys by randomly selecting a secret-key $sk \in_R \{0,1\}^n$ and outputting $(sk, pk = \mathbf{g}(sk))$; given a public key $pk$ of size $3n$, generated by querying $\mathbf{g}$, we can encrypt a bit $b \in \{0, 1\}$ by querying $\mathbf{e}(pk, b, r) = c$ and generating the ciphertext $c$, where $r \in_R \{0,1\}^n$; and finally given a secret-key $sk$ of size $n$ we can decrypt a ciphertext $c$ of size $3n$ by querying $\mathbf{d}(sk, c) = b$, and outputting the result $b$. What needs to be approached more carefully is the semantic security of such a system.

**Theorem 18.** *For every oracle adversary $A$ limited to a polynomial number of oracle queries, there exists a negligible function $\mu$ such that for all sufficiently large $n$:*

$$\Pr_{\mathcal{O}=(\mathbf{O},R)\leftarrow\Upsilon}\left[\Pr_{\substack{sk\in_R\{0,1\}^n\\ b\in_R\{0,1\}\\ r\in_R\{0,1\}^n}}[A^{\mathcal{O}}(pk, c) = b] \leq 1/2 + \mu(n)\right] \geq 1 - 1/2^{n/2}$$

*where $pk = \mathbf{g}(sk)$ and $c = \mathbf{e}(pk, b, r)$ and $\mu$ is a negligible function.*

*Sketch.* If the oracle $\mathcal{O}$ was comprised solely of $\mathbf{O}$ then for each $n \in \mathbb{N}$ the probability that the PKEP would be secure would follow in a straight-forward manner from the probabilistic construction of $\mathcal{O}$. To observe that $\mathbf{u}$ does not effect the security of $\mathbf{O}$, we note that with very high probability the adversary could simulate the response to a query $\mathbf{u}(pk, c)$ as follows: if there has previously been a query/response $(< \mathbf{e}, pk, b, * >, c)$ then respond with $b$, otherwise respond with $\perp$. This simulation succeeds because for sufficiently large $n$ it is only with exponentially small probability that an adversary is able to find a $c \in \{0,1\}^{3n}$ such that there exists a $pk \in \{0,1\}^{3n}$, $b \in \{0, 1\}$ and $r \in \{0,1\}^n$ where $\mathbf{e}(pk, b, r) = c$, without actually making the query $\mathbf{e}(pk, b, r)$, due to the random construction of $\mathcal{O}$.

Similarly, with very high probability the adversary could simulate the response to a query $\mathbf{w}(pk, i)$ as follows: if there had previously been a query/response $(< \mathbf{g}, sk >, pk)$, then it can respond with a random encryption of each bit of $sk$ (i.e. we output $c_i = \mathbf{e}(pk, sk_i, r_i)$ for the random encryption of the $i$th bit of $sk$, which we denote by $sk_i$ and were $r_i$ is a random string). Alternatively, if there has never been a query of the form $\mathbf{g}(sk) = pk$ then it can respond with $\perp$, because due to the random selection of $\mathbf{g}$ it is exponentially unlikely that there will exist a string $sk$ such that $\mathbf{g}(sk) = pk$. □

We know briefly sketch why the system provided by the oracle is not secure against chosen-ciphertext attacks. The following theorem describes a CCA#1 adversary that completely breaks the system, independent of the choice of $\mathcal{O}$.

**Theorem 19.** *There exists an adversary $A = (A_1, A_2)$ limited to a polynomial number of oracle queries such* *that for all sufficiently large n:*

$$\Pr_{\mathcal{O}=(\mathbf{O},R)\leftarrow\Upsilon}\left[\Pr_{\substack{sk\in_R\{0,1\}^n,pk\leftarrow\mathbf{g}(sk)\\(0,1,\sigma)\leftarrow A_1^{\mathbf{d}(sk,\cdot),\mathcal{O}}(pk)\\b\in_R\{0,1\};r\in_R\{0,1\}^n c\leftarrow E(pk,b,r)}}[A_2^{\mathcal{O}}(\sigma,c)=b]=1\right]=1,$$

*and therefore the PKEP defined by* $(\mathbf{g},\mathbf{e},\mathbf{d})$ *in the oracle is not CCA#1 secure*

*sketch.* Fix the oracle $\mathcal{O}$. The adversary $A_1$ takes input $pk$ and makes the query $\mathbf{w}(pk,\overline{0})\to(c_1,...,c_n)$, where $\overline{0}$ represents a string of $n$ zeros and each $c_i\in\{0,1\}^{3n}$. For each $i$, ciphertext $c_i$ represents an encryption of the $i$th bit of $sk$. Therefore, for each $i$ the adversary $A_1$ retrieves the $i$th bit of $sk$ by making the query $c_i$ to the decryption oracle. Once, $sk$ has been decoded by $A_1$, it passes it to $A_2$ as the state information represented by $\sigma=sk$ in the statement of the experiment. The adversary takes its inputs $sk$ and $c$ and outputs $\mathbf{d}(sk,c)$, which will always equal $b$. □

## H A Large Class of Black-Box Constructions

We wish to show that there is no black-box proof that semantically secure PKEPs imply CCA#1 secure PKEPs for a large class of black-box constructions of PKEP. In this section we formalize exactly which constructions we will consider. We will consider constructions $(\mathbf{G^O},\mathbf{E^O},\mathbf{D^O})$ of PKEPs that are purportedly CCA#1 secure. We require that there exist constants $\rho_0$, $\rho_1$, $\rho_2$ and $\rho_3$ such that for each $n\in\mathbb{N}$ we have:

$\mathbf{G^O}$ computes a function of the form $\mathbf{G^O}:\{0,1\}^n\to\{0,1\}^{n^{\rho_0}}\times\{0,1\}^{n^{\rho_1}}$. If $\mathbf{G^O}(S)=(SK,PK)$ then we interpret $S$ as a seed, $SK$ as a secret-key and $PK$ as a public-key.

$\mathbf{E^O}$ computes a function of the form $\mathbf{E^O}:\{0,1\}^{n^{\rho_1}}\times\{0,1\}\times\{0,1\}^{n^{\rho_2}}\to\{0,1\}^{n^{\rho_3}}$. If $\mathbf{E^O}(PK,M,R)=C$, we interpret $PK$ as the public-key, $M$ as the message to be encrypted, $R$ as a sequence of random bits and $C$ as the resulting ciphertext.

$\mathbf{D^O}$ computes a function of the form $\mathbf{D^O}:\{0,1\}^{n^{\rho_0}}\times\{0,1\}^{n^{\rho_3}}\to\{0,1\}\cup\{\perp\}$. If $\mathbf{D^O}(SK,C)=M$ then we interpret $SK$ as a secret-key, $C$ as a ciphertext, and $M$ as the decrypted message

We note that in the above description we think of $n$ as a security parameter, and thus given a PKEP $(\mathbf{G},\mathbf{E},\mathbf{D})$ as described, then any inputs to $\mathbf{G}$, $\mathbf{E}$ or $\mathbf{D}$ that have a size consistent with the above definition for a fixed value $n$, are said to be **inputs consistent with the security parameter** $n$.

We point out that we have implicitly assumed that the message $M$ to be encrypted by $\mathbf{E}$ is a single bit. We note that there are trivial black-box transformations that take CCA#1 secure PKEP from a single-bit encryption primitive to a many-bit encryption primitive[8], and therefore the assumption is without loss of generality.

**Assumption 20.** *Without loss of generality we assume that there exist constants s and q such that on all inputs that on all inputs of correspond to security parameter n, each execution of each of* $\mathbf{G^O},\mathbf{E^O}$ *and* $\mathbf{D^O}$ *makes exactly* $n^q$ *queries to* $\mathbf{O}$, *and each is of size no larger than* $n^s$. *We further assume, without loss of generality, that* $\mathbf{G},\mathbf{E}$, *and* $\mathbf{D}$ *never make duplicate queries to* $\mathbf{O}$.

**Assumption 21.** *We assume that* $\mathbf{G^O}$ *never queries the sub-oracle* $\mathbf{d}$. *This assumption is only made to simplify presentation and the results presented in this chapter can be proven without such an assumption. To observe that this is not a severe restriction we note that given a randomly chosen* $\mathcal{O}=(\mathbf{O},R)$ *that* $\mathbf{G^O}$ *could effectively predict the result of any query* $\mathbf{d}(sk,c)$*: in the case that it had previously queried* $(<\mathbf{e},\mathbf{g}(sk),b,r>,c)$ *it can respond with b, and if no such query has been made, then with high probability the result is* $\perp$.

[8]In the case of CCA#1 security, one can produce a many-bit encryption system by concatenating together the independent bitwise encryptions of a CCA#1 secure single-bit encryption system

**Assumption 22.** *We assume that $\mathbf{E^O}$ is constructed such that for any strings $sk$ and $c$ prior to making the query $\mathbf{d}(sk, c)$ it makes the query $\mathbf{g}(sk)$, and then checks to see if it has previously made a query/response $(<\mathbf{e}, \mathbf{g}(sk), b, *>, c)$, and if so it responds to its own query $\mathbf{d}(sk, c)$ with $b$. Otherwise it performs the query $\mathbf{d}(sk, c)$.*

**Assumption 23.** *We assume that for every oracle $\mathcal{O} \leftarrow \Upsilon$, and every construction $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^O})$ it is the case that if there exits an $S$ such that $\mathbf{G^O}(S) = (PK, SK)$ and there exists $R$ and $M$ such that $\mathbf{E^O}(PK, M, R) = C$ then $\mathbf{D^O}(SK, C) = M$. That is we assume that our construction meets the correctness criteria of a PKEP.*

Any PKEP construction could easily be modified to satisfy the above assumptions. We might hope to show that relative to a randomly chosen $\mathcal{O} \leftarrow \Upsilon$ that any PKEP black-box construction $\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^O}$ is not CCA#1 secure. Unfortunately, this is not the case. In the following subsection we show a counter-example, and it demonstrates the necessity of our final restriction on constructions. This final restriction is then presented in Section H.2.

## H.1 A CCA#2 Secure Scheme Relative To O

We are faced with the issue that relative to a randomly chosen $(\mathbf{O}, R) = \mathcal{O} \leftarrow \Upsilon$ there are black-box constructions $(\mathbf{G^O}, \mathbf{E^O}, \mathbf{D^O})$ that are CCA#2 secure. Below we provide a brief and informal description of such a construction. For each $n \in \mathbb{N}$:

$\mathbf{G^O}(S)$ outputs $(SK = S, PK = \mathbf{g}(S))$ where $S \in \{0,1\}^n$.

$\mathbf{E^O}(PK, M, R)$ outputs $C = (\mathbf{e}(PK, M, R), \mathbf{e}(pk, r_1, R+1), \ldots, \mathbf{e}(pk, r_n, R+n))$, where $PK \in \{0,1\}^{3n}, M \in \{0,1\}$ and $R \in \{0,1\}^n$. Further, we represent the $i$th bit of $R$ by $r_i$. We calculate $R+i$ by using a standard method of converting back and forth between bit-strings and integers.

$\mathbf{D^O}(SK, C = (c_0, ..., c_n))$ computes $\mathbf{d}(SK, c_i) = r_i$ for each $i \leq n$; where $SK \in \{0,1\}^n$, $C \in \{0,1\}^{3n^2+3n}$ and for each $i$ we have $|c_i| = 3n$. Next, $\mathbf{D}$ sets $R' = (r_1, ..., r_n)$, and if it's the case that for every $i \leq n$ that $\mathbf{e}(\mathbf{g}(SK), r_i, R+i) = u_i$ then $\mathbf{D}$ outputs $r_0$, otherwise $\mathbf{D}$ outputs $\perp$.

We now provide a high-level argument that the scheme that was just presented is CCA#2 secure.

**Theorem 24.** *For every CCA#2 adversary $A$, there exist negligible functions $\mu$ and $\mu'$ such that for all sufficiently large $n$:*

$$\Pr_{\mathcal{O}\leftarrow\Upsilon}\left[\Pr_{\substack{S\in_R\{0,1\}^n,(PK,SK)\leftarrow\mathbf{G^O}(S)\\ M\in_R\{0,1\},R\in_R\{0,1\}^n\\ C\leftarrow\mathbf{E^O}(PK,M,R)}}[A^{\widetilde{\mathbf{D}}^{\mathbf{O}}_C(SK,\cdot),\mathcal{O}}(PK,C)=b]\leq 1/2+\mu(n)\right]\geq 1-\mu'(n),$$

*where for all queries $C' \neq C$ it holds that $\widetilde{\mathbf{D}}^{\mathbf{O}}_C(SK, C') = \mathbf{D^O}(SK, \cdot)$, and on the query $C' = C$ we have $\widetilde{\mathbf{D}}^{\mathbf{O}}_C(SK, C') = \perp$.*

*Sketch.* In order to prove the CCA#2 security of the proposed PKEP, we show how an adversary can effectively simulate the decryption oracle, $\widetilde{\mathbf{D}}^{\mathbf{O}}(SK, \cdot)$. Therefore, we can convert adversaries that make use of decryption oracles into adversaries that don't. From there it suffices to argue that all adversaries with no access to a decryption oracle are ineffective at breaking the security of the scheme. This is done by relying on the semantic security of the PKEP defined by $\mathbf{O}$ (as was shown in Theorem 18). We outline the decryption oracle simulator and its proof of correctness. The remaining details follow from standard techniques.

Let $C = (c_0, c_1, ..., c_n) \leftarrow \mathbf{E}(PK, B, R)$ be the challenge ciphertext that the adversary $A$ is given, where for each $i$ we have $|c_i| = 3n$. We modify the adversary so that every time it makes a query $C' = (c'_0, ..., c'_n)$ to the decryption oracle $\widetilde{\mathbf{D}}^{\mathbf{O}}(SK, \cdot)$ it checks to see if there exists an $n$-bit string $R' = (r'_1, ..., r'_n)$, and a bit $r'_0$ with the property that the adversary has previously made the set of query/responses $Q = \{(<\mathbf{e}, PK, r'_i, R'+i>, c'_i) | i \leq n\}$. In such a case the adversary uses $r'_0$ as the simulated reply of the decryption oracle, and otherwise it simulates the reply of $\perp$. Clearly, if the simulator's output is $r'_0$, then it has simulated the same response as that given by $\widetilde{\mathbf{D}}^{\mathbf{O}}(SK, C')$. Alternatively, if the simulator's output is $\perp$ then the simulated response is correct with probability that is exponentially close to 1, as is shown by the following cases.

If it's the case that for each $c_i'$ that makes up $C'$ the adversary has previously made a query/response $(< \mathbf{e}, PK, *, * >, c_i')$, but the set of such query/responses does not equal $Q$ then it is necessarily the case that $\perp$ is the correct output, as $\mathbf{e}(pk, \cdot, \cdot)$ is a one-to-one function. Alternatively, consider the smallest $i \leq n$ for which the adversary has not previously made a query/response $(< \mathbf{e}, PK, *, * >, c_i')$, in this case we will consider three possible sub-cases.

1. The first case is that the adversary has made a previous query $\mathbf{w}(PK, *) = (\bar{c}_1, ..., \bar{c}_n)$ where $c_i' = \bar{c}_j$ for some $j$. In this case the probability that there will exist an $R$ such that $c_i' = \mathbf{e}(PK, *, R + i)$ and $c_{i+1}' = \mathbf{e}(PK, *, R + i + 1)$ is exponentially small as the random bits used to generate the encryption $c_i'$ by the oracle $\mathbf{w}$ were chosen uniformly at random.
2. The second case is that there is a $j \leq n$ for which $c_i' = c_j$. We consider two sub-cases:
   (a) If $\{c_\ell' | \ell \leq n\} \subseteq \{c_\ell | \ell \leq n\}$ then the simulated response of $\perp$ is correct, as there is definitely an error if $C \neq C'$ because there will necessarily be a value $k$ such that $c_k' \neq \mathbf{e}(pk, *, R + k)$. Otherwise, if $C = C'$ then the correct response is $\perp$ as this is an invalid query.
   (b) If $\{c_\ell' | \ell \leq n\} \nsubseteq \{c_\ell | \ell \leq n\}$, then let $k$ be the value for which $c_k' \notin \{c_\ell | \ell \leq n\}$. We assume that $j < k$ (note the case where $j > k$ follows a similar argument), and note that the probability that the adversary can find a $c_k'$ such that $c_k' = \mathbf{e}(PK, *, R + k - j)$ is negligible, since the adversary has no knowledge of the randomly selected string $R$ that was used in the creation of $C$.
3. If neither sub-case above applies, then there is no reason to expect that for any of the $c_i'$ in $C'$ that there exist any values $b$ and $r$ such that $\mathbf{e}(PK, b, r) = c_i'$. This is because the probability of such an event is exponentially small, as $\mathbf{e}(PK, *, *)$ is a random one-to-one, length-tripling function, and so the probability of finding a value $c$ in the range of $\mathbf{e}(PK, *, *)$ without directly querying $\mathbf{e}$ or $\mathbf{w}$, or making use of the ciphertexts that constitute $C$ is negligible.

$\square$

### H.2 A Significant Restriction on Constructions

The odd characteristic of the CCA#2 construction in Section H.1 is that $\mathbf{D}^{\mathbf{O}}$ queries $\mathbf{e}$. That is the *decryption* algorithm of the PKEP that is CCA#2 secure must call the *encryption* oracle, $\mathbf{e}$, of the semantically secure PKEP. This is a rather unnatural property and it is not an obvious requirement to achieve CCA#1 security.

Our final assumption on the type of constructions we consider is that our constructions will not perform these types of queries, and this is the first assumption that places a significant restriction on the class of constructions we consider. This is formalized below.

**Assumption 25.** *We will assume that all of the purportedly CCA#1 secure PKEPs,* $(\mathbf{G}^{\mathbf{O}}, \mathbf{E}^{\mathbf{O}}, \mathbf{D}^{\mathbf{O}})$*, that we consider in this chapter have the property that* $\mathbf{D}$ *does not make queries to the sub-oracle* $\mathbf{e}$*. In other words, we can think of our PKEPs as* $(\mathbf{G}^{\mathbf{O}}, \mathbf{E}^{\mathbf{O}}, \mathbf{D}^{\mathbf{g},\mathbf{d}})$

## I A CCA#1 Adversary to Break our Class of PKEPs

In the remaining sections of this chapter we fix an arbitrary PKEP construction $\mathcal{EP} = (\mathbf{G}, \mathbf{E}, \mathbf{D})$ that satisfies all of the assumptions in Section H. We will demonstrate how to construct an adversary that will break the CCA#1 security of $\mathcal{EP}$ given a random oracle $\mathcal{O} \leftarrow \Upsilon$. More specifically, most of the work in this section — in fact most of the work in this chapter — goes towards proving the following theorem.

**Theorem 26.** *There exists a CCA#1 adversary $A = (A_1, A_2)$ for which it's the case that for all sufficiently large $n$:*

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

A simple averaging argument gives the following corollary.

**Corollary 27.** *There exists an adversary $A = (A_1, A_2)$ for which it's the case that for all sufficiently large $n$:*

$$\Pr_{\mathcal{O}=(\mathbf{O},R)\leftarrow\Upsilon}\left[\Pr_{\substack{S\in_R\{0,1\},M\in_R\{0,1\}\\ R\in_R\{0,1\}^{n^{\rho_2(n)}}\\ (PK,SK)\leftarrow G^{\mathbf{O}}(S)\\ C\leftarrow\mathbf{E}^{\mathbf{O}}(PK,M,R)}}\left[A_1^{\mathbf{D}^{\mathbf{O}}(SK,\cdot),\mathcal{O}}(PK)\rightarrow\sigma; A_2^{\mathcal{O}}(\sigma,C)=M\right]\geq 3/4\right]\geq 1-4/n.$$

In the remainder of this chapter we will describe how to construct an adversary $A = (A_1, A_2)$ that will satisfy Theorem 26. The first part of the adversary, $A_1$, will have to perform a rather large probabilistic experiment. In order to aid in the presentation and understanding of this experiment we will break it up and present it in three separate parts, named $\textsc{Exp}_1, \textsc{Exp}_2$ and $\textsc{Exp}_3$. The primary output of the sequence of experiments is a new secret-key $SK'$ and set of queries and responses. These, in conjunction with a modified decryption algorithm $\widehat{\mathbf{D}}$, will allow , with high probability, the second part of the adversary, $A_2$, to decrypt the challenge ciphertext, $C \leftarrow \mathbf{E}^{\mathbf{O}}(PK, M, R)$.

We note that at points in the description of $\textsc{Exp}_1, \textsc{Exp}_2$ and $\textsc{Exp}_3$ we make use of knowledge of $SK$, and this may appear troubling, as we intend for $A_1$ to execute these experiments, and yet it has no access to $SK$. This is not a problem because $SK$ is only used in the experiments to calculate $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ and $A_1$ can easily compute such values by using its decryption oracle.

In the following three sections of this chapter, we will go over the three parts of the experiment in depth, but we will first give a brief overview of the overall goals of the combined experiments and then explain the individual aims of the of the three parts of the experiment.

One of the strongest intuitions behind the notion of CCA#1 security, is that the decryption oracle provided to the first part of the adversary should not leak any information about the secret-key $SK$ that corresponds to the adversaries public-key $PK$, for if the decryption oracle leaks such information, then it may be passed along to the second part of the adversary and used to help decrypt the challenge ciphertext $C$. Since $A_1$ does not have access to the challenge ciphertext, it cannot use the decryption oracle to learn any information specifically about it. The main idea behind the proof of Theorem 26, is that a decryption oracle for a proposed construction of a CCA#1 PKEP, $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ that satisfies the restrictions of Section H, and which builds its security from a black-box semantically secure PKEP will have no choice but to possibly leak information about the secret-key. We remind the reader that in our model the adversary is a computationally unlimited machine, whose only complexity bound is on the number of queries it makes to the oracle $\mathcal{O}$ and the decryption oracle. Therefore, any security in the purported CCA#1 secure PKEP $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ must come from the (black-box) semantically secure PKEP that is given.

To describe our intuition about why information about $SK$ is leaked by the decryption oracle, we consider a proposed construction $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ that makes use of the semantically secure PKEP $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ that is a sub-oracle in a randomly chosen $\mathcal{O}$. Given that the security must be built from $\mathbf{O}$, it seems likely, if not necessary, that $\mathbf{E}$ will need to query $\mathbf{e}$ and embed its responses into the ciphertext that $\mathbf{E}$ will output. Similarly, $\mathbf{D}$ will likely need to query $\mathbf{d}$ to decrypt the response to the queries that $\mathbf{e}$ made and embedded in the ciphertext. Now in order for all of these queries to $\mathbf{e}$ and $\mathbf{d}$ to be secure and provide some sort of secrecy for $\mathbf{E}$ and $\mathbf{D}$, the algorithm $\mathbf{G}$ we will need to provide $\mathbf{E}$ and $\mathbf{D}$ with matching public-/secret-key pairs $(pk, sk)$ that are generated by $\mathbf{g}$ and which are embedded into a public-/secret-key pair $(PK, SK)$. Therefore, it is the secret-keys from $\mathbf{g}$ that are embedded into the secret-keys $SK$ by $\mathbf{G}$ that all of the security is boot-strapped from. Our experiment will show how to use the decryption oracle to retrieve many of the secret-keys from $\mathbf{g}$ that have been embedded into $SK$, by the use of the decryption oracle. Once we have retrieved these values, we can reconstruct a new secret-key $SK'$, and pass it to the second part of the adversary, $A_2$, and it can use this new secret-key to decrypt the challenge ciphertext.

We can now concentrate on providing intuition for how we retrieve a value $sk$, that was generated in the pair $(pk, sk)$ by $\mathbf{g}$, that is embedded into a secret-key $SK$, that was generated in the pair $(PK, SK)$ by $\mathbf{G}$. In order to do this, we will first need to find the value $pk$, which is most likely embedded into $PK$. In fact this is the main goal of $\textsc{Exp}_1$: to determine those values $pk$ that are embedded into $PK$, which might have corresponding values $sk$ embedded into $SK$. We find such $pk$ by encrypting many random messages under $\mathbf{E}$

by executing $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen $M$ and $R$, and looking at the queries to $\mathbf{e}$ that are made during these executions, we can find those values of $pk$ that are likely to be embedded into $PK$.

The goal of $\text{Exp}_2$ is to take those values $pk$ that were retrieved in $\text{Exp}_1$ and flagged as being embedded in $PK$, and find their corresponding secret-keys $sk = \mathbf{g}^{-1}(pk)$. In order to do this we make use of the oracle $\mathbf{w}$ and the main restriction on our class of constructions. The idea is that if $pk$ is embedded in $PK$ and $sk$ is embedded in $SK$, then it is likely that for some values $b$ and $r$ an encryption $c = \mathbf{e}(pk, b, r)$ will get embedded into a ciphertext $C \leftarrow \mathbf{E}^{\mathbf{O}}(PK, M, R)$ for some values $M$ and $R$. Further, the same value $c$ will likely be decrypted by a query $\mathbf{d}(sk, c) = b$ when we execute $\mathbf{D}^{\mathbf{O}}(SK, C)$. Further, presumably the value $b$ should have some effect on the outputted result of $\mathbf{D}(SK, C)$.

Now, ignoring for the moment how we might do such a thing, we note that if during the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ we were to respond to the query $\mathbf{e}(pk, b, r)$ with the value $c' = \mathbf{e}(pk, b, r')$, then we would likely end-up with a new ciphertext $C'$ where $c'$ is embedded in $C'$. Further, it is likely to have the property that $\mathbf{D}^{\mathbf{O}}(SK, C) = \mathbf{D}^{\mathbf{O}}(SK, C')$, as there is no way for the $\mathbf{D}$ to confirm if $\mathbf{e}(pk, b, r)$ is equal to $c$ or $c'$, due to the fact that $\mathbf{D}$ cannot query $\mathbf{e}$ because of the restriction specified in Section H.2, and $\mathbf{d}(sk, c') = \mathbf{d}(sk, c) = b$. Now suppose during the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ that we were to respond to the query $\mathbf{e}(pk, b, r)$ with the response $c'' = \mathbf{e}(pk, 1-b, r')$, then we would likely end up with a new ciphertext $C''$, where $c''$ is embedded in $C''$. However, now it is not clear that $\mathbf{D}^{\mathbf{O}}(SK, C) \overset{?}{=} \mathbf{D}^{\mathbf{O}}(SK, C'')$, as we know the value returned by $\mathbf{d}(sk, c')$ is $1 - b$, and this may have an effect on the outcome of the decryption of $C''$. The important observation is that if it is likely that $\mathbf{D}^{\mathbf{O}}(SK, C) \neq \mathbf{D}^{\mathbf{O}}(SK, C'')$, then this gives us a method for simulating $\mathbf{d}(sk, \cdot)$. This is done as follows: if we wish to compute $\mathbf{d}(sk, \tilde{c})$ then we execute $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ and respond to the query $\mathbf{e}(pk, b, r)$ with the value $\tilde{c}$; at the end of the execution we retrieve its output $\tilde{C}$, and use the decryption oracle to compute $\mathbf{D}^{\mathbf{O}}(SK, \tilde{C})$; if the result is $M$ then we believe the response to $\mathbf{d}(sk, \tilde{c})$ to be $b$, and otherwise we believe it to be $1 - b$. If we combine this effective method of simulating $\mathbf{d}(sk, \cdot)$ with the decryption oracle $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$, with the encrypted value of $sk$ that can be gotten through a query to $\mathbf{w}$, then we see that $sk$ is retrievable by the adversary.

The goal of $\text{Exp}_2$ is to retrieve the values of as many secret-keys $sk$ that are embedded into $SK$, as is possible. This is done by using techniques based on the ideas just presented. Unfortunately, the intuitive description just presented skipped over many possible subtleties and these have to be handled, so the resulting experiment is more complicated than the above intuitive explanation might suggest.

Finally, once we have retrieved the different secret-keys $sk$ that are embedded into $SK$, then we will reconstruct a new secret-key $SK'$ that can be used by the adversary to decrypt the challenge ciphertext. This is not a simple task, for although we may have many of the secret-keys $sk$ that are embedded into $SK$, we have no idea how to take them and reconstruct a new secret-key $SK'$, that embeds the $sk$ values. Intuitively, the algorithm $\mathbf{G}$ may be quite complicated, and so it is not at all clear how one can combine the secret-keys retrieved together into a new secret-key $SK'$. In order to do this use a brute-force search to find an oracle $\widehat{\mathcal{O}} = (\widehat{\mathbf{O}}, \widehat{R})$ that could have been produced by the random process $\Upsilon$, such that on the vast majority of queries $\mathcal{O}$ and $\widehat{\mathcal{O}}$ have the same responses; and for which there exists an $S$ such that $(PK, SK') \leftarrow \mathbf{G}^{\widehat{\mathbf{O}}}(S)$. Because of the strong correlation between $\mathbf{O}$ and $\widehat{\mathbf{O}}$ it will be the case that on most inputs $\mathbf{D}^{\mathbf{O}}(SK, \cdot) = \mathbf{D}^{\widehat{\mathbf{O}}}(SK', \cdot)$, and this will permit the adversary to decrypt the challenge ciphertext. The goal of $\text{Exp}_3$ is to find $SK'$ and the oracle $\widehat{\mathcal{O}}$.

We stress that the intuitive description just given hides many subtleties, and thus the experiments described are more complicated than it might appear they need to be.

Now that the intuitive descriptions have been given, we can begin with the proof of Theorem 26. We will begin with some necessary preliminaries.

## I.1 The Adversary's Environment

In performing the experiment described in Theorem 26 the first part of the adversary, $A_1$, is initially given access to a randomly selected oracle, $\mathcal{O} = (\mathbf{O}, R)$, a decryption oracle $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ and a randomly generated public-key, $PK$. We consider these the *environment* that the first part of the adversary has to work in. To facilitate the exposition of the experiments, we will need a way to efficiently describe the experiment that generates the random environment of the adversary, as well as a simple method for describing the environment. Therefore, we use the following experiment to describe the random generation of an environment:

```
Env(n)
(1)    $\mathcal{O} = (\mathbf{O}, R) \leftarrow \Upsilon$
(2)    $S^* \in_R \{0,1\}^n$
(3)    $(PK, SK) \leftarrow \mathbf{G}^{\mathbf{O}}(S^*)$
(4)    Output $\mathbf{Env}_n = (\mathcal{O}, PK, SK, S^*)$
```

We note that the selection of the challenge ciphertext is not included in this environment, as it is not needed until we describe the second part of the adversary, which will be much later in the chapter.

## I.2 Some Necessary Constants

Throughout the description of $\textsc{Exp}_1, \textsc{Exp}_2$ and $\textsc{Exp}_3$, we will need to introduce several constants that depend on the PKEP construction $(\mathbf{G}, \mathbf{E}, \mathbf{D})$ but nothing else. We label these constants $\alpha_0, \cdots, \alpha_6$, and intuitively we think of them as having the property that $\alpha_i << \alpha_{i+1}$ for $0 \leq i < 6$, and $\alpha_6 > s > q$: remembering that $n^q$ is the number of queries that any of the algorithms $\mathbf{G}, \mathbf{E}$ or $\mathbf{D}$ will make when given an input consistent with security parameter $n$; and $n^s$ is an upper-bound on the size of the largest query that the algorithms $\mathbf{G}, \mathbf{E}$ or $\mathbf{D}$ will make when given an input consistent with security parameter $n$.

Formally, it is sufficient that the following inequalities hold for all sufficiently large $n$:

- For each i where $0 \leq i \leq 6$:$\alpha_i \geq 20\Pi_{(i+1 \leq j \leq 6)}\alpha_j \cdot (qs)$
- $s \geq 20 \cdot q$

It should be noted that these inequalities ensure that all of the inequalities stated in this chapter hold for sufficiently large $n$, but they are not tight. Since this result is a negative one, there is not much to be gained by looking for tight bounds to these inequalities, even though, as will be made clear throughout the chapter, they have a large effect on the running time of the constructed adversary. Further, we note that given $q$, it is easy to choose $s$ and each of the $\alpha_i$s so that they satisfy these inequalities.

## I.3 Small and Large Queries and Surprising Responses

We need to cover several more useful definitions before describing $\textsc{Exp}_1, \textsc{Exp}_2$ and $\textsc{Exp}_3$. In this section we introduce the notion of small, large and possibly surprising queries and the notion of surprising responses.

Let us intuitively describe the notion of small and large queries, and surprising responses before we formalize them. Due to the definition of the random process $\Upsilon$, it is intuitively unlikely that given a random oracle $\mathcal{O} \leftarrow \Upsilon$ that one can find a pair of values $sk$ and $c$ such that response to the query $\mathbf{d}(sk, c)$ is in $\{0,1\}$, when there has been no previous queries to $\mathbf{g}, \mathbf{e}$ or $\mathbf{w}$ to suggest that $c$ has a valid decryption. This is true so long as the sizes of $sk$ and $c$ are large enough, but one could find such a $c$ with a reasonable probability if their sizes were small enough. Alternatively, if we were able to find such pair $sk$ and $c$ where $\mathbf{d}(sk, c) \in \{0,1\}$, without querying $\mathbf{g}, \mathbf{e}$ or $\mathbf{w}$ and $sk$ and $c$ were relatively large, then the response to the query $\mathbf{d}(sk, c)$ is surprising, as we would have expected it to be $\perp$.

We will need to formalize the above notions because, while we would expect the algorithms $\mathbf{G}, \mathbf{E}$ and $\mathbf{D}$ to make queries to their oracles that have a size roughly proportional to the algorithms inputs (i.e. if the input to $\mathbf{G}$ is of size $n$, we would expect all of its queries to $\mathbf{O}$ to be of size $\Theta(n)$), there is no such formal requirement. And, while it is hard to see how making small queries could be of benefit in making $\mathbf{G}, \mathbf{E}$ and $\mathbf{D}$ secure, it possibly might.

Therefore, in order to retain our intuition about the likelihood of the algorithms $\mathbf{G}, \mathbf{E}$ and $\mathbf{D}$ making queries that have surprising responses, we will formalize the notion of small and large queries. In order to do this, we use the first constant we defined $\alpha_0$. We remind the reader that the definition of $\alpha_0$ was dependent on the definition of $\mathbf{G}, \mathbf{E}$ and $\mathbf{D}$ (namely $q$ and $s$, where $n^q$ represents the number of queries made by each algorithm, and $n^s$ represents the size of the largest query made), and now give the formal definitions for the concepts we have described.

**Definition 28.** *Given an environment* $\mathbf{Env}_n = (\mathcal{O}, PK, SK, S^*) \leftarrow Env(n)$ *we define a set of* small *queries: We say that a query* $(o, q)$ *to an oracle* $\mathcal{O} \leftarrow \Upsilon$ *is small if* $|q| \leq \lceil v_o \cdot \alpha_0 \log_n \rceil$ *where* $v_{\mathbf{g}} = 1$*,* $v_{\mathbf{e}} = 5$*,* $v_{\mathbf{d}} = 4$*,* $v_{\mathbf{u}} = 6$ *and* $v_{\mathbf{w}} = 4$*. A query that is not* **small** *is called* **large***.*

*We note that the* $v_o$ *values are chosen so that they correspond to calls to* $\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}$ *and* $\mathbf{u}$ *with respect to the same sized string* $sk$ *where* $|sk| \leq \alpha_0 \log n$*, or its corresponding string* $pk = \mathbf{g}(sk)$*.*

**Definition 29 (Possibly Surprising Queries & Surprising Responses to d).** *For a specified random experiment (resp. process or algorithm), for each* $n$ *and each* $sk \in \{0,1\}^n$*, we say a query* $(\mathbf{d}, sk, c)$ *made during the experiment (resp. process or algorithm) is* **possibly surprising***, if it is large and if when the query* $\mathbf{d}(sk, c)$ *was made none of the following query/response cases enumerated below had previously been made in the experiment (resp. process or algorithm) specified:*

1. $(< \mathbf{e}, \mathbf{g}(sk), b, * >, c)$ *together with* $(< \mathbf{g}, sk >, \mathbf{g}(sk))$
2. $(< \mathbf{w}, \mathbf{g}(sk), * >, c_1, ..., c_n)$ *where* $c = c_i$ *for some* $i$*.*
3. $(< \mathbf{d}, sk, c >, *)$*.*
4. $(< \mathbf{u}, \mathbf{g}(sk), c >, *)$*.*

*We say that the query/response* $(< \mathbf{d}, sk, c >, b)$ *for* $b \in \{0,1\}$ *is* **surprising** *if the query* $(\mathbf{d}, sk, c)$ *was* **possibly surprising***.*

Next we define a similar notion of possibly surprising queries to $\mathbf{u}$. We will be interested in queries to $\mathbf{u}(pk, c)$ where there has already been a query $\mathbf{e}(pk, b, r)$ that has been made. The purpose of the query to $\mathbf{u}(pk, c)$ is therefore to determine if $\mathbf{g}^{-1}(pk)$ is well defined. Therefore, we would consider a query to $\mathbf{u}(pk, c)$ to be surprising if there had not been a previous query $\mathbf{g}(*) = pk$. This is formalized below.

**Definition 30 (Possibly Surprising Queries & Surprising Responses to u).** *] Given a specified random experiment (resp. process or algorithm), for each* $n$ *and each* $pk \in \{0,1\}^{3n}$*, we say a query* $(\mathbf{u}, pk, c)$ *made during the experiment is* **possibly surprising***, if when the query* $\mathbf{u}(pk, c)$ *is made neither of the following query/responses has previously been made in the experiment, process or algorithm specified:*

1. $(< \mathbf{g}, * >, pk)$*.*
2. $(< \mathbf{u}, pk, c >, *)$*.*

*We say that the query/response* $(< \mathbf{u}, pk, c >, b)$ *is* **surprising** *if the query* $(\mathbf{u}, pk, c)$ *was* **possibly surprising** *and* $b = \top$*.*

## J $\text{Exp}_1$: Finding Likely Queries and Embedded Public-keys

Given a random variable $\mathbf{Env}_n \leftarrow Env(n)$ that denotes the adversary's initial environment, we can now describe the first part of the experiment the adversary will perform. The first part of the experiment takes the random variable $\mathbf{Env}_n$ as an input. The experiment's goal is to retrieve three useful sets. The first set, $SQ$, contains all of the small queries to $\mathbf{O}$. The second set, $KS$, contains (w.h.p) many of the $pk$ generated by $\mathbf{g}$ that are likely to have been embedded into $PK$ given to us by the environment. By this we mean that an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for randomly chosen $M$ and $R$, is likely to make queries of the form $\mathbf{e}(pk, *, *)$. Finally, the third set, $\mathcal{E}$, contains (w.h.p.) all of the queries that are "likely" to be made to $\mathbf{e}$ during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for a randomly chosen $M$ and $R$.

We are interested in these sets because they intuitively capture all of the information about the oracle $\mathcal{O}$ that is effectively embedded into $PK$ (or in the case of small queries, information about $\mathcal{O}$ that is easily guessed). In particular, in the third part of the experiment, when we go to find an oracle $\mathcal{O}'$ that is a close approximation to $\mathcal{O}$ we would like it to be the case that it is likely that $\mathbf{E}^{\mathbf{O}'}(PK, M, R) = \mathbf{E}^{\mathbf{O}}(PK, M, R)$ for a randomly chosen $M$ and $R$, and this is much less likely if $\mathcal{O}'$ is not consistent with these sets.

Additionally, we want to find the public-keys in $KS$, so that we might find their corresponding secret-keys in the second part of the experiment, as previously outlined.

We remind the reader that $n^q$ specifies the number of queries performed by $\mathbf{G}$, $\mathbf{E}$ and $\mathbf{D}$ on inputs consistent with a security parameter of $n$.

We briefly describe the experiment. First, the experiment retrieves every *small* query/response pair of the sub-oracle $\mathbf{O}$, by making every such query. Next, it samples a large number of random encryptions with $\mathbf{E}^{\mathbf{O}}(PK,\cdot,\cdot)$ and monitors the oracle calls to $\mathbf{e}$ and $\mathbf{g}$ that are made during these encryptions. On query/responses $(<\mathbf{e},pk,*,*>,c)$ the experiment decides that it is likely that $pk$ was embedded into $PK$ by determining if there were no previous queries to $\mathbf{g}(*)=pk$ and ensuring that $\mathbf{g}^{-1}(pk)$ is well defined by ensuring that $\mathbf{u}(pk,c)=\top$. If the experiment finds more than $n^q$ public-keys that it thinks were embedded into $PK$, then it halts, as $\mathbf{G}^(S^*)$ queried $\mathbf{g}$ at most $n^q$ times, and so there are at most $n^q$ public-keys from $\mathbf{g}$ embedded into $PK$.

```
Exp_1(Env_n = (O, PK, SK, S*))
(1)   Make every small query in O and let SQ be the set of resulting query/response pairs.
(2)    for $i = 1$ to $n^{2\alpha_1}$
(3)       $M_i \in_R \{0,1\}$
(4)       $R_i \in_R \{0,1\}^{n^{\rho_2}}$
(5)       $C_i \leftarrow \mathbf{E}^{\mathbf{O}}(PK, M_i, R_i)$
(6)   Let $\mathcal{E} = \{(<\mathbf{e},pk,m,r>,c) | (<\mathbf{e},pk,m,r>,c)$ queried in the for loop $\}$.
(7)   Compute $KS = \{pk |$ The query $(<\mathbf{e},pk,m,r>,c)$ is large, made in the for loop, there
      is no query $(<\mathbf{g},*>,pk)$ made in the for loop prior to it and $\mathbf{u}(pk,c)=\top$ $\}$
(8)   Let $\mathcal{IQ} \leftarrow \mathcal{E} \cup SQ$
(9)   If $|KS| > n^q$ HALT
(10)  Output $(\mathbf{Env}_n, KS, \mathcal{IQ}, \mathcal{E}, SQ)$.
```

## J.1 A Running Example

In order to attempt to make the different portions of $\textsc{Exp}_1, \textsc{Exp}_2$ and $\textsc{Exp}_3$ more concrete, we are going to introduce a PKEP $(\mathfrak{G},\mathfrak{E},\mathfrak{D})$ and show the types of results we would expect the experiment to output when given such a PKEP.

To begin with we will present the PKEP which we will use in our running example. We note it is designed to highlight the need for some of the different parts of $\textsc{Exp}_1, \textsc{Exp}_2$ and $\textsc{Exp}_3$, which may not be immediately obvious, and this makes the example slightly convoluted.

**Definition 31.** *For each $n\in\mathbb{N}$ we define:*

- $\mathfrak{G}^{\mathbf{O}}(S)$*: We let $S=(S_0,...,...,S_9)$, where each $S_i\in\{0,1\}^n$. The algorithm queries $\mathbf{g}(S_i)=pk_i$ for each $i$, $0\le i\le 6$. It then computes $k_1=\mathbf{e}(pk_6,0,S_8)$ and $k_2=\mathbf{e}(pk_6,0,S_9)$, and outputs $PK=(pk_0,..,pk_5,pk_6,S_8,S_9)$ and $SK=(sk_0=S_0,...,sk_5=S_5,sk_6=S_6,k_1,k_2)$.*
- $\mathfrak{E}^{\mathbf{O}}(PK,M,R)$*: We let $PK=(pk_0,...,pk_5,pk_6,S_8,S_9)$, $M\in\{0,1\}$ 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 $1\le i\le 5$. Compute $k_1=\mathbf{e}(pk_6,0,S_8)$ and $k_2=\mathbf{e}(pk_6,0,S_9)$. If $R_6$ is the bit-string of all zeros, then we query $\mathbf{e}(pk_0,M,R_0)=c_0$ and output $C=(0,k_1,k_2,\underbrace{0\ldots0}_{3n},\underbrace{0\ldots0}_{3n},\underbrace{0\ldots0}_{3n},\underbrace{0\ldots0}_{3n},c_0)$,*
  *Otherwise, output $C=(1,k_1,k_2,c_1,c_2,\ldots,c_5)$.*
- $\mathfrak{D}^{\mathbf{O}}(SK,C)$*: We let $C=(b,k_1',k_2',c_1,c_2,\ldots,c_5)$ where $b\in\{0,1\}$, $k_1',k_2'\in\{0,1\}^n$ and each $c_i\in\{0,1\}^n$. We let $SK=(sk_0,...,sk_5,sk_6,k_1,k_2)$ where each $sk_i\in\{0,1\}^n$.*
  *If $k_1'\ne k_1$ or $k_2'\ne k_2$ output $\perp$.*
  *Otherwise, if $\mathbf{d}(sk_6,k_1')\ne 0$ or $\mathbf{d}(sk_6,k_2')\ne 0$ 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 $i\le 5$, and output $Majority(M_1,...,M_5)$,*

**Running Example 1.** *We now imagine an execution of $\mathbf{Env}_n\leftarrow Env(n)$ and $E_1\leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ when the PKEP that is being used in the experiments is $(\mathfrak{G},\mathfrak{E},\mathfrak{D})$ as defined in Defn. 31. To begin with, let's consider the execution $Env(n)$ that generates a random oracle $\mathcal{O}=(\mathbf{O},R)$ and a seed $S^*$ for which $(PK,SK)\leftarrow$*

*$\mathfrak{G}^{\mathbf{O}}(S^*)$, where $PK = (pk_0, ..., pk_6, S_8, S_9)$ and $SK = (sk_0, ..., sk_6, k_1, k_2)$. This sets up and defines all of the information that the adversary $A_1$ will have access to in Theorem 26, namely the string $PK$ and the oracles $\mathcal{O}$ and $\mathfrak{D}^{\mathbf{O}}(SK, \cdot)$.*

*Next, the adversary will execute* $\text{Exp}_1(\mathbf{Env}_n)$*, which will compute the sets $KS$, $\mathcal{IQ}$, $\mathcal{E}$ and $SQ$. We note that $KS$ is most likely equal to $\{pk_1, ..., pk_6\}$. This is because we are guaranteed that there will be queries $\mathbf{e}(pk_i, *, *)$ for $1 \leq i \leq 6$ during an execution of $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$, but it is highly unlikely that a query $\mathbf{e}(pk_0, *, *)$ will ever be made, as such a query is only made if $R_6$ is the string of all zeros, where $R = (R_1, ..., R_6)$. Therefore, for the remainder of the running example we will assume that $KS = \{pk_1, ..., pk_6\}$.*

*Next, we note that while $\mathcal{E}$ will contain a large number of query/responses that are made during each execution of $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$, for randomly chosen $M$ and $R$, it will necessarily contain the query/responses $(< \mathbf{e}, pk_6, 0, S_8 >, k_1)$ and $(< \mathbf{e}, pk_6, 0, S_9 >, k_2)$. This is because in every execution of $\mathfrak{E}^{\mathbf{O}}(PK, *, *)$ it is the case that the corresponding queries are made.*

*We note that through this execution we have retrieved many of the public-keys generated by $\mathbf{g}$ that have been embedded into $PK$. Further, we have retrieved the information $S_8$ and $S_9$ that has been embedded into $PK$. The goal of the second part of the experiment will be to find important information that is embedded into $SK$: many of the secret-keys that correspond to the public-keys in the set $KS$.*

## J.2 Bounding the Probability of Bad Events in $\mathbf{Exp}_1$

The next lemma demonstrates that the set $KS$ constructed in $\text{Exp}_1$ is a subset of the public-keys that were discovered through calls to $\mathbf{g}$ during the execution of $\mathbf{G}^{\mathbf{O}}(S^*)$, unless a surprising query/response to $\mathbf{u}$ is made. We will later show that, unsurprisingly, such surprising query/response events are rare.

**Lemma 32.** *If after a random experiment $\mathbf{Env}_n \leftarrow Env(n)$, $\text{Exp}_1(\mathbf{Env}_n)$ it's the case $KS \not\subseteq KS^*$ then a surprising query/response to $\mathbf{u}$ occurred during the experiment, where*

$$KS^* \overset{defn}{=} \{pk | \mathbf{G}^{\mathbf{O}}(S^*) \text{ makes the query } (< \mathbf{g}, * >, pk)\}.$$

*Proof.* The reasoning for this proof follows almost immediately from the definition of $KS$, $KS^*$ and surprising query/response of $\mathbf{u}$. Clearly, in order for $KS \not\subseteq KS^*$ to hold, there must be an element $pk$ that is in $KS$ and not in $KS^*$. Let's consider the addition of such an element $pk$ into $KS$. Before $\text{Exp}_1(\mathbf{Env}_n)$ placed the element $pk$ in $KS$ it performed a query/response $\mathbf{e}(pk, b, r) = c$ for some $b$ and $r$, and a query/response $\mathbf{u}(pk, c) = \top$. Finally, it ensured there were no prior query/responses during $\text{Exp}_1(\mathbf{Env}_n)$ of the form $\mathbf{g}(*) = pk$. Since $pk \notin KS^*$ there were also no query/responses $\mathbf{g}(*) = pk$ performed during $\mathbf{G}^{\mathbf{O}}(S^*)$ in $Env(n)$, which also implies no such queries were made during $Env(n)$. Therefore, the query $\mathbf{u}(pk, c)$ made by $\text{Exp}_1$ was necessarily surprising. □

The previous lemma implies that a surprising query is necessary for $\text{Exp}_1$ to halt on line 9 of its description (on page 27).

**Corollary 33.** *If after a random experiment $\mathbf{Env}_n \leftarrow Env(n)$, $\text{Exp}_1(\mathbf{Env}_n)$ there have been no surprising query/response to $\mathbf{u}$, then $|KS| \leq n^q$.*

*Proof.* Follows immediately from the previous lemma and the fact that $\mathbf{G}$ makes no more than $n^q$ queries when given a seed $S^* \in \{0, 1\}^n$ as input, and therefore $|KS^*| \leq n^q$. □

We now bound the probability that there were any surprising query/responses that have been made in the experiment so far. Because it will be useful later, we not only bound the probability that there were surprising queries to $\mathbf{u}$, but also the probability that there were surprising query/responses to $\mathbf{d}$. We obviously consider it a bad event if such surprising query/responses occur, and we label it thus.

**Lemma 34.** *For all sufficiently large $n$:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n) \\ \text{Exp}_1(\mathbf{Env}_n)}} [\mathbf{BAD}_1] \leq \frac{1}{n^{\alpha_0}}$$

*where we denote by $\mathbf{BAD}_1$ the event that there is a surprising query/response to $\mathbf{d}$ or $\mathbf{u}$ in the probabilistic experiment specified.*

*Proof.* Since, by Assumption 21, $\mathbf{G}$ does not query $\mathbf{d}$, we know there are no surprising query/responses performed in $Env(n)$. By observation of $Env(n)$ and $\textsc{Exp}_1$, there are only two places that a large surprising query can occur:

1. During queries to $\mathbf{d}$ made during the varying executions of $\mathbf{E}^{\mathbf{O}}(PK, M_i, R_i)$ in the **for loop** of $\textsc{Exp}_1(\mathbf{Env}_n)$ (lines 2–5).
2. During the queries to $\mathbf{u}$ made during the computation of the set $KS$ in $\textsc{Exp}_1(\mathbf{Env}_n)$.

First, we bound the probability of surprising query/response corresponding to the first case. There are $n^{2\alpha_1}$ executions of $\mathbf{E}^{\mathbf{O}}(PK, M_i, R_i)$ in the **for loop** of $\textsc{Exp}_1$. We bound the probability of the $i$th such execution. We consider the probability that the response to a possibly surprising query $\mathbf{d}(sk, c)$ made during the execution of $\mathbf{E}$ is actually surprising. In $Env(n)$ the execution of $\mathbf{G}^{\mathbf{O}}(S^*)$ makes at most $n^q$ queries of the form $\mathbf{e}(\mathbf{g}(sk), *, *)$ and no queries to $\mathbf{d}(sk, *)$. In the previous $i-1$ executions of $\mathbf{E}$ there were at most $(i-1)n^q$ queries of the form $\mathbf{e}(pk, *, *)$ and at most $(i-1)n^q$ queries of the form $\mathbf{d}(sk, *)$; therefore, we can bound the probability that the query $\mathbf{d}(sk, c)$ is surprising to be less than $\frac{n^{\alpha_0} - i \cdot n^q}{n^{3\cdot\alpha_0} - 2i \cdot n^q}$ which is less than $\frac{1}{n^{2\alpha_0 - 1}}$ for sufficiently large $n$. There are at most $n^q$ possibly surprising queries to $\mathbf{d}$ that are made during the $\mathbf{E}^{\mathbf{O}}(PK, M_i, R_i)$, and therefore we can bound the probability that a large surprising query/response is made to be less than $\frac{n^q}{n^{2\alpha_0 - 1}}$. Finally, there are $n^{2\alpha_1}$ executions of $\mathbf{E}$ in the **for loop** of $\textsc{Exp}_1$, so the probability of a surprising query occurring during the **for loop** is less than $\frac{n^{2\alpha_1 + q}}{n^{2\alpha_0 - 1}}$.

Next, we perform a similar counting argument to show the probability of a surprising query/response to $\mathbf{u}$ occurring during the the computation of the set $KS$ is small. In order to construct the set $KS$, $\textsc{Exp}_1(\mathbf{Env}_n)$ needs to query $\mathbf{u}(pk, c)$ for each query/response $(< pk, *, * >, c)$ that was performed in the **for loop**. There are at most $n^{q + 2\alpha_1}$ such queries to $\mathbf{u}$. Consider the $i$th such query $(\mathbf{u}, pk, c)$ that is possibly surprising. The probability that the reply is surprising is at most $\frac{n^{\alpha_0} - n^q - n^{2\alpha_1 + q} - i}{n^{3\alpha_0} - n^q - n^{2\alpha_1 + q} - i}$, which is less than $\frac{1}{n^{2\alpha_0 - 1}}$ for sufficiently large $n$. This can be explained as follows: we can be overly generous and assume that each large query that is made in $Env(n)$ or $\textsc{Exp}_n(\mathbf{Env}_n)$ reveals the image of a different element in the range of for the function $\mathbf{g} : \{0,1\}^{|pk|/3} \to \{0,1\}^{|pk|}$. There have been at most $n^q$ queries made during $Env(n)$, and at most $n^{2\alpha_1 + q}$ queries in the for loop, and at most $i$ previous queries to $\mathbf{u}$ in the determination of $KS$, and noting that $\mathbf{g}$ is a random one-to-one function subject to only these constraints, we get the bound specified above. Therefore, the probability that there is a surprising query/response during the construction of $KS$ is at most: $\frac{n^{2\alpha_1 + q}}{n^{2\alpha_0 - 1}}$, as there are at most $n^{q + 2\alpha_1}$ queries to $\mathbf{u}$.

Finally, we can bound the probability that there is a surprising query to $\mathbf{d}$ or $\mathbf{u}$ to be less than $\frac{2n^{2\alpha_1 + q}}{n^{2\alpha_0 - 1}} \leq \frac{1}{n^{\alpha_0}}$ for sufficiently large $n$. □

Finally, we wanted the set $\mathcal{E}$ to contain all of the queries that were likely to occur during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen $M$ and $R$. This was done in $\textsc{Exp}_1$ by sampling. It executes $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ many times and observes what queries are made. Below we formalize the event that we do not find all of the likely queries. If we do not find them, then our adversary will not likely be able to find an alternate secret-key $SK'$, and therefore we denote it as a bad event. We also show that it is extremely unlikely that we do not find all of the likely queries.

**Lemma 35.** *For all sufficiently large $n$ and all $\mathbf{Env}_n \leftarrow Env(n)$:*

$$\Pr_{\textsc{Exp}_1(\mathbf{Env}_n)} [\mathcal{CQ} \not\subseteq \mathcal{E}] \leq 1/2^n,$$

*where*

$$\mathcal{CQ} = \{(\mathbf{e}, pk, b, r) | \Pr_{\substack{M \in \{0,1\} \\ R \in \{0,1\}^{n^{\rho_2}}}} [\mathbf{E}^{\mathbf{O}}(PK, M, R) \textit{ makes the query } (\mathbf{e}, pk, b, r)] \geq 1/n^{\alpha_1 - 2}\}$$

*$\mathcal{CQ}$ denotes the set of queries to $\mathbf{e}$ that are likely to made during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for randomly chosen $M$ and $R$.*

*Proof.* Follows easily from the Chernoff-Hoeffding bound. □

**Lemma 36.** *For all sufficiently large $n$ and all $\mathbf{Env}_n \leftarrow Env(n)$:*

$$\Pr_{\textsc{Exp}_1(\mathbf{Env}_n)}[\{pk|(\mathbf{e}, pk, *, *) \in KS\} \not\subseteq \mathcal{CK}\}] \leq 1/2^n,$$

*where*

$$\mathcal{CK} = \{pk| \Pr_{\substack{M\in\{0,1\} \\ R\in\{0,1\}^{n^{\rho_2}}}}[\mathbf{E}^{\mathbf{O}}(PK, M, R) \text{ makes any query } (\mathbf{e}, pk, *, *)] \geq 1/n^{\alpha_1 - 2}\}.$$

*$\mathcal{CK}$ denotes the set of public-keys which are likely to be used in queries to $\mathbf{e}$ during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for randomly chosen $M$ and $R$.*

*Proof.* Follows easily from the Chernoff-Hoeffding bound. □

**Definition 37.** *We will denote by $\mathbf{BAD}_2$ the event during an experiment $\mathbf{Env}_n \leftarrow Env(n)$, $\textsc{Exp}_1(\mathbf{Env}_n)$ that $\mathcal{CQ} \not\subseteq \mathcal{E}$ or $\{pk|(\mathbf{e}, pk, *, *) \in KS\} \not\subseteq \mathcal{CK}$, where the sets are defined as in Lemmas 35 and 36. Based on these same lemmas, it is easy to see that the probability of the event is less than $2/2^n$.*

## K Exp$_2$: Retrieving Embedded Secret-Keys with the Decryption Oracle

We give a high-level, informal and intuitive description of the second part of the adversary's experiment, and in later sections we will formalize all of the introduced concepts.

The goal of this part of the experiment is to retrieve a large set of secret-keys for the PKEP defined by $\mathbf{O}$ that are embedded into the adversary's secret-key $SK$. Specifically, we are interested in those secret-keys that have their corresponding public-keys in the set $KS$ that was retrieved in $\textsc{Exp}_1$ (i.e. those $sk$ for which $\mathbf{g}(sk) \in KS$). The public-keys in $KS$ are the ones that are used frequently during executions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for random $M$ and $R$, and so it stands to reason that their corresponding secret-keys might be necessary for decrypting the ciphertexts generated by calls to $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for random $M$ and $R$. We retrieve these secret-keys by using an alternate encryption algorithm, $\widehat{\mathbf{E}}$, that allows us to construct ciphertexts that are slightly perturbed versions of those produced by $\mathbf{E}$. The way in which $\widehat{\mathbf{E}}$ perturbes the ciphertexts generated by $\mathbf{E}$ is roughly as follows: $\widehat{\mathbf{E}}$ executes $\mathbf{E}$, but on oracle queries $(\mathbf{e}, pk, *, *)$ for some $pk \in KS$, it may change the response of the oracle, and thereby produce a slightly different ciphertext. By looking at how such perturbed ciphertexts decrypt, in conjunction with the use of results from queries to the sub-oracle $\mathbf{w}$, we are able to learn $\mathbf{g}^{-1}(pk)$.

We remind the reader that our overarching goal is to have $A_1$, the first part of the CCA#1 adversary, run this experiment, and so our goal must be to retrieve the secret-keys embedded in $SK$ exclusively through the use of the decryption oracle (i.e. the ability to compute $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$), and thus we must design our experiment so that is never makes direct use of $SK$ or $S^*$.

### K.1 Altering the Encryption Algorithm E

We will retrieve secret-keys corresponding to the public-keys in the set $KS$ by using $\mathbf{w}$ to generate a series of ciphertexts (for the PKEP defined by the oracle $\mathbf{O}$) that are an encrypted encodings of said secret-keys. We will then embed these ciphertexts into perturbed ciphertexts of the PKEP $\mathcal{EP} = (\mathbf{G}, \mathbf{E}, \mathbf{D})$ by the use of a modified encryption algorithm $\widehat{\mathbf{E}}$. These embeddings will be done in such a way that when the decryption oracle is fed with these perturbed ciphertexts, its output will be correlated with the decrypted value of the ciphertexts generated by $\mathbf{w}$. Once we learn this correlation, then we can predict the value of $\mathbf{g}^{-1}(pk)$ by decrypting perturbed ciphertexts with the decryption oracle.

We will introduce two modified versions of the encryption algorithm $\mathbf{E}$ in this section, the algorithms $\overline{\mathbf{E}}$ and $\widehat{\mathbf{E}}$. The algorithm $\widehat{\mathbf{E}}$ is essentially a syntactic modification of $\overline{\mathbf{E}}$ that is useful in specifying exactly how we would like to perturb the ciphertexts.

We want to construct the algorithm $\overline{\mathbf{E}}$ so that it constructs perturbed ciphertexts that the decryption oracle will not always decrypt to $\perp$. In order to do this we will make clear use of our restriction on $\mathbf{D}$ in

Assumption 25 (page 22) that requires that $\mathbf{D}$ not query the sub-oracle $\mathbf{e}$. We observe that if this restriction is upheld, then $\mathbf{D}$ does not have the ability to distinguish between different random encryptions of the same bit under the same public-key, even when $\mathbf{D}$ is given the random bits $r$ used to construct the encryptions. More specifically, for randomly chosen $b, r$ and $r'$, given the encryptions $c = \mathbf{e}(pk, b, r)$, $c' = \mathbf{e}(pk, b, r')$ and the strings $b, r$ and $r'$ an algorithm with access to $SK$, such as the algorithm $\mathbf{D}$, cannot determine if $c$ or $c'$ corresponds to the response to $\mathbf{e}(pk, b, r)$ with probability substantially better than 1/2. This is because $\mathbf{O}$ has been chosen randomly, and due to the random selection of $r$ and $r'$ it is highly unlikely that information about $c$ or $c'$ has been embedded into $SK$, and this would be the only way in which the algorithm could learn such information.

To see how the above observation relates to creating perturbed ciphertexts, let us consider an execution $\mathbf{E}^{\mathbf{O}}(PK, M, R) \rightarrow C$ that performs the query/response $(<\mathbf{e}, pk, b, r>, c)$. Now consider the same execution $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ except that on the query $(\mathbf{e}, pk, b, r)$ we replace the oracle's response with $\mathbf{e}(pk, b, r') = c'$. Let the output of the modified execution of $\mathbf{E}$ be $C'$. We should intuitively expect that $\mathbf{D}^{\mathbf{O}}(SK, C') = M$, as $\mathbf{D}$ cannot distinguish between the two different executions of $\mathbf{E}$. Of course things aren't so simple! There is an exception to this intuition: if during the execution of $\mathbf{E}$ we replace $\mathbf{e}$'s response to the query $(\mathbf{e}, pk, b, r)$ with $\mathbf{e}(pk, b, r')$, but the query $(\mathbf{e}, pk, b, r)$ was made during the execution of $\mathbf{G}^{\mathbf{O}}(S^*)$ and the values $pk, b, r$ and $\mathbf{e}(pk, b, r)$ were embedded into $SK$ then $\mathbf{D}$ might easily distinguish between the two cases, and in the case of such an embedding there is no reason to expect $\mathbf{D}$ to output $M$. Fortunately, we can handle this special case by only modifying responses to queries $(\mathbf{e}, pk, b, r)$ that are unlikely.

The algorithm $\overline{\mathbf{E}}$ is a simple modification of $\mathbf{E}$. The algorithm $\overline{\mathbf{E}}$ will simulate the execution of $\mathbf{E}$, and thus must take all of $\mathbf{E}$'s inputs as its own. In addition $\overline{\mathbf{E}}$ will take two additional inputs, both of which are sets. The first set, $KS$, describes those public-keys for which it is possible for $\overline{\mathbf{E}}$ to modify the results of encryption queries performed by $\mathbf{E}$ (we are interested in those public-keys $pk$ that are likely to be embedded into $PK$, and so we have in mind using the set $KS$ retrieved in $\textsc{Exp}_1$ as the input). The second set, $\mathcal{IQ}$, describes oracle query/responses that are immutable. That is, if a query in $\mathcal{IQ}$ is made during the simulation of $\mathbf{E}$ then the result of the query will not be modified (we have in mind providing the set $\mathcal{IQ}$ computed in $\textsc{Exp}_1$ as this input). During the execution of $\overline{\mathbf{E}}$, if there is a query $(\mathbf{e}, pk, b, r)$ where $pk \in KS$ and $(\mathbf{e}, pk, b, r) \notin \mathcal{IQ}$ then the query is modified and the result of $\mathbf{e}(pk, b, r')$, for a random $r'$, is returned instead of the anticipated $\mathbf{e}(pk, b, r)$.

This new algorithm is formally described below. In its description a sequence $\mathcal{MQ}$ (Modified Queries) is constructed, but has no effect on the algorithm. The purpose of this sequence is to help prove certain properties of this algorithm in a later lemma.

For each $n \in \mathbb{N}$ our algorithm takes as input strings $PK \in \{0,1\}^{n^{\rho_1}}$, $M \in \{0,1\}$, $R' = (R_1, R_2) \in \{0,1\}^{n^{\rho_2}+n^{qs}}$ and a set of public-keys $KS$ and a set of query/responses $\mathcal{IQ}$. Let $|R_1| = n^{\rho_1}$ and $|R_2| = n^{qs}$.

```
$\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R' = (R_1, R_2), KS, \mathcal{IQ})$
(1)   Let $\mathcal{MQ} = \lambda$ (An empty sequence)
(2)   Simulate the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R_1)$
(3)       On oracle query $(\mathbf{g}, sk)$ reply with $\mathbf{g}(sk)$.
(4)       On oracle query $(\mathbf{d}, sk, c)$ reply with $\mathbf{d}(sk, c)$.
(5)       On oracle query $(\mathbf{e}, pk, b, r)$
(6)           If $pk \notin KS$ or $(\mathbf{e}, pk, b, r) \in \mathcal{IQ}$ reply with $\mathbf{e}(pk, b, r)$.
(7)           otherwise
(8)               say $(\mathbf{e}, pk, b, r)$ is **marked for interchange**
(9)               reply with **replacement response** $\mathbf{e}(pk, b, r')$ for $r' \in_R \{0,1\}^{|pk|/3}$ (from $R_2$).
(10)              $\mathcal{MQ} \leftarrow \mathcal{MQ}, < \mathbf{e}(pk, b, r), (\mathbf{e}, pk, b, r), \mathbf{e}(pk, b, r'), (\mathbf{e}, pk, b, r') >$
(11)  Output the result of simulation
```

We note that $R_1$ is used to provide random bits for the simulation of the execution of $\mathbf{E}^{\mathbf{O}}$ and $R_2$ is used to provide the random bits for replacement queries to $\mathbf{e}$ that are made during the simulation (line 9 of $\overline{\mathbf{E}}$). Since $\mathbf{E}^{\mathbf{O}}$ will make at most $n^q$ queries of size at most $n^s$, we are guaranteed that $R_2$ contains a sufficient number of bits[9]. We now prove that with properly specified sets $\mathcal{IQ}$ and $KS$, a ciphertext $C'$ that results from $\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R', KS, \mathcal{IQ})$, for random $M$ and $R'$, is likely to to decrypt properly (i.e. $\mathbf{D}^{\mathbf{O}}(SK, C') = M$).

**Running Example 2.** *We continue the running example. We would like to ground our algorithm $\overline{\mathbf{E}}$ by considering an execution of $\overline{\mathfrak{E}}^{\mathbf{O}}(PK, M, R' = (R_1, R_2), KS, \mathcal{IQ})$, for the public-key $PK$ and sub-oracle $\mathbf{O}$ in $\mathbf{Env}_n$, and $KS$ and $\mathcal{IQ}$ in $E_n^1$ found in Running Example 1. We remind the reader, that based on Running Example 1 we have $KS = \{pk_1, ..., pk_6\}$ and that $(< \mathbf{e}, pk_6, 0, S_8 >, k_1), (< \mathbf{e}, pk_6, 0, S_9 >, k_2) \in \mathcal{IQ}$.*

*Now suppose the execution of $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$ outputs a ciphertext $C = (1, k_1, k_2, c_1, ..., c_5)$, then an execution of $\overline{\mathfrak{E}}^{\mathbf{O}}(PK, M, R' = (R, \overline{R}), KS, \mathcal{IQ})$ will output $C' = (1, k_1, k_2, c'_1, ..., c'_5)$, which is explained as follows: for each value of $i$ where $1 \leq i \leq 5$, the query $(\mathbf{e}, pk_i, M, R_i)$, with corresponding response $c_i$, computed in $\mathfrak{E}^{\mathbf{O}}(PK, M, R)$ is marked for interchange in the execution of $\overline{\mathfrak{E}}$ and replied to with a replacement response $\mathbf{e}(pk_i, M, r'_i) = c'_i$ where $r'_i$ is now a random string derived from $\overline{R}$ instead of $R$. We note that $k_1$ and $k_2$ are not modified since $(< \mathbf{e}, pk_6, 0, S_8 >, k_1), (< \mathbf{e}, pk_6, 0, S_9 >, k_2) \in \mathcal{IQ}$.*

*We note that it should not be surprising that $\mathbf{D}^{\mathbf{O}}(SK, C') = M$, as we have simply replaced random encryptions $\mathbf{e}(pk_i, M, R_i) = c_i$ with alternative encryptions $\mathbf{e}(pk_i, M, r'_i) = c'_i$. We point out that had we designed $\overline{\mathbf{E}}$ so that it modified the response to the query $\mathbf{e}(pk_6, 0, S_8) = k_1$ or $\mathbf{e}(pk_6, 0, S_9) = k_2$, then had the resulting ciphertext been given to the decryption oracle, the response would almost surely have been $\perp$.*

The next lemma demonstrates that in our experiment the ciphertexts produced by $\overline{\mathbf{E}}$ are likely to decrypt to the intuitively correct value.

**Lemma 38.** *For every $\mathbf{Env}_n \leftarrow Env(n)$ and every $\textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1} \wedge \overline{\mathbf{BAD}_2}$ holds:*

$$\Pr_{\substack{M \in \{0,1\}, R' = (R_1, R_2) \in \{0,1\}^{n^{\rho 2} + qs} \\ \overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R', KS, \mathcal{IQ}) \rightarrow C'}} [\mathbf{D}^{\mathbf{O}}(SK, C') \neq M] \leq n^{2q}/n^{\alpha_1 - 2} + n^{2q}/n^{\alpha_0},$$

*where $PK$ is defined by $\mathbf{Env}_n$ and $KS$ and $\mathcal{IQ}$ are defined in* $\textsc{Exp}_1$.

*Proof.* We construct a sub-oracle $\mathbf{O}' = (\mathbf{g}, \mathbf{e}', \mathbf{d})$ that is nearly identical to $\mathbf{O}$. The difference between $\mathbf{O}$ and $\mathbf{O}'$ are such that they guarantee that $\mathcal{O}' \overset{defn}{=} (\mathbf{g}, \mathbf{e}', \mathbf{d}, \mathbf{w}, \mathbf{u})$ could have been produced by the random process $\Upsilon$. Our goal is to show that $G^{\mathbf{O}'}(S^*) = (PK, SK)$ and that $E^{\mathbf{O}'}(PK, M, R_1) \rightarrow C'$. If these conditions hold, then by Assumption 23 it's necessarily the case that $\mathbf{D}^{\mathbf{O}'}(SK, C') = M$. However, since $\mathbf{D}^{\mathbf{O}'}(SK, C') = \mathbf{D}^{\mathbf{g}, \mathbf{d}}(SK, C')$, by our restriction on queries to $\mathbf{e}$ by $\mathbf{D}$ and the fact that $\mathbf{O}'$ and $\mathbf{O}$ differ only between $\mathbf{e}$ and $\mathbf{e}'$, we have that $\mathbf{D}^{\mathbf{O}}(SK, C') = M$, proving the lemma.

Let $(< c_1, \gamma_1, c'_1, \gamma'_1 >, ..., < c_i, \gamma_i, c'_i, \gamma'_i >) = \mathcal{MQ}$ be the sequence of queries marked for interchanged and replacement responses that were generated during the execution of $\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R', KS, \mathcal{IQ})$. To construct $\mathbf{O}'$ we begin with $\mathbf{O}' \overset{defn}{=} \mathbf{O}$. Next, for each $j$ from 1 to $i$ we modify $\mathbf{O}'$ by setting $\mathbf{O}'(\gamma_j) \leftarrow c'_j$ and $\mathbf{O}'(\gamma'_j) \leftarrow c_j$ in sequence. By the construction of $\mathbf{O}'$ it's the case that $\mathbf{E}^{\mathbf{O}'}(PK, M, R_1) \rightarrow C'$, and therefore, it suffices to show that with high probability $\mathbf{G}^{\mathbf{O}'}(S^*) = (PK, SK)$. It is sufficient to show that with high probability $\mathbf{O}$ and $\mathbf{O}'$ are consistent on the set that contains all of the queries made during the execution of $\mathbf{G}^{\mathbf{O}}(S^*)$ during $Env(n)$. Call this set $\mathcal{G}$. We show that for every $(c, \gamma, c', \gamma')$ in the sequence $\mathcal{MQ}$ that with high probability neither the query $\gamma$ that is marked for interchange, nor the replacement query $\gamma'$ is in $\mathcal{G}$, and since none of the queries that are modified between $\mathbf{O}$ and $\mathbf{O}'$ are in $\mathcal{G}$, it must be the case that $\mathbf{G}^{\mathbf{O}'}(S^*) = (PK, SK)$.

Since $\overline{\mathbf{BAD}_2}$ holds, any query that is made with probability at least $1/n^{\alpha_1 - 2}$ is contained in $\mathcal{E} \subseteq \mathcal{IQ}$. Therefore, for any fixed query $\phi \in \mathcal{G}$, the probability that the execution of $\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R', KS, \mathcal{IQ})$ will *mark $\phi$ for interchange* is upper-bounded by $n^q/n^{\alpha_1 - 2}$, as queries in $\mathcal{IQ}$ will not be interchanged and $\overline{\mathbf{E}}$ will perform at most $n^q$ queries. There are at most $n^q$ queries in $\mathcal{G}$, so the probability that there is a marked query in $\mathcal{G}$ is upper-bounded by $n^{2q}/n^{\alpha_1 - 2}$.

There are at most $n^q$ *replacement* queries made during the execution $\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R', KS, \mathcal{IQ})$, and the probability that a *replacement* query is in $\mathcal{G}$ is less than $|\mathcal{G}|/n^{\alpha_0}$, as the query being replaced is not in $SQ$, as $SQ \subseteq \mathcal{IQ}$ and is therefore no larger than $\alpha_0 \cdot \log n$. As $\overline{\mathbf{BAD}_1}$ holds, there are at most $n^q$ *replacement* queries made, and therefore the probability that a replacement query is made that is in $\mathcal{G}$ is upper-bounded by $n^{2q}/n^{\alpha_0}$ □

[9]We remind the reader that these restrictions are defined in Assumption 20

### K.1.1 A Generalized Version of the $\overline{\mathbf{E}}$ Algorithm

Now that we've shown that $\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, R', KS, \mathcal{IQ})$, for random $M \in \{0,1\}$ and $R' \in \{0,1\}^{n^{\rho_2}+n^{q s}}$, is likely to produce a perturbed ciphertext $C$ for which $\mathbf{D}^{\mathbf{O}}(SK, C) = M$, we consider a variant of the algorithm, $\widehat{\mathbf{D}}$, that will allow us to specify how to perturb the ciphertexts produced by $\overline{\mathbf{E}}$ in a precise manner. This specificity will facilitate the exposition of $\textsc{Exp}_2$.

Given a set $KS$, where $|KS| = m$, for each $i \leq m$ let $pk_i = \mathit{Index}(i, KS)$[10]. Let:

$$\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, KS, \mathcal{IQ}, c_1^{pk_1,0}, ..., c_{n^q}^{pk_1,0}, c_1^{pk_1,1}, ..., c_{n^q}^{pk_1,1}, \cdots, c_1^{pk_m,0}, ..., c_{n^q}^{pk_m,0}, c_1^{pk_m,1}, ..., c_{n^q}^{pk_m,1})$$

be a modified version of the algorithm $\overline{\mathbf{E}}$: on the $i$th query $(\mathbf{e}, pk_j, b, r)$ where $(\mathbf{e}, pk_j, b, r) \notin \mathcal{IQ}$ the algorithm will respond with the value $c_i^{pk_j,b}$ specified in the input. This is formally defined below.

```
$\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, KS, \mathcal{IQ}, c_1^{pk_1,0}, ..., c_{n^q}^{pk_1,0}, c_1^{pk_1,1}, ..., c_{n^q}^{pk_1,1}, \cdots, c_1^{pk_m,0}, ..., c_{n^q}^{pk_m,0}, c_1^{pk_m,1}, ..., c_{n^q}^{pk_m,1})$
$\forall\, pk' \in KS,\ b' \in \{0,1\}$ 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 $(\mathbf{e}, pk, b, r) \in \mathcal{IQ}$ 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
```

As was stated previously, it is easy to use $\widehat{\mathbf{E}}$ to simulate $\overline{\mathbf{E}}$, and this is formalized in the following observation.

**Observation 39.** *For every $n$, $\mathbf{Env}_n = (\mathcal{O}, PK, SK, S^*) \leftarrow Env(n)$ and $E_1 = (\mathbf{Env}_n, KS, \mathcal{IQ}, \mathcal{E}, SQ) \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$, $KS = \{pk_1, \ldots, pk_m\}$, $\mathcal{IQ}$, $M \in \{0,1\}$ and $R_1 \in \{0,1\}^{n_2^{\rho}}$ the following experiments result in the same distribution of outputs:*

$\overline{\mathbf{E}}^{\mathbf{O}}(PK, M, (R_1, R_2), KS, \mathcal{IQ})$ *for randomly chosen* $R_2 \in \{0,1\}^{n^q s}$

$\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R_1, KS, \mathcal{IQ}, c_1^{pk_1,0}, ..., c_{n^q}^{pk_m,1})$ *where* $c_i^{pk,b} \leftarrow \mathbf{e}(pk, b, r_{(pk,b,i)})$ *for each* $pk \in KS$, $b \in \{0,1\}$, $1 \leq i \leq n^q$ *and for randomly chosen* $r_{(pk,b,i)} \in_R \{0,1\}^{|pk|/3}$.

The reason for introducing $\widehat{\mathbf{E}}$ is to permit it to simulate executions of $\overline{\mathbf{E}}$ in addition to performing experiments in which each input $c_i^{pk,b}$ to $\widehat{\mathbf{E}}$ is not necessarily assigned a value $\mathbf{e}(pk, b, r_{(pk,b,i)})$ for a randomly chosen $r_{(pk,b,i)}$. Specifically, we will perform a number of hybridization experiments, where we vary the different inputs $c_*^{*,*}$ to $\widehat{\mathbf{E}}$, so that we can see how the changes effect the decryption oracle's ability to decrypt the perturbed ciphertexts it produces. To facilitate the discussion of these hybrid experiments, we introduce a function $\zeta$ that is used to index the different values $c_i^{pk,b}$ that are input into $\widehat{\mathbf{E}}$. We also define some terms that will help to describe these hybridization experiments.

**Definition 40.** *Let $\zeta : \mathbb{Z}^+ \times \{0,1\} \times [n^q] \rightarrow \mathbb{Z}^+$, be defined so that $\zeta(a, b, c) = (2(a-1) + b) \cdot n^q + c$. Note that $\zeta$ is one-to-one and onto and therefore its inverse is well defined.*

The definition of $\zeta$ permits us to discuss executions of $\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R_1, KS, \mathcal{IQ}, c_1, \ldots, c_{\zeta(|KS|,1,n^q)})$ instead of $\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R_1, KS, \mathcal{IQ}, c_1^{pk_1,0}, ..., c_{n^q}^{pk_m,1})$, which will make it easier to discuss certain hybridization experiments. To facilitate discussion about the inputs to $\widehat{\mathbf{E}}$ we introduce the following terms.

[10]We remind the reader that the definition of *Index* is given in Notn. 8

**Definition 41.** *Given an execution of* $\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R_1, KS, \mathcal{IQ}, c_1, \ldots, c_{\zeta(|KS|,1,n^q)})$ *we will refer to* $c_1$ *through* $c_{\zeta(|KS|,1,n^q)}$ *as the* **ciphertext inputs of** $\widehat{\mathbf{E}}$*, and* $c_i$ *as the* $i$*th ciphertext input of* $\widehat{\mathbf{E}}$*. Similarly, if* $i = \zeta(j, b, m)$ *and the* $i$*th ciphertext input is of the form* $c_i = \mathbf{e}(pk_j, b, *)$ *then we say the* $i$*th ciphertext input is an encryption of* **the correct bit***; if* $c_i = \mathbf{e}(pk_j, 1-b, *)$ *then we say the* $i$*th ciphertext input is an encryption of* **the wrong bit***; and finally if* $c_i = \mathbf{e}(pk_j, b', *)$ *for a randomly chosen* $b'$*, then we say the* $i$*th ciphertext input is an encryption of* **a random bit***.*

This definition makes it easier to discuss hybrid experiments involving $\widehat{\mathbf{E}}$. Another term that is useful for exposition with regard to the algorithms $\overline{\mathbf{E}}$ and $\widehat{\mathbf{E}}$ is the notion of proper decryptions.

**Definition 42.** *For any* $n$*, any* $Env_n \leftarrow Env(n)$*,* $M \in \{0,1\}$*, and* $C \leftarrow \overline{\mathbf{E}}^{\mathbf{O}}(PK, M, *, *, *)$ *we say that* $C$ **decrypts properly** *if* $\mathbf{D}^{\mathbf{O}}(SK, C) = M$*. Similarly, we say* $C$ **decrypts properly** *if* $C \leftarrow \widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, *, ..., *)$ *and* $\mathbf{D}^{\mathbf{O}}(SK, C) = M$*.*

## K.2 Finding the Useful Secret-keys That are Embedded into *SK*

We will now give a high-level description of the second part of $A_1$'s experiment (we remind the reader that $A_1$ is the first part of the CCA#1 adversary). The description is both intuitive and informal; the intuition will be properly formalized later. The goal of $\textsc{Exp}_2$, the second part of the experiment, is to retrieve a large set of secret-keys for the PKEP defined by $\mathbf{O}$ that are embedded into the secret-key $SK$ defined by the adversary's environment. As the adversary does not have access to $SK$, the experiment must be designed to learn what it can about these embedded secret-keys through access to the adversary's decryption oracle.

The second part of the experiment retrieves many of the secret-keys (of the PKEP $\mathbf{O}$) whose corresponding public-keys are contained in the set $KS$ that was retrieved in $\textsc{Exp}_1$. These public-keys are the ones that are used frequently during executions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for random $M$ and $R$, and so it stands to reason that knowledge of the corresponding secret-keys may be necessary to decrypt the ciphertexts generated by the random executions of $\mathbf{E}$. The adversary retrieves these keys by using the algorithm $\widehat{\mathbf{E}}$, the oracle $\mathbf{w}$ and its ability to compute $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ through queries to its decryption oracle

The second part of the adversary's experiment begins by defining the set $BKS \overset{defn}{=} KS$ as the set of bad public-keys retrieved in $\textsc{Exp}_1$ whose *corresponding* secret-keys are not known ($BKS$ is short for Bad Key Set). As $\textsc{Exp}_2$ progresses and such secret-keys are found, the contents of the set $BKS$ will be updated. The experiment $\textsc{Exp}_2$ will perform a series of $w = \zeta(|BKS|, 1, n^q)$ sub-experiments. These sub-experiments set-up a traditional cryptographic hybridization experiment, where the hybridization is occurring over the ciphertext inputs to $\widehat{\mathbf{E}}$. At the beginning of the hybridization, in the first sub-experiment, all of the ciphertext inputs to $\widehat{\mathbf{E}}$ come from the same distribution as they would during a random execution of $\overline{\mathbf{E}}$. In other words, each ciphertext input to $\widehat{\mathbf{E}}$ corresponds to a random encryption of a *correct bit* (Defn. 41); as the experiment progresses through the hybridization, each sub-experiment replaces more and more of the ciphertext inputs to $\widehat{\mathbf{E}}$ with encryptions of *randomly chosen bits* (Defn. 41). At the end of the hybridization, in the last sub-experiment, each ciphertext input to $\widehat{\mathbf{E}}$ is an encryption of a randomly chosen bit. In each sub-experiment, we will be interested in the probability that the constructed perturbed ciphertexts decrypt properly. By Lemma 38 and Observation 39 , the ciphertexts produced by $\widehat{\mathbf{E}}$ in the first hybridization sub-experiment will *decrypt properly* with high probability, as the decryption algorithm essentially cannot tell the difference between ciphertexts produced by $\mathbf{E}$ and $\overline{\mathbf{E}}$, and $\widehat{\mathbf{E}}$'s output is equivalent to $\overline{\mathbf{E}}$'s in this sub-experiment. In contrast, we expect that the ciphertexts produced at the end of the hybridization, in the last sub-experiment, will not *decrypt properly*, as the ciphertext inputs to $\widehat{\mathbf{E}}$ in this sub-experiment are not in any way correlated to the *correct bits*. As is standard with hybridization experiments, we will make use of the first large gap between successive sub-experiments of the probability that perturbed ciphertexts decrypt properly. We will perform $w = \zeta(|BKS|, 1, n^q)$ such sub-experiments: one for each ciphertext input to $\widehat{\mathbf{E}}$.

Let us describe the sub-experiments in more detail. The $i$th sub-experiment estimates the probability, $p_i$, that $\mathbf{D}^{\mathbf{O}}(SK, C) = M$ when $C$ is generated by an execution

$$C = \widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{IQ}, c_1, ..., c_w),$$

where $M$ and $R$ are chosen randomly; $\mathcal{IQ}$ is the set constructed during $\textsc{Exp}_1$; $c_1, ..., c_i$ correspond to random encryptions under $\mathbf{e}$ of random bits; and $c_{i+1}, ..., c_w$ correspond to ciphertext encryptions of *correct bits* [11]. Once the different values $p_1, ..., p_w$ have been calculated, the experiment finds the smallest value $\ell$ for which $|p_\ell - p_{\ell-1}|$ is large. This value $\ell$ is interesting and useful, as there is a big difference between the probability that the ciphertexts generated in sub-experiment $\ell - 1$ *decrypt properly* as opposed to those generated in sub-experiment $\ell$.

The experiment makes use of the large gap in order to help recover the secret-key $sk^*_\ell = \mathbf{g}^{-1}(pk_\ell)$. In order to simplify the discussion of how $sk^*_\ell$ is retrieved, we will assume that $|sk^*_\ell| = n$, and thus $|pk_\ell| = 3n$; and further, we let the bit-wise representation of $sk^*_\ell$ be $(sk^*_{\ell,1}, ..., sk^*_{\ell,n})$. The experiment will retrieve each bit of $sk^*_\ell$ individually. We will describe how the experiment retrieves $sk^*_{\ell,1}$, and note that the experiment uses a similar process to retrieve the remaining bits of $sk^*_\ell$.

In order to retrieve $sk^*_{\ell,1}$, the experiment generates an estimate $\pi$ of the probability that the ciphertexts generated in the following experiment *decrypt properly.*

The experiment generates

$$C = \widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{IQ}, c_1, ..., c_w),$$

where $M$ and $R$ are chosen randomly, $\mathcal{IQ}$ is the set constructed during $\textsc{Exp}_1$. The ciphertext inputs $c_1, ..., c_{i-1}$ are encryptions of random bits. The input ciphertexts $c_{i+1}, ..., c_w$ are encryptions of *correct bits.* Finally, the $i$th ciphertext input is an encryption, $\mathbf{e}(pk_\ell, sk_{\ell,1}, r)$, that is retrieved by using the response from an oracle query $\mathbf{w}(pk_\ell, \cdot)$.

The key observation about the estimate $\pi$ is the following: if $sk^*_{\ell,1} = b_\ell$ then the probability that the ciphertexts generated in the described experiment will *decrypt properly* is exactly $p_{\ell-1}$, as the experiment generates the same distribution on ciphertexts as that generated in sub-experiment $\ell - 1$. In contrast, if $sk^*_{\ell,1} = 1 - b_\ell$ then the probability that the ciphertexts generated in the described experiment will *decrypt properly* is closer to $p_\ell$. Therefore, because of the large gap between $p_\ell$ and $p_{\ell-1}$ the experiment is able to confidently predict the value of $sk^*_{\ell,1}$.

Once we have retrieved $sk^*_\ell$, we remove $pk_\ell$ from $BKS$ and we begin the whole process of generating hybrid sub-experiments to calculate the different values $p_i$ over again, but note that the set $BKS$ is now smaller, and so this process cannot continue ad infinitum. Therefore, this process continues until either $BKS$ is the empty set, and thus we have retrieved all of the secret-keys that we were interested in; or until for a fixed set $BKS$ there does not exit an $\ell$ where the gap between $p_\ell$ and $p_{\ell-1}$ is large. This latter case intuitively corresponds to the situation where it is not essential to retrieve the remaining secret-keys corresponding to the remaining public-keys in $BKS$ in order to properly decrypt the ciphertext generated by $\mathbf{E}$. For an example of this latter case, consider an public-key $pk = \mathbf{g}(sk)$ that is embedded into $PK$, and for which an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R) = C$ is likely to embed a value $c = \mathbf{e}(pk, b, r)$ into $C$; and finally during $\mathbf{D}^{\mathbf{O}}(SK, C)$ the query $\mathbf{d}(sk, c)$ is made, but the returned value is ignored, clearly in such a situation it is not necessary, nor even possible using the techniques presented here, to retrieve $sk$.

## K.3 The Formal Description of the Second Part of the Experiment

We now give the formal definition of the second part of the experiment. As described in Section K.2, there are many situations in which the experiment needs to perform sampling to approximate the probability that certain ciphertexts generated by $\widehat{\mathbf{E}}$ *decrypt properly.* Further, in all of these samplings we perform many different executions $\widehat{\mathbf{E}}$ with inputs that come from nearly identical distributions. Therefore, in order to allow us to succinctly describe the sampling in this part of the experiment we introduce two helper routines: $\widehat{\mathbf{E}}$-*Err* and *ApproxErrorRate*.

### K.3.1 The subroutine $\widehat{\mathbf{E}}$-*Err*

The subroutine $\widehat{\mathbf{E}}$-*Err* both generates the different encryptions needed in the different hybridization sub-experiments and determines if the generated ciphertexts *decrypt properly.* The protocol takes as inputs the set of bad public-keys, $BKS$; a set of immutable queries, $\mathcal{IQ}$; a public/secret-key pair $(PK, SK)$; an index value $t$ between 1 and $w = \zeta(|BKS|, 1, n^q)$; and a value $c^*$ representing the $t^{\text{th}}$ ciphertext input to be used

[11]More specifically, for each $t \leq w$ let $(j_t, b_t, *) \leftarrow \zeta^{-1}(t)$ and $pk_t \leftarrow Index(j_t, BKS)$; then for each $t \leq i$ we have $c_t \leftarrow \mathbf{e}(pk_t, b'_t, r_t)$ for randomly chosen $b'_t$ and $r_t$; and he for each $t > i$, we have $c_t \leftarrow \mathbf{e}(pk_t, b_t, r_t)$ for randomly chosen $r_t$.

as an input for $\widehat{\mathbf{E}}$. The subroutine returns 1 or 0 depending on whether or not $\mathbf{D^O}(SK, C) = M$ for a ciphertext $C$ generated as follows: the subroutine computes $C = \widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{IQ}, c_1, ..., c_w)$, where the ciphertext input $c_k$ correspond to a random encryption under $\mathbf{e}$ of random bits for $k < t$; $c_t$ is set to the input $c^*$; and, for $k \geq t{+}1$, the input $c_k$ is a random encryption under $\mathbf{e}$ of the bit $b_k$, where $(*, b_k, *) = \zeta^{-1}(k)$. The algorithm's pseudo-code is given below:

```
$\widehat{\mathbf{E}}$-$Err(t, BKS, \mathcal{IQ}, (PK, SK), c^*)$
(1)     Choose $M \in_R \{0,1\}$
(2)     Choose $R \in_R \{0,1\}^{n^{\rho_2}}$
(3)     Let $w \leftarrow \zeta(|BKS|, 1, n^q)$
(4)     for $k = 1$ to $w$
(5)         Let $(\ell_k, b_k, *) \leftarrow \zeta^{-1}(k)$ and let $pk_k \leftarrow Index(\ell_k, BKS)$.
(6)         IF $k < t$ THEN $\bar{c}_k \leftarrow \mathbf{e}(pk_k, \bar{b}, r)$ WHERE $r \in_R \{0,1\}^{|pk_k|/3}$ AND $\bar{b} \in_R \{0,1\}$
(7)         ELSE IF $k = t$ THEN $\bar{c}_k \leftarrow c^*$
(8)         ELSE $c_k \leftarrow \mathbf{e}(pk_k, b_k, r)$ WHERE $r \in_R \{0,1\}^{|pk_k|/3}$
(9)     $C \leftarrow \widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{IQ}, \bar{c}_1, ..., \bar{c}_t, c_{t+1}, ..., c_w)$
(10)    $\overline{M} \leftarrow \mathbf{D^O}(SK, C)$
(11)    IF $M \neq \overline{M}$ output 1 ELSE output 0
```

### K.3.2 The Subroutine *ApproxErrorRate*

The subroutine *ApproxErrorRate* approximates the probability that certain distributions of ciphertexts **do not** *decrypt properly.* The algorithm *ApproxErrorRate* does this by randomly executing $\widehat{\mathbf{E}}$-*Err* a polynomial number of times and then averaging the returned results.

We wish to use *ApproxErrorRate* to approximate the error rate for the different hybridization sub-experiments, and therefore we need to ensure that we can specify those different distributions by appropriately setting the inputs to *ApproxErrorRate*. In order to be able to execute $\widehat{\mathbf{E}}$-*Err*, *ApproxErrorRate* takes as input a set of immutable queries $\mathcal{IQ}$, a set of bad public-keys $BKS$, an index value $t$ and a public-/secret-key pair $(PK, SK)$ and a set of ciphertext inputs $\mathcal{D}$. Given these values, *ApproxErrorRate* performs $n^{2\alpha_4}$ independent and random executions of $\widehat{\mathbf{E}}$-$Err(t, BKS, \mathcal{IQ}, (PK, SK), c^*)$. Observe that all of the inputs to $\widehat{\mathbf{E}}$-*Err* have been specified except for the inputs $c^*$, and these values are specified by the set $\mathcal{D}$ in two possible ways: if $\mathcal{D} = \emptyset$ then $c^*$ is chosen to be a random encryption of a random bit; alternatively, if $c^*$ is not empty, then the value $c^*$ is specified by $\mathcal{D}$. The experiment will execute *ApproxErrorRate* with $\mathcal{D} = \emptyset$ when it is estimating probabilities in the hybrid sub-experiments. In contrast, the experiment will execute *ApproxErrorRate* with $\mathcal{D}$ containing results returned from queries to $\mathbf{w}$, when it is attempting to retrieve the secret-keys embedded in $SK$.

The formal specification of the algorithm is given below [12].

[12] In the definition of the subroutine *ApproxErrorRate* and the definition of $\text{Exp}_2$ we use the *Index* function. We remind the reader that this is defined in Notn. 8.

```
$ApproxErrorRate(t, \mathcal{D}, (PK, SK), \mathcal{IQ}, BKS)$
(1)   $error \leftarrow 0$
(2)   for $z = 1$ to $n^{2\alpha_4}$
(3)       IF $\mathcal{D} \neq \emptyset$ let $c^* \leftarrow Index(z, \mathcal{D})$
(4)       ELSE
(5)           Let $(\ell_t, *, *) \leftarrow \zeta^{-1}(t)$ and let $pk_t \leftarrow Index(\ell_t, BKS)$.
(6)           $c^* \leftarrow \mathbf{e}(pk_t, b, r)$ WHERE $r \in_R \{0,1\}^{|pk_k|/3}$ AND $\bar{b} \in_R \{0,1\}$
(7)       $error \leftarrow error + \widehat{\mathbf{E}}\text{-}Err(t, BKS, \mathcal{IQ}, (PK, SK), c^*)$.
(8)   OUTPUT $error/n^{\alpha_4}$.
```

### K.3.3 $\mathbf{Exp}_2$: The Second Part of the Experiment

We finally give the formal description of the second part of the experiment. Because $\textsc{Exp}_2$ is properly viewed as a continuation of $\textsc{Exp}_1$, it takes as input a value $E_n^1 = (\mathbf{Env}_n, KS, \mathcal{IQ}, \mathcal{E}, SQ) \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ that includes all of the values and sets calculated in $\textsc{Exp}_1(\mathbf{Env}_n)$ and all of the elements contained in $\mathbf{Env}_n \leftarrow Env(n)$.

```
$\textsc{Exp}_2(E_n^1 = (\mathbf{Env}_n = (\mathcal{O}, PK, SK, S^*), KS, \mathcal{IQ}, \mathcal{E}, SQ))$
(1)   Let $BKS_0 \leftarrow KS$
(2)   Let $GKS_0 \leftarrow \emptyset$
(3)   for $\tau = 1$ to $|KS|$
(4)       $w \leftarrow \zeta(|BKS|, 1, n^q)$
(5)       for $t = 0$ to $w$
(6)           Let $p_t = ApproxErrorRate(t, \emptyset, (PK, SK), \mathcal{IQ}, BKS_{\tau-1})$
(7)       $\Delta \leftarrow \{j \mid 1 \leq j \leq w$ and $|p_j - p_{j-1}| > \frac{1}{n^{\alpha_6}}\}$
(8)       If $\Delta = \emptyset$ then
(9)           $\tau^* \leftarrow \tau$
(10)          $BKS^* \leftarrow BKS_{\tau-1}$ and $GKS^* \leftarrow GKS_{\tau-1}$
(11)          Output $(\tau^*, BKS^*, GKS^*, E_n^1)$
(12)          FINISH EXPERIMENT
(13)      $\ell \leftarrow \min \Delta$
(14)      $(i, b, *) \leftarrow \zeta^{-1}(\ell)$
(15)      Let $pk \leftarrow Index(i, BKS_{\tau-1})$ and say it is CHOSEN by $\ell$
(16)      $\psi \leftarrow |pk|/3$
(17)      For each $k$ $(1 \leq k \leq \psi)$ set $\mathcal{D}_k \leftarrow \emptyset$
(18)      for $z = 1$ to $n^{2\alpha_4}$
(19)          $(d_1, ..., d_\psi) \leftarrow \mathbf{w}(pk, z)$ (we consider $z$ a binary string)
(20)          For each $k$ $(1 \leq k \leq \psi)$ set $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{d_k\}$
(21)      for $k = 1$ to $\psi$
(22)          $\pi_k \leftarrow ApproxErrorRate(\ell, \mathcal{D}_k, (PK, SK), \mathcal{IQ}, BKS_{\tau-1})$
(23)          If $|\pi_k - p_{\ell-1}| \geq 1/n^{\alpha_5}$ then $\overline{sk}_k \leftarrow 1 - b$ otherwise $\overline{sk}_k \leftarrow b$
(24)      Let $\overline{sk} = (\overline{sk}_1, ..., \overline{sk}_\psi)$
(25)      If $\mathbf{g}(\overline{sk}) \neq pk$ HALT
(26)      For each $k \leq \psi$ and each $d \in \mathcal{D}_k$ query $\mathbf{d}(\overline{sk}, d)$
(27)      $GKS_\tau \leftarrow GKS_{\tau-1} \cup \{(pk, \overline{sk})\}$
(28)      $BKS_\tau \leftarrow BKS_{\tau-1} \setminus \{pk\}$
```

We briefly map how the intuitive description $\textsc{Exp}_2$ that was presented in Section K.2 corresponds to the formal description of $\text{Exp}_2$ presented above. The experiment is iteratively attempting to retrieve secret-keys that correspond to the public-keys in $KS$. In each iteration of the **for loop** between lines 3 and 28, the algorithm attempts to retrieve one such secret-key. We consider one such iteration.

In lines 5 and 6 the experiment iteratively performs the different hybrid sub-experiments that find estimates, $p_t$, of the probability that ciphertexts decrypt properly when they are generated by executions of the form:

$$\widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS_{\tau-1}, \mathcal{IQ}, c_1, ..., c_w),$$

where $c_1, .., c_t$ correspond to ciphertext inputs of encryptions of random bits, and $c_{t+1}, ..., c_w$ correspond to ciphertext inputs of encryptions of correct bits. In lines 7 and 12 the experiment looks for the smallest value $t$ where there is a big gap between $p_t$ and $p_{t+1}$, and the experiment is terminated if there is no such gap. In lines 18 to 20 the experiment queries $\mathbf{w}$ in order to get ciphertexts that represent an encrypted version of the secret-key $sk = \mathbf{g}^{-1}(pk)$, where $pk = Index(t, BKS_{\tau-1})$.

In lines 22 to 25 the experiment attempts to decrypt the encrypted versions of $sk = \mathbf{g}^{-1}(pk)$ that were retrieved from $\mathbf{w}$. This is done by taking the ciphertexts returned by $\mathbf{w}$ and separating them into sets that represent different encryptions of each bit of $sk$. The experiment then separates the different encryptions of each bit of $sk$ into different sets on line 20. Once these sets have been established, the experiment uses them as inputs to *ApproxErrorRate*. For every such set, the experiment calls *ApproxErrorRate* and estimates the probability that a certain distribution of ciphertexts generated by $\widehat{\mathbf{E}}$ will properly decrypt. This probability will either be close to $p_t$ or $p_{t+1}$, and this will indicate if the bit of $sk$ is likely a 1 or 0. The experiment is halted on line 25 if $sk$ is retrieved incorrectly, as this will cause problems with future analysis. However, we show that such a case is unlikely.

We want to show that with high probability the second part of the experiment will finish on line 12 and not on line 25. If the experiment finishes on line 12, it implies that all of the secret-keys $sk = \mathbf{g}^{-1}(pk)$ that are in some sense necessary to properly perform decryptions with the algorithm $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ have been retrieved. Showing this will be the main focus of later sections of the chapter. For now, we will settle on the more immediate goal of showing that with high probability the experiment will halt on line 12 of $\text{Exp}_2$ and thus retrieve all of the secret-keys of interest. However before we do this, we will continue our running example, in an attempt to ground the reader's understanding of $\text{Exp}_2$.

## K.4 The Running Example for $\text{Exp}_2$

**Running Example 3.** *We remind the reader that in Running Example 1 it was decided that $KS = \{pk_1, ..., pk_6\}$. It is the goal of $\text{Exp}_2$ to retrieve as many values in the set $\{sk_1 = \mathbf{g}^{-1}(pk_1), ..., sk_6 = \mathbf{g}^{-1}(pk_6)\}$ as is possible. In order to do so, $\text{Exp}_2$ will begin by performing the hybridization sub-experiments, described in lines 5 through 6, in order to calculate the estimates, $p_t$, of the probability that ciphertexts decrypt properly when they are generated by executions of the form:*

$$\widehat{\mathfrak{E}}^{\mathbf{O}}(PK, M, R, BKS_{\tau-1}, \mathcal{IQ}, c_1, ..., c_w),$$

*where $c_1, .., c_t$ correspond to ciphertext inputs of encryptions of random bits, and $c_{t+1}, ..., c_w$ correspond to ciphertext inputs of encryptions of correct bits.*

*For the sake of exposition, we will assume that $pk_i \prec pk_{i+1}$. This assumption implies that in the execution of $\text{Exp}_2$, it is likely that the first large gap that appears in the respective estimates is between $p_{4q}$ and $p_{4q+1}$. The reason for this is that the replacement of ciphertext inputs corresponding to* correct bits *(Defn. 41) with ciphertext inputs corresponding to* random bits*, will have no effect in the first $4q$ hybridization sub-experiments; to see why, we recall the definition of $\mathfrak{D}$, presenting it below, and then observe the difference in its behaviour between the the two sub-experiments.*

- $\mathfrak{D}^{\mathbf{O}}(SK, C)$*: Let $C = (b, k_1', k_2', c_1, c_2, \ldots, c_5)$ where $b \in \{0,1\}$, $k_1', k_2' \in \{0,1\}^n$ and each $c_i \in \{0,1\}^n$. Let $SK = (sk_0, ..., sk_5, sk_6, k_1, k_2)$ where each $sk_i \in \{0,1\}^n$.*
  *If $k_1' \neq k_1$ or $k_2' \neq k_2$ output $\perp$.*
  *Otherwise, if $\mathbf{d}(sk_6, k_1') \neq 0$ or $\mathbf{d}(sk_6, k_2') \neq 0$ 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 $Majority(M_1, ..., M_5)$,*

*We observe the difference in behaviour between the $4q$ and $4q+1$st sub-experiment*

- *In the first $4q$ hybridization sub-experiments, it will be the case that a ciphertext $C = (b, k_1', k_2', c_1, c_2, \dots, c_5)$ generated by a call to $\widehat{\mathfrak{E}}^{\mathbf{O}}(PK, M, R, BKS_0, \mathcal{IQ}, c_1, ..., c_w)$ in ApproxErrorRate will have the property that neither $\mathbf{d}(sk_1, c_1) = M_1$ nor $\mathbf{d}(sk_2, c_2) = M_2$ will necessarily equal $M$. However, due to the fact that $\mathfrak{D}$ outputs Majority$(M_1, ..., M_5)$ and $M_3$ through $M_5$ will necessarily equal $m$, this will will not effect the probability of a correct decryption in an execution of $\mathfrak{D}^{\mathbf{O}}(SK, C)$, and therefore $p_{4q} \approx p_0$*
- *In the $4q+1$st hybridization sub-experiment, approximately half the ciphertexts generated by a call to $\widehat{\mathfrak{E}}^{\mathbf{O}}(PK, M, R, BKS_0, \mathcal{IQ}, c_1, ..., c_w)$ in ApproxErrorRate will have $M = 0$. Consider such a case where*

$$C^* = (b, k_1', k_2', c_1^*, c_2^*, \dots, c_5^*) = \widehat{\mathfrak{E}}^{\mathbf{O}}(PK, 0, R, BKS_0, \mathcal{IQ}, c_1, ..., c_w)$$

  *in ApproxErrorRate. In this case $c_1^* = c_1, c_2^* = c_{2q+1}$ and $c_3^* = c_{4q+1}$, and therefore we have the property that $M_1 = \mathbf{d}(sk_1, c_1^*) = M_2 = \mathbf{d}(sk_1, c_2^*) = M_3 = \mathbf{d}(sk_3, c_3^*) = 1$ with a constant probability of $1/8$. Now, due to the fact that $\mathfrak{D}$ outputs Majority$(M_1, ..., M_5)$, this will will imply that with constant probability $\mathfrak{D}^{\mathbf{O}}(SK, C^*)$ would output $1$ as opposed to $0$, and therefore there will be an improper decryption. This implies that there should be a large gap of approximately $1/16$ between the estimates $p_{4q}$ and $p_{4q+1}$.*

*Based on the above analysis, we expect $\ell$ to be assigned the value $4q+1$ on line 13 of* Exp$_2$*, and thus $pk_3$ is* chosen by *$\ell$ on line 15. This implies that* Exp$_2$ *will attempt to retrieve $sk_3$ in the first iteration of the* **for loop** *that falls between lines 3 and 28.*

*In order to retrieve $sk_3$,* Exp$_2$ *queries $\mathbf{w}(pk_3, *)$ a number of times, in order to retrieve a number of different bit-wise encryptions of $sk_3$. This is done on line 19 of* Exp$_2$*. Next,* Exp$_2$ *separates these bit-wise encryptions into a number of sets $\mathcal{D}_1, \dots, \mathcal{D}_n$, where $\mathcal{D}_i$ contains all of the bit encryptions of the $i$th bit of $sk_3$. This is done in lines 18 through 20 of* Exp$_2$*.*

*Given each set $\mathcal{D}_i$,* Exp$_2$ *will attempt to retrieve the $i$th bit of $sk_3$. The same process is used to retrieve each bit, so we only consider the process used to retrieve the first bit. We consider two cases:*

**The first bit of $sk_3$ is 0** *, and this implies that all of the ciphertext encryptions in $\mathcal{D}_0$ represent random encryptions of the bit $0$. Therefore, the execution of ApproxErrorRate$(4q+1, \mathcal{D}_0, (PK, SK), \mathcal{IQ}, BKS_0))$ will calculate an estimate, $\pi_0$, of the probability of a decryption error for ciphertexts generated randomly by an execution of*

$$\widehat{\mathfrak{E}}^{\mathbf{O}}(PK, M, R, BKS_{\tau-1}, \mathcal{IQ}, c_1, ..., c_w),$$

*where $c_1, .., c_{4q}$ correspond to ciphertext inputs of encryptions of random bits and $c_{4q+1}, ..., c_w$ correspond to ciphertext inputs of correct bits. Thus $\pi_0$ is an estimate of exactly the same value as is $p_{4q}$.Therefore, if $\pi_0 \approx p_{4q}$ then* Exp$_2$ *assumes the first bit of sk is a $0$.*

**The first bit of $sk_3$ is 1** *and this implies that all of the ciphertext encryptions in $\mathcal{D}_0$ represent random encryptions of the bit $1$. Therefore, the execution of ApproxErrorRate$(4q+1, \mathcal{D}_0, (PK, SK), \mathcal{IQ}, BKS_0))$ will calculate an estimate, $\pi_0$, of the probability of a decryption error for ciphertexts generated randomly by an execution of*

$$\widehat{\mathfrak{E}}^{\mathbf{O}}(PK, M, R, BKS_{\tau-1}, \mathcal{IQ}, c_1, ..., c_w),$$

*where $c_1, .., c_{4q}$ correspond to ciphertext inputs of encryptions of random bits; $c_{4q+1}$ corresponds to a ciphertext input of an encryption of an* incorrect bit*; and $c_{4q+2}, ..., c_w$ correspond to ciphertext inputs of correct bits. We stress that the ciphertext $c_{4q+1}$ will always represent an encryption of $1$, whereas if it represented a correct bit, it would be an encryption of $0$. In this case $\pi_0$ is not an estimate of $p_{4q}$ or $p_{4q+1}$. However, $\pi_0$ is not going to be close to $p_{4q}$. During ApproxErrorRate, in the half of the cases where a ciphertext is constructed as follows: $C = (b, k_1', k_2', c_1, c_2, \dots, c_5) = \widehat{\mathfrak{E}}^{\mathbf{O}}(PK, M, R, BKS_{\tau-1}, \mathcal{IQ}, c_1, ..., c_w)$ and $M = 0$, then with probability $1/4$ $C$ will not decrypt properly. It will be the case that $M_1 = \mathbf{d}(sk_1, c_1^*) = M_2 = \mathbf{d}(sk_1, c_2^*) = 1$ with probability $1/4$ and since in this case it is always the case that $M_3 = \mathbf{d}(sk_3, c_3^*) = 1$, Majority$(M_1, ..., M_5) = 1$ with probability $1/4$, and therefore the ciphertext decrypts improperly with same probability. Therefore, $|p_{4q} - \pi_0| \approx \frac{1}{2} \cdot \frac{1}{8}$ and so* Exp$_2$ *assumes that the first bit of sk is a 1.*

*The above process is repeated to retrieve each bit of $sk_3$, until the experiment has a hypothesized version of the key $\overline{sk}$. Next,* $\textsc{Exp}_2$ *verifies that it retrieved the correct secret-key, by querying $\mathbf{g}(\overline{sk})$ and ensuring it responds with $pk_3$. In the unlikely event that correct secret-key was not retrieved the experiment halts. Otherwise, the key pair $(pk_3, sk_3)$ is added to GKS (the Good Key Set), $pk_3$ is removed from BKS (the bad key set), and the entire process is repeated in an attempt to find more of the secret-keys that correspond to the public-keys remaining in BKS.*

*We point out that in our running example, we expect that* $\textsc{Exp}_2$ *will finish with $GKS = \{(pk_3, sk_3), ..., (pk_6, sk_6)\}$. We note that $sk_1$ and $sk_2$ will not be retrieved, as they are effectively unessential for decryption, given that we know $sk_3$ through $sk_6$.*

## K.5 Bounding the Probability that $\mathbf{Exp}_2$ Halts

The next definition formalizes the different probabilities that we are trying to estimate in line 6 of $\textsc{Exp}_2$ with the calculation of $p_t$ and $\pi_k$.

**Definition 43.** *Given any $n$, $\mathbf{Env}_n \leftarrow Env(n)$, $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ holds, for every $BKS \subseteq KS$, $w = \zeta(|BKS|, 1, n^q)$ and $t$ such that $1 \le t \le w$, we define the values:*

$$P_{t,BKS,E_n^1} = \Pr[\widehat{\mathbf{E}}\text{-}Err(t, BKS, \mathcal{IQ}, (PK, SK), \mathbf{e}(pk, b, r)) = 1] \text{ and}$$

$$\widehat{P}_{t,BKS,E_n^1} = \Pr[\widehat{\mathbf{E}}\text{-}Err(t, BKS, \mathcal{IQ}, (PK, SK), \mathbf{e}(pk, 1-b, r)) = 1],$$

*where $(\ell, b, *) = \zeta^{-1}(t)$ and $pk = Index(\ell, BKS)$. In both cases the probability is over the random choices made by $\widehat{\mathbf{E}}$-Err and the random choice of $r \in_R \{0,1\}^{|pk|/3}$. When the random variable $E_n^1$ is clear from context we denote $P_{t,BKS,E_n^1}$ as $P_{t,BKS}$, and, similarly, we denote $\widehat{P}_{t,BKS,E_n^1}$ as $\widehat{P}_{t,BKS}$.*

We note that if we fix any $n$, for every $\mathbf{Env}_n \leftarrow Env(n)$ and $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ holds, then for every $BKS \subseteq KS$ and every $\ell > 1$ there is a natural relation between $P_{\ell,BKS}$ and $\widehat{P}_{\ell,BKS}$:

$$P_{\ell,BKS} = 1/2(P_{\ell-1,BKS} + \widehat{P}_{\ell,BKS}). \tag{1}$$

This is observed by noting that the only difference between the definitions of $P_{\ell-1,BKS}$ and $\widehat{P}_{\ell,BKS}$ is the input $c^*$ to $\widehat{\mathbf{E}}$-*Err*, which is set to be $\mathbf{e}(pk, b, r)$ or $\mathbf{e}(pk, 1-b, r)$ in the respective definitions. However, $P_{\ell,BKS}$ can be viewed as the probability that a random encryption produced by $\widehat{\mathbf{E}}$ will properly decrypt when its first $\ell$ ciphertext inputs are encryptions of random bits, and the remaining ciphertext inputs are encryptions of "correct" bits. Similarly, $P_{\ell,BKS}$ can be viewed as the probability that a random encryption produced by $\widehat{\mathbf{E}}$ will decrypt properly when its first $\ell - 1$ ciphertext inputs are encryptions of random bits, the $\ell$th ciphertext input is an encryption of the *i*ncorrect bit, and the remaining ciphertext inputs are encryptions of correct bits. When viewed this way, it is clear that $P_{\ell,BKS}$ is simply the expectation over the outcomes of $P_{\ell-1,BKS}$ or $\widehat{P}_{\ell,BKS}$ where each event is equally likely.

We present two lemmas and a corollary that demonstrate that *ApproxErrorRate* can be used to approximate the values $P_{\ell,BKS}$ and $\widehat{P}_{\ell,BKS}$.

**Lemma 44.** *For all sufficiently large $n$, for every $\mathbf{Env}_n \leftarrow Env(n)$ and $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ holds, for every $BKS \subseteq KS$, for $w = \zeta(|BKS|, 1, n^q)$ and every $t$ such that $1 \le t \le w$:*

$$\Pr[|ApproxErrorRate(t, \emptyset, (PK, SK), \mathcal{IQ}, BKS) - P_{t,BKS}| > 1/16n^{\alpha_5}] \le 1/2^n,$$

*where the probability is over the random choices made during the execution of ApproxErrorRate.*

*Proof.* This follows directly from the Chernoff-Hoeffding bound, as long as $n^{2\alpha_4} \ge 6 \cdot 16^2 \cdot n^{2\alpha_5+1}$. ☐

**Corollary 45.** *For all sufficiently large $n$, for every $\mathbf{Env}_n \leftarrow Env(n)$ and $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ holds, for every $BKS \subseteq KS$, for $w = \zeta(|BKS|, 1, n^q)$ and every $t$ such that $1 \le t \le w$:*

$$\Pr[|ApproxErrorRate(t, \mathcal{D}, (PK, SK), \mathcal{IQ}, BKS) - P_{t-1,BKS}| > 1/16n^{\alpha_5}] \le 1/2^n,$$

*where the probability is over the random choices made during the execution of ApproxErrorRate and the random construction of the set*

$$\mathcal{D} = \{c_1, ..., c_{n^{2\alpha_4}}\}.$$

*The set $\mathcal{D}$ is randomly generated as follows: let $\zeta^{-1}(t) = (i, b, *)$ and $pk = Index(BKS, i)$, then for all $j$ ($1 \le j \le n^{2\alpha_4}$) we let $c_j \leftarrow \mathbf{e}(pk, b, r_j)$ for a random $r_j \in_R \{0,1\}^{|pk|/3}$.*

*Proof.* By the distribution on $\mathcal{D}$, the random variable $ApproxErrorRate(t, \mathcal{D}, (PK, SK), \mathcal{IQ}, BKS)$ has the same distribution as $ApproxErrorRate(t-1, \emptyset, (PK, SK), \mathcal{IQ}, BKS)$, and therefore by Lemma 44 the corollary holds. (Note: Same relations on $\alpha_4$ and $\alpha_5$ hold as in the lemma) $\square$

**Lemma 46.** *For all sufficiently large $n$, for every $\mathbf{Env}_n \leftarrow Env(n)$ and $E^1_n \leftarrow \text{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ holds, for every $BKS \subseteq KS$, for $w = \zeta(|BKS|, 1, n^q)$ and every $t$ such that $1 \le t \le w$:*

$$\Pr[|ApproxErrorRate(t, \mathcal{D}, (PK, SK), \mathcal{IQ}, BKS) - \widehat{P}_{t,BKS})| > 1/16n^{\alpha_5}] \le 1/2^n,$$

*where the probability is over the random choices made during the execution of ApproxErrorRate and the randomized set*

$$\mathcal{D} = \{c_1, ..., c_{n^{2\alpha_4}}\}.$$

*The set $\mathcal{D}$ is randomly generated as follows: let $\zeta^{-1}(t) = (i, b, *)$ and $pk = Index(BKS, i)$, then for all $j$ ($1 \le j \le n^{2\alpha_4}$) we let $c_j \leftarrow \mathbf{e}(pk, 1-b, r_j)$ for a random $r_j \in_R \{0,1\}^{|pk|/3}$.*

*Proof.* Essentially the same proof as a combination of the proofs of the Lemma 44 and Corollary 45. $\square$

Next, we have a lemma that shows that with high probability $\text{Exp}_2(E^1_n)$ doesn't halt on line 25 (page 37) during any iteration of the **for loop** (lines 3 to 28). In other words, $\text{Exp}_2$ never erroneously retrieves a secret-key that does not correspond to a public-key in $KS$.

**Lemma 47.** *For all sufficiently large $n$:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n) \\ E^1_n \leftarrow \text{Exp}_1(\mathbf{Env}_n) \\ \text{Exp}_2(E^1_n)}} [\text{Exp}_2(E^1_n) \textit{ HALTS on line 25 of } \text{Exp}_2 | \overline{\mathbf{BAD}_1}] \le \frac{5 \cdot n^{q+s}}{2^n}.$$

*We denote the event that $\text{Exp}_2(E^1_n)$ halts in such an experiment as $\mathbf{BAD}_3$.*

*Proof.* As $\overline{\mathbf{BAD}_1}$ holds, we know that $|KS| \le n^q$. Therefore, in $\text{Exp}_2(E^1_n)$ the variable $\tau$ can only take on $n^q$ different values, and therefore at most $n^q$ different secret-keys can be retrieved. Therefore, we can inductively apply the following claim $n^q$ times, and the result follows by an application of the union-bound.

**Claim 48.** *Consider an experiment that is the same as that described in Lemma 47, and that has the property that during the first $i$ iterations of the* **for loop** *(lines 3 to 28) of $\text{Exp}_2(E^1_n)$ (where $0 \le i < |KS|$) the experiment has not HALTED (line 25). In this experiment, if during the $i+1$st iteration of the* **for loop** *(where $\tau = i+1$) the value $\ell \leftarrow \min \Delta$ was selected and the public-key $pk$ was* chosen *on lines 13 and 15 respectively of $\text{Exp}_2(E^1_n)$, then the probability that $\mathbf{g}(\overline{sk}) \ne pk$ is less than $\frac{5n^s}{2^n}$. Here, $\overline{sk}$ is the value retrieved on line 24 of $\text{Exp}_2$, and the probability is over the random choices made in choosing the sub-oracle $\mathbf{w}(pk, \cdot)$ and the random choices made in $\text{Exp}_2(E^1_n)$ in the $i+1$st execution of the* **for loop**.

*Proof.* As in $\text{Exp}_2$, let $\psi = |pk|/3$ and $(i, b, *) = \zeta^{-1}(\ell)$. By the statement of the claim, $\tau = i+1$. Because $pk \in BKS_{\tau-1} \subseteq KS$, and because by $KS$'s construction for every $pk' \in KS$ $\mathbf{g}^{-1}(pk')$ is well defined, exists an $sk = (sk_1, .., sk_\psi)$ such that $sk = \mathbf{g}^{-1}(pk)$. We will show that with high probability an arbitrary $\overline{sk_\gamma} \in \{sk_1, .., sk_\psi\}$ is retrieved correctly by $\text{Exp}_2$. The union bound can then be used to argue that the value $\overline{sk}$, assembled on line 24 of $\text{Exp}_2$, is equal to $sk$.

$\textsc{Exp}_2$ retrieves $n^{2\alpha_4}$ random bit-wise encryptions of $sk_\gamma$ lines by using the sub-oracle $\mathbf{w}$ (lines 18 through 20 ) [13] . They are then stored in $\mathcal{D}_\gamma$. The encryptions generated by queries to $\mathbf{w}$ are random due to the random process that constructed $\mathbf{w}$ and the fact that no queries of the form $(\mathbf{w}, pk, *)$ have previously been made in the experiment (this can be seen by observation of $\textsc{Exp}_1$ and $\textsc{Exp}_2$). Next, the experiment calculates $\pi_\gamma = \mathit{ApproxErrorRate}(\ell, \mathcal{D}_\gamma, (PK, SK), \mathcal{IQ}, BKS_{\tau-1})$. At this point we'll consider the two possible cases $sk_\gamma = b$ and $sk_\gamma = 1 - b$.

**Case** $sk_\gamma = b$**:** by Corollary 45, with high probability the value $\mathit{ApproxErrorRate}(\ell, \mathcal{D}_\gamma, (PK, SK), \mathcal{IQ}, BKS_{\tau-1}) = \pi_\gamma$ is an accurate approximation of $P_{\ell-1,BKS_{\tau-1}}$. Namely, with probability at least $1 - 1/2^n$ we know $|\pi_\gamma - P_{\ell-1,BKS_{\tau-1}}| < 1/16n^{\alpha_5}$. Since it's also the case, by Lemma 44, that with probability at least $1 - 1/2^n$ that $|p_{\ell-1} - P_{\ell-1,BKS_{\tau-1}}| \leq 1/16n^{\alpha_5}$ we know that $|\pi_\gamma - p_{\ell-1}| \leq 1/8n^{\alpha_5} \leq 1/n^{\alpha_5}$ with probability at least $1 - 2/2^n$. and therefore, by line 23 of $\textsc{Exp}_2$, the experiment will set $\overline{sk_\gamma} \leftarrow b$.

**Case** $sk_\gamma = 1 - b$**:** $\pi_\gamma$ is an approximation to $\widehat{P}_{\ell,BKS_{\tau-1}}$ by Lemma 46. Specifically, with probability at least $1 - 1/2^n$ it's the case that $|\pi_\gamma - \widehat{P}_{\ell,BKS_{\tau-1}}| \leq 1/16n^{\alpha_5}$. Similarly, by Lemma 44, $p_{\ell-1}$ is a good approximation of $P_{\ell-1,BKS_{\tau-1}}$. Combining these approximation with Eqn. 1 ($P_{\ell,BKS} = 1/2(P_{\ell-1,BKS} + \widehat{P}_{\ell,BKS})$ we get that $|\pi_\gamma - p_{\ell-1}|$ is an of approximation of $|(2P_{\ell,BKS_{\tau-1}} - P_{\ell-1,BKS_{\tau-1}}) - P_{\ell-1,BKS_{\tau-1}}| = 2|P_{\ell,BKS_{\tau-1}} - P_{\ell-1,BKS_{\tau-1}}|$. More specifically, with probability at least $1 - 2/2^n$ it's the case that $\left||\pi_\gamma - p_{\ell-1}| - 2\left|P_{\ell,BKS_{\tau-1}} - P_{\ell-1,BKS_{\tau-1}}\right|\right| < 1/8n^{\alpha_5}$. By the construction of $\Delta$ in $\textsc{Exp}_2$ (line 7) and since $\ell \leftarrow \min \Delta$ (line 13), it's the case that $|p_\ell - p_{\ell-1}| > 1/n^{\alpha_6}$. Further, with probability at least $1 - 1/2^n$ we have that $\left||p_\ell - p_{\ell-1}| - |P_{\ell,BKS_{\tau-1}} - P_{\ell-1,BKS_{\tau-1}}|\right| \leq 1/8n^{\alpha_5}$. Combining the fact that $|p_\ell - p_{\ell-1}| > 1/n^{\alpha_6}$ with the approximation of $|p_\ell - p_{\ell-1}| \approx |P_{\ell,BKS_{\tau-1}} - P_{\ell-1,BKS_{\tau-1}}|$ and with the approximation $|\pi_\gamma - p_{\ell-1}| \approx 2\left|P_{\ell,BKS_{\tau-1}} - P_{\ell-1,BKS_{\tau-1}}\right|$ we get that with probability at least $1 - 4/2^n$ we have $|\pi_\gamma - p_{\ell-1}| \geq 2/n^{\alpha_6} - 1/2n^{\alpha_5}$ which is greater than $1/n^{\alpha_5}$ for all sufficiently large $n$, and therefore, by line 23 of $\textsc{Exp}_2$, $\overline{sk_\gamma} \leftarrow 1 - b$.

Obviously, in order for $\overline{sk} = sk$ in line 24 it must be the case $\overline{sk_\gamma} = sk_\gamma$ for each $\gamma \leq \psi$. A simple application of the union bound shows that $\overline{sk} = sk$ with a probability of at least $1 - \frac{5\psi}{2^n}$. By Assumption 20, we know that $\psi \leq n^s$, proving the lemma. □

□

Lemma 47 demonstrates that $\textsc{Exp}_2(E_n^1)$ is not likely to halt due to the fact that it improperly retrieved an embedded secret-key. However, we would like assurance that if $\textsc{Exp}_2$ finished on line 12 (page 37) all of the embedded keys that are likely to be retrieved, were in fact retrieved. In order for this to be the case, it is sufficient that for every iteration of the **for loop** in $\textsc{Exp}_2$ (lines 3 through 28), every approximation of $P_{t,BKS_{\tau-1}}$ that was actually calculated on line 6 was actually a good approximation. If this is the case, then it, along with Lemma 47, implies that with high likelihood $\textsc{Exp}_2(E_n^1)$ retrieved all of the secret-keys that are in some sense necessary to simulate $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$. Therefore, we will bound the probability that any of the approximations of $P_{t,BKS_{\tau-1}}$ are poor.

We define $\mathbf{BAD}_4$ to be the event that for at least one pair of values, $(t, \tau)$ that the approximations $p_t$, calculated in the $\tau$th iteration of the **for loop** in $\textsc{Exp}_2(E_n^1)$ (lines 3 through 28), is not a close approximation to the value $P_{t,BKS_{\tau-1}}$. We then show that the probability of the event is low.

**Definition 49.** *Given any $n$, $\mathbf{Env}_n \leftarrow Env(n)$ and $E \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ held, define $\mathbf{BAD}_4$ to be the event that in experiment $\textsc{Exp}_2(E_n^1)$, there exist values $\tau$ and $t$ such that the value $p_t = \mathit{ApproxErrorRate}(t, \emptyset, (PK, SK), \mathcal{IQ}, BK$ computed on line 6 of $\textsc{Exp}_2$ has the property that $|p_t - P_{t,BKS_{\tau-1}}| > 1/16n^{\alpha_5}$.*

**Claim 50.** *For all sufficiently large $n$, $\mathbf{Env}_n \leftarrow Env(n)$ and $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$ where $\overline{\mathbf{BAD}_1}$ held:*

$$\Pr_{\textsc{Exp}_2(E_n^1)}[\mathbf{BAD}_4] \leq n^{2q}/2^n$$

[13]It is always possible to make $n^{2\alpha_4}$ different queries of the form $(\mathbf{w}, pk, z)$, where $1 \leq z \leq n^{2\alpha_4}$ and $z$ is represented as a binary string. This is because $|pk| > 3 \cdot \alpha_0 \cdot \log n$. This implies that $|\psi| > \alpha_0 \cdot \log n$, and, therefore since $z$ is viewed as being in $\{0,1\}^{\psi/3}$, there are at least $2^{\frac{\alpha_0 \cdot \log n}{3}} \geq n^{2\alpha_4}$ different queries of the form $(\mathbf{w}, pk, z)$; the inequality holds since for all sufficiently large $n$: $n^{\alpha_0} >> n^{2\alpha_4}$ .

*Proof.* By Lemma 44, for any particular pair of values $\tau$ and $t$ the probability that $|p_t - P_{t,BKS_{\tau-1}}| > 1/16n^{\alpha_5}$ is less than $1/2^n$, for the value $p_t$ computed on line 6 of $\text{Exp}_2$ . We then note that there are at most $n^{2q}$ executions of line 6 in $\text{Exp}_2$ because $|KS| \leq n^q$ due to the fact that $\overline{\mathbf{BAD}_1}$ holds. The claim follows from an application of the union-bound. $\square$

## K.6 What has $\mathbf{Exp}_2$ Achieved?

Assuming no **BAD** events occurred, what has been achieved after $\text{Exp}_1$ and $\text{Exp}_2$? We have retrieved a number of public-/secret-key pairs for $\mathbf{g}$, $(pk, sk)$, that are embedded into the public-/secret-key pair $(PK, SK)$ that the adversary is interested in. A natural question to ask is if all such embedded keys have necessarily been retrieved, and the answer is no. There are several reasons why this might be the case. First, consider the case where there is a public-key $pk$ that is embedded into $PK$, but which is only "used" to embed an encryption $\mathbf{e}(pk, b, r)$ with a very small probability during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen $M$ and $R$. In this case there is only a small chance that the key $pk$ will turn up in $\text{Exp}_1$ and an even smaller chance that its corresponding secret-key $sk$ will be retrieved in $\text{Exp}_2$. However, we point out that if $pk$ is "used" very infrequently during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, then it is very unlikely that a value $\mathbf{e}(pk, b, r)$ has been embedded into the challenge ciphertext that the CCA#1 adversary is interested in. Another reason that a secret-key $sk$ corresponding to such a public-key may not be retrieved in $\text{Exp}_2$ is that the results of queries to $\mathbf{d}(sk, \cdot)$ made during an execution of $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK, \cdot)$ may not greatly effect the probability that $\mathbf{D}$ performs a *valid decryption.* In such a case, the hybridization experiment in $\text{Exp}_2$ would never detect a large probability gap in the probability that valid decryptions are made when the responses to queries of the form $\mathbf{e}(pk, b, r)$ are replaced with a random response $\mathbf{e}(pk, b', r')$ for randomly chosen $b'$ and $r'$. Since there is no large gap, $\text{Exp}_2$ will never retrieve $pk$'s corresponding secret-key $sk = \mathbf{g}^{-1}(pk)$.

Still assuming no **BAD** events occurred, at the end of $\text{Exp}_2$ the adversary has retrieved all of the secret-keys that are necessary to properly decrypt (with high probability) a ciphertext generated by an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ (for randomly chosen $M$ and $R$) when all of the responses to oracle queries $\mathbf{e}(pk, b, r)$ for $pk \in BKS$ are replaced with responses $\mathbf{e}(pk, b', r')$ for randomly chosen $b'$ and $r'$. Looking forward slightly, this suggests knowing the correct responses to queries of the form $\mathbf{d}(\mathbf{g}^{-1}(pk), \cdot)$ for $pk \in BKS$ are in some sense unessential in properly decrypting ciphertexts generated by executions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$.

## K.7 A Useful Subroutine: *RandCipher*

Given a successful completion of the first two parts of the experiment where no **BAD** events occurred, the adversary $A_1$ is should be able to use its decryption oracle to *properly decrypt* the perturbed ciphertexts that are generated from the same distribution as those perturbed ciphertexts generated in the final hybridization sub-experiment that took place in $\text{Exp}_2$. In this distribute, responses to queries to $\mathbf{e}(pk, b, r)$ for $pk \in BKS$ are replaced with $\mathbf{e}(pk, b', r')$ for randomly chosen $b'$ and $r'$. Going forward there will be many instances where we will need to generate ciphertexts from this same distribution, so we introduce the subroutine *RandCipher* that generates ciphertexts from exactly this distribution.

**Definition 51.** *Given an oracle $\mathcal{O} - (\mathbf{O}, R) \leftarrow \Upsilon$, $PK \in \{0,1\}^{n^{\rho_1}}$, $M \in \{0,1\}$, $R \in \{0,1\}^{n^{\rho_2}}$, a set $\mathcal{IQ}$ of immutable queries in $\mathbf{O}$ and a set BKS of public-keys in $\mathbf{g}$ we define the $\mathit{RandCipher}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{IQ})$ subroutine as follows :*

```
RandCipher^O(PK,M,R,BKS,IQ)
(1)    $w \leftarrow \zeta(|BKS|, n^q, 1)$
(2)    For each $i \leq w$
(3)        $(\ell_i, *, *) \leftarrow \zeta^{-1}(i)$
(4)        $pk_i \leftarrow Index(\ell_i, BKS)$
(5)        $b_i \in_R \{0,1\}$
(6)        $r_i \in_R \{0,1\}^{|pk_i|/3}$
(7)        $c_i \leftarrow \mathbf{e}(pk_i, b_i, r_i)$
(8)    $C \leftarrow \widehat{\mathbf{E}}^{\mathbf{O}}(PK, M, R, BKS, \mathcal{IQ}, c_1, .., c_w)$
(9)    Output $C$
```

*We call the queries made on line 7 of RandCipher* **replacement queries***, and their responses* **replacement responses***. We call the query/response pair a* **replacement query/response***.*

## K.8 Decrypting the Output of *RandCipher*

Now, as stated in the previous section, at the end of $\text{Exp}_2$ the adversary expects to be able to properly decrypt ciphertexts that are generated from the distribution constructed by *RandCipher*. We now show that ciphertexts that are constructed by a random execution of *RandCipher* are likely to decrypt properly.

**Lemma 52.** *For all sufficiently large $n$, for every $\mathbf{Env}_n \leftarrow Env(n)$, $E_n^1 \leftarrow \text{Exp}_1(Env_n)$ and $E_n^2 = (\tau^*, BKS^*, GKS^*, E_n^1) \leftarrow \text{Exp}_2(E_n^1)$ where $\wedge_{i=1}^4 \overline{\mathbf{BAD}_i}$ hold:*

$$\Pr_{\substack{M \in_R \{0,1\}, R \in_R \{0,1\}^{n^{\rho 2}} \\ C \leftarrow RandCipher^{\mathbf{O}}(PK, M, BKS^*, \mathcal{IQ})}} [\mathbf{D}^{(\mathbf{g},\mathbf{d})}(SK, C) = M] \geq 1 - \frac{3n^{2q}}{n^6}.$$

*Proof.* We observe that by the description of *RandCipher*, proving this lemma is equivalent to lower-bounding the value $1 - P_{w,BKS^*}$, where $w = \zeta(|BKS^*|, n^q, 1)$. To this end, we know that in the iteration of the **for loop** of $\text{Exp}_2(E_n^1)$ where $\tau = \tau^*$, we computed an approximation $p_w$ of $P_{w,BKS^*}$. Because $\overline{\mathbf{BAD}_3}$ holds, we know that $|p_0 - p_w| \leq \frac{w}{n^{\alpha_6}}$ for the values $p_0$ and $p_w$ calculated in the same iteration of the **for loop**. Further, because $\overline{\mathbf{BAD}_4}$ holds, it's the case that $|p_i - P_{i,BKS*}| \leq \frac{1}{16n^{\alpha_5}}$ for each $i$. Additionally, by Lemma 38 and Observation 39 it is the case that $P_{0,BKS^*} \leq n^{2q}/n^{\alpha_1 - 2} + n^{2q}/n^{\alpha_0}$. Combining all of these bounds we get that for all sufficiently large $n$: $P_{w,BKS^*} \leq \frac{n^{2q}}{n^{\alpha_1 - 2}} + \frac{n^{2q}}{n^{\alpha_0}} + \frac{2}{16n^{\alpha_5}} + \frac{w}{n^{\alpha_6}}$. Since $\overline{\mathbf{BAD}_1}$ holds we know that $|KS| \leq n^q$, and therefore $w \leq 2n^{2q}$. Therefore, for all sufficiently large $n$: $P_{w,BKS^*} \leq \frac{3n^{2q}}{n^6}$, and the result follows. □

## K.9 Surprising Query/Responses in Exp$_2$

As was the case with $\text{Exp}_1$, we are concerned with the possibility of *possibly surprising queries* in $\text{Exp}_2$ yielding surprising responses. Going forward the analysis of our experiment will often need to assume that none of these surprising responses have occurred, and therefore we need to bound the probability of such a bad event. The next lemma introduces and bounds the probability of the event $\mathbf{BAD}_5$, where $\mathbf{BAD}_5$ represents a surprising query/response occurring in $\text{Exp}_2$. Once we have established this bound we can proceed to the third part of the experiment.

**Lemma 53.** *For all sufficiently large $n$:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n) \\ E_n^1 \leftarrow \text{Exp}_1(\mathbf{Env}_n) \\ \text{Exp}_2(E_n^1)}} \left[\mathbf{BAD}_5 | \overline{\mathbf{BAD}_1}\right] \leq \frac{1}{n^{\alpha_0}},$$

*where $\mathbf{BAD}_5$ is the event that there is a large surprising query/response in the probabilistic experiment.*

*Proof.* Because of the conditioning on $\overline{\mathbf{BAD}_1}$, there are no surprising queries that occur during $\mathbf{Env}_n \leftarrow \mathit{Env}(n)$ or $E^1_n \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$. Therefore, the only place a surprising query can occur is during the experiment $\textsc{Exp}_2(E^1_n)$. Since there are no queries to $\mathbf{u}$ in $\textsc{Exp}_2$, the only way there can be a surprising query/response is if a possibly surprising query $\mathbf{d}(sk, c)$ is performed. By observation of $\textsc{Exp}_2$, the only place a surprising query can occur is in the sub-routine $\widehat{\mathbf{E}}$-*Err*, called by the sub-routine *ApproxErrorRate*. In this subroutine only the executions of $\widehat{\mathbf{E}}^{\mathbf{O}}$ and $\mathbf{D}^{\mathbf{O}}$ can cause surprising query/responses.

Consider an arbitrary $i$th query, $(\mathbf{d}, sk, c)$, performed in $\widehat{\mathbf{E}}$-*Err* during $\textsc{Exp}_2$ that is possibly surprising. Let $pk = \mathbf{g}(sk)$ We bound the probability that the query's response is surprising. We bound the information that can be known about the randomly chosen function $\mathbf{e}(pk, \cdot, \cdot)$ and the decryption function it specifies $\mathbf{d}(sk, \cdot)$. We will grossly over-estimate the upper-bound by supposing that every query to sub-oracles $\mathbf{g}, \mathbf{e}, \mathbf{d}$ and $\mathbf{u}$ reveals the value of $\mathbf{e}(pk, \cdot, \cdot)$ on a different element in the domain of $\mathbf{e}(pk, \cdot, \cdot)$. Similarly, we will assume that each query to $\mathbf{w}(pk, \cdot)$ reveals the value of $\mathbf{e}(pk, \cdot, \cdot)$ for $|sk|$ different elements in its range. Observation shows that $\mathit{Env}(n)$ performed at most $n^q$ queries to $\mathbf{g}$ or $\mathbf{e}$; next, $\textsc{Exp}_1$ performed at most $2n^{2\alpha_1+q}$ queries to $\mathbf{g}, \mathbf{e}, \mathbf{d}$ and $\mathbf{u}$; and finally, $\textsc{Exp}_2$ performers at most $2n^{4q+2\alpha_4} + 4n^{s+2\alpha_4+2q} + n^{s+2\alpha_4+q} \le n^{\alpha_2}$ queries to all of the sub-oracles $\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}$ and $\mathbf{w}$, where the inequality holds for all sufficiently large $n$.

Therefore, remembering that $\mathbf{e}(pk, \cdot, \cdot)$ is a randomly selected function from $\{0,1\} \times \{0,1\}^{|sk|}$ to $\{0,1\}^{3|sk|}$ for which at most $n^q + 2n^{2\alpha_1+q} - |sk|n^{\alpha_2}$ input/output pairing are known, the probability that the $i$th query in $\textsc{Exp}_2$ has a surprising response is upper-bounded by $\frac{n^{|sk|} - n^q - 2n^{2\alpha_1+q} - |sk|n^{\alpha_2}}{n^{3|sk|} - n^q - 2n^{2\alpha_1+q} - |sk|n^{\alpha_2}}$. Because the query is *possibly surprising* it is necessarily *large*, and this implies that $|sk| \ge \alpha_0$; therefore, our bound is necessarily smaller than $\frac{n^{\alpha_0} - n^q - 2n^{2\alpha_1+q} - |sk|n^{\alpha_2}}{n^{3\alpha_0} - n^q - 2n^{2\alpha_1+q} - |sk|n^{\alpha_2}}$. This bound is smaller than $\frac{n^{\alpha_0}}{n^{3\alpha_0} - n^{3\alpha_1}}$ for all sufficiently large $n$, by our restrictions on the values of $\alpha_0$, $\alpha_1$, $\alpha_2$, $s$ and $q$.

Finally, since there are at most $n^{\alpha_2}$ queries in $\textsc{Exp}_2(E^1_n)$, we can use the union-bound to bound the probability of surprising query/response occurring during $\textsc{Exp}_2(E^1_n)$ to be less than $\frac{n^{\alpha_0+\alpha_2}}{n^{3\alpha_0} - n^{3\alpha_1}}$, which is less than $\frac{1}{n^{\alpha_0}}$ for all sufficiently large $n$. □

## K.10 Bounding the Probability of Bad Events in $\mathbf{Exp}_1$ and $\mathbf{Exp}_2$

Before going on to $\textsc{Exp}_3$, we present several book-keeping lemmas that bound the probability that different configurations of **BAD** events occur in $\textsc{Exp}_1$ and $\textsc{Exp}_2$. Rather then presenting on lemma that bound the probability of any bad events, we break the lemma up into distinct lemmas. The first bounds the probability of the event $\overline{\mathbf{BAD}_1} \wedge \overline{\mathbf{BAD}_3} \wedge \overline{\mathbf{BAD}_5}$, and the second bounds the probability of $\overline{\mathbf{BAD}_2} \wedge \overline{\mathbf{BAD}_4}$ given that $\overline{\mathbf{BAD}_1} \wedge \overline{\mathbf{BAD}_3} \wedge \overline{\mathbf{BAD}_5}$ hold in the experiment. This split is useful going foreword, as we shall see when $\textsc{Exp}_3$ is described.

**Lemma 54.** *For all sufficiently large $n$:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow \mathit{Env}(n) \\ E^1_n \leftarrow \textsc{Exp}_{1,r_1}(\mathbf{Env}_n) \\ E^2_n \leftarrow \textsc{Exp}_{2,r_2}(E^1_n)}} \left[ \bigwedge_{i \in \{1,3,5\}} \overline{\mathbf{BAD}_i} \right] \ge 1 - \frac{3}{n^{\alpha_0}}.$$

*Proof.* This is just an application of Lemmas 32, 47 and 53. □

**Lemma 55.** *For all sufficiently large $n$:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow \mathit{Env}(n) \\ E^1_n \leftarrow \textsc{Exp}_{1,r_1}(\mathbf{Env}_n) \\ E^2_n \leftarrow \textsc{Exp}_{2,r_2}(E^1_n)}} \left[ \bigwedge_{i \in \{2,4\}} \overline{\mathbf{BAD}_i} \,\middle|\, \bigwedge_{i \in \{1,3,5\}} \overline{\mathbf{BAD}_i} \right] \ge 1 - \frac{3n^{2q}}{2^n},$$

*Proof.* We independently upper-bound the probability that $\mathbf{BAD}_2$, and $\mathbf{BAD}_4$ hold, and use the union bound to derive a lower bound on the probability that $\overline{\mathbf{BAD}_2} \wedge \overline{\mathbf{BAD}_4}$ holds.

First, we bound $\mathbf{BAD}_2$ by noting:

$$\begin{aligned}\Pr[\mathbf{BAD}_2 | \wedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}] &= \frac{\Pr\left[\mathbf{BAD}_2 \wedge \left(\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}\right)\right]}{\Pr[\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}]} \\ &\leq \frac{\Pr[\overline{\mathbf{BAD}_2}]}{\Pr[\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}]} \\ &\leq \frac{2/2^n}{1-\frac{3}{n^{\alpha_0}}} \\ &\leq \frac{4}{2^n},\end{aligned}$$

where the last two inequalities hold for all sufficiently large $n$, and follow from Lemmas 54 and 37.

Second, we bound $\mathbf{BAD}_4$ by noting:

$$\begin{aligned}\Pr[\mathbf{BAD}_4 | \wedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}] &= \frac{\Pr\left[\mathbf{BAD}_4 \wedge \left(\wedge_{i\in\{1,3,4,5\}}\overline{\mathbf{BAD}_i}\right)\right]}{\Pr[\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}]} \\ &\leq \frac{\Pr[\overline{\mathbf{BAD}_4}|\overline{\mathbf{BAD}_1}]}{\Pr[\wedge_{i\in\{3,5\}}\overline{\mathbf{BAD}_i}|\overline{\mathbf{BAD}_1}]} \\ &\leq \frac{\frac{n^{2q}}{2^n}}{1-\frac{5\cdot n^{q+s}}{2^n}-\frac{1}{n^{\alpha_0}}} \\ &\leq \frac{2n^{2q}}{2^n},\end{aligned}$$

where the second last inequality follows from Lemmas 50, 47 and 53, and holds for all sufficiently large $n$; and the last inequality stems from the fact that $1-\frac{5\cdot n^{q+s}}{2^n}-\frac{1}{n^{\alpha_0}} \geq 1/2$ for all sufficiently large $n$. □

## L Exp$_3$: Finding a New Secret-Key and a New Oracle

Now that the adversary has retrieved all of the secret-keys *sk* that are embedded into *SK* that are in some sense necessary to perform decryptions with **D**, it must use them to recreate a new secret-key $SK'$ that it can use to decrypt its challenge ciphertext. While for many PKEPs, it may seem obvious how this should be done, it is by no means obvious how this should be done in general. For example, consider traditional PKEPs based on different assumptions such as the system by Goldwasser-Micali [18] based on the Quadratic-Residuosity assumption and one based on the RSA trapdoor permutation assumption [33] [14], the adversary's approach to coming up with a secret-key $SK'$ will have to work for both these systems, even though it should be an easy case as there clearly no values *sk* that need to be embedded into $SK'$, as neither of these systems will query $\mathcal{O}$. Therefore, in order to construct a key $SK'$ the adversary will perform a brute-force search by looking at all possible seeds, and finding one for which **G** generates a key-pair $(SK', PK)$. However, in addition to a brute-force search through all possible seeds, the adversary needs to look at the queries that the different executions of **G** will make to **O** during the search, and ensure that the responses are consistent with both its knowledge of the oracle **O** and the adversary's observations of *SK* through the use of the decryption oracle. Finally, since the adversary knows that all of the keys pairs $(pk, sk) \in GKS$ that were retrieved in Exp$_2$ were embedded into $(PK, SK)$, then for each $(pk, sk) \in GKS$ the query/response $(<\mathbf{g}, sk>, pk)$ should be made during the execution of **G** for the retrieved key. The reader may be worried that we did not retrieve all of the secret-keys *sk* that might possibly be embedded into *SK* during Exp$_2$, and therefore the adversary cannot ensure that all such keys are embedded into the retrieved key $SK'$. For the moment, it may be easier for the reader suspend their disbelief and to assume that all such embedded keys were retrieved to help focus on

[14]Neither of these systems will make any queries to the oracle, and so will satisfy all of our required assumptions given on page 20

understanding how the adversary reconstructs the key $SK'$. Later, we will compensate for the fact that the adversary has not retrieved all the embedded keys, by having the adversary execute a modified decryption algorithm. However, for now we focus on constructing a key $SK'$, and this is the main goal of $\textsc{Exp}_3$.

In the third part of the experiment the adversary will generate all pairs of oracles and seeds that are consistent with the public-key $PK$ given to the adversary and its observations of the oracle $\mathbf{O}$ and the decryption oracle $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK,\cdot)$. It will then uniformly at random choose one such pair $(\mathcal{O}', SK')$, as there is no reason one of these pairs should be more likely than another. Because of the random method by which oracle $\mathcal{O}'$ and the seed $S'$ are chosen, with high probability over the choice it will be the case that with high probability $\mathbf{D}^{\mathbf{g}',\mathbf{d}'}(SK',C) = M$, for a ciphertext randomly generated by an execution of $\mathit{RandCipher}^{\mathbf{O}'}(PK, M, R, BKS^*, \mathcal{IQ})$ for randomly chosen $M$ and $R$. Remembering that if we are temporarily pretending that all of the secret-keys $sk$ that are embedded into $SK$ were retrieved in $\textsc{Exp}_2$, then $BKS^* = \emptyset$ and therefore the output of $\mathit{RandCipher}$ corresponds to the output of $\mathbf{E}^{\mathbf{O}'}(PK, M, R)$ for randomly chosen $M$ and $R$. Therefore, the adversary has found a way to reassemble the secret-keys $sk$ that were embedded into $SK$ so that they form a new functioning secret-key $SK'$, but decryption with $SK'$ works for encryptions made relative to the oracle $\mathbf{O}'$, and while $\mathbf{O}'$ and $\mathbf{O}$ will agree on all of the query/responses the adversary has seen prior to $\textsc{Exp}_3$, there are probably very few other queries whose responses are equal. For instance, for a pair $(pk, sk) \in GKS$ it will be the case that $\mathbf{g}(sk) = \mathbf{g}'(sk) = pk$, as the oracle $\mathcal{O}'$ that is selected is chosen to be consistent with known information about $\mathcal{O}$, but it is highly unlikely that $\mathbf{e}(pk, b, r) = \mathbf{e}'(pk, b, r)$ for randomly chosen $b$ and $r$. Because of this, it is highly unlikely that $\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\mathbf{O}'}(PK, M, R)$ for a randomly chosen $M$ and $R$, and therefore we need to do some work to show how to use $SK'$ to decrypt encryptions constructued from executions of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. In order to do this, after showing that the adversary can find $\mathcal{O}'$ and $SK'$, we show that the adversary can actually choose $\mathcal{O}'$ so that it is nearly identical to $\mathcal{O}$. Thus, for a randomly chosen $M$ and $R$ it's highly likely that if $\mathbf{E}^{\mathbf{O}}(PK, M, R) = C$ then $\mathbf{E}^{\mathbf{O}'}(PK, M, R) = C$. This has strong implications, as the adversary could then use the decryption algorithm to execute $\mathbf{D}^{\mathbf{g}',\mathbf{d}'}(SK', \cdot)$, and thus it could use this ability to decrypt the challenge ciphertext from the CCA#1 security definition. Unfortunately, things are not so simple. In order to generate an oracle that is nearly identical to $\mathbf{O}$, the adversary would have to have a complete description of it, whereas it is only allowed to query $\mathbf{O}$ a polynomial number of times. Therefore, the adversary will need to simulate access to $\mathbf{O}'$ by using its ability to query $\mathbf{O}$ and $\mathbf{u}$. However, the adversary's simulation will not be perfect: it will be unable to simulate several types of queries. From the vantage point of the adversary, the most crucial queries whose responses cannot be simulated are for queries to $\mathbf{d}'(sk', \cdot)$ for those values $sk'$ where $\mathbf{g}'(sk) \in BKS^*$. That is we cannot properly simulate queries $\mathbf{d}'(sk', \cdot)$ for values of $sk'$ that were not properly retrieved in $\textsc{Exp}_2$. The reason this is problematic is that it is quite likely that such queries will be made during an execution of $\mathbf{D}^{\mathbf{g}',\mathbf{d}'}(SK', \cdot)$, and this is the computation that the adversary interested is interested in. To compensate for this problem an alternate decryption algorithm is constructed that is unlikely to make such queries. We briefly give some intuition for why we can construct such an alternate decryption algorithm. If, after the end of $\textsc{Exp}_2$, the adversary were to randomly generate a ciphertext $C \leftarrow \mathit{RandCipher}^{\mathbf{O}}(PK, M, R, BKS^*, \mathcal{IQ})$ then it would be highly likely that $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK, C) = M$, as has been shown. However, the execution of $\mathit{RandCipher}$ effectively simulates an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ which responds to every query $(\mathbf{e}, pk, b, r)$ where $pk \in BKS^*$ with $\tilde{c} = \mathbf{e}(pk, \tilde{b}, \tilde{r})$ for randomly chosen $\tilde{b}$ and $\tilde{r}$. Therefore, if during the execution of $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK, C)$ there is a query $\mathbf{d}(\mathbf{g}^{-1}(pk), \tilde{c})$, and we were to respond with a random coin-flip as opposed to the appropriate response from the oracle, then presumably we could still expect this modified execution of $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK, C)$ to output $M$. Because of the similarity between $\mathbf{O}$ and $\mathbf{O}'$, the same holds true if we replace the oracle $\mathbf{O}$ with $\mathbf{O}'$ and $SK$ with $SK'$.

Thus, once we have shown how $\textsc{Exp}_3$ retrieves $\mathbf{O}'$ and $SK'$, we will show that the construction of an alternate probabilistic decryption algorithm $\widehat{\mathbf{D}}^{\mathbf{O}'}$ that does not need to make queries to $\mathbf{d}'(sk, \cdot)$ for $\mathbf{g}'(sk) \in BKS$. It is this algorithm that the adversary will use to decrypt the challenge ciphertext from the CCA#1 security definition, proving the main theorem.

## L.1 The Formal Description of $\mathbf{Exp}_3$

We now turn our attention to the issue of retrieving an oracle $\mathcal{O}'$ that is consistent with all of the query/responses in $GKS^*$ and $\mathcal{IQ}$, and relative to which there exists seed $S'$ such that $\mathbf{G}^{\mathbf{O}'}(S^*) = (PK, SK')$. We remind the reader that although we are describing this experiment in three parts, we really have in mind considering it one large experiment. Thus, it is permissible for the experiment to refer to queries that were made in the previous two experiments, even if they are not officially represented by an input to the experiment. This is done to prevent unnecessary notation.

We also note that in this section of the experiment we consider every possible oracle $\mathcal{O}'$ that can be generated by the random process $\Upsilon$ that is consistent with the responses to a number of previously made queries that the adversary has performed. Because each oracle $\mathcal{O}'$ is infinitely large, this may seem impossible even for our computationally unlimited adversary. However, we remind the reader that by Assumption 20, we know that no queries larger than $n^s$ have been made in our experiments. Therefore, given any oracle $\mathcal{O}'$, we are only interested in its definition for queries of sizes less than or equal to $n^s$. Therefore, in the third experiment when the adversary is to enumerate over all oracles, it need only enumerate over the finite prefixes of these oracles that contain all queries of sizes $n^s$ or less. Finally, we note that the expectation of the adversary during $\textsc{Exp}_3$ is that the likely event $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$ held in $\textsc{Exp}_1$ and $\textsc{Exp}_2$. The reason that it is not assumed that $\wedge_{i\in\{2,4\}}\overline{\mathbf{BAD}_i}$ held, is that we actually want these events to hold relative to the oracle selected in $\textsc{Exp}_3$, and this will be described in greater detail later.

```
$\textsc{Exp}_3(E_n^2 = (\tau^*, BKS^*, GKS^*, E_n^1))$
(1)   Let $E_n^1 = (\mathbf{Env}_n, KS, \mathcal{IQ}, \mathcal{E}, SQ))$
(2)   Let $\mathbf{Env}_n = (\mathcal{O}, PK, SK, S^*)$
(3)   Let $\mathit{KnownQueries}$ be the set of every query/response that has been made to $\mathcal{O}$ during the
      calculation of $E \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$, and then add all query/responses made in $\textsc{Exp}_2(E_n^1)$
      unless that query/response was made only during an execution of $\mathbf{D}^{\mathbf{O}}(SK, *)$.
(4)   Let $\mathit{KnownDecryptions} = \{(C, b)|$ There was an execution of $\mathbf{D}^{\mathbf{O}}(SK, C)$
      that resulted in $b \in \{0, 1, \perp\}$ during $\textsc{Exp}_2(E_n^1)\}$.
(5)   Let $\mathit{ValidEnvironments}$ be the set of every pair $(\mathcal{O}', S')$, for every oracle $\mathcal{O}' = (\mathbf{O}', R') \leftarrow$
      $\Psi$ and seed $S' \in \{0,1\}^n$ where:
(6)      Each $\mathcal{O}'$ is consistent with every query/response in $\mathit{KnownQueries}$;
(7)      $\mathbf{G}^{\mathbf{O}'}(S') = (PK, SK')$;
(8)      $\mathbf{G}^{\mathbf{O}'}(S')$ makes no surprising query/responses;
(9)      $\mathbf{G}^{\mathbf{O}'}(S')$ makes a query/respns. $(<\mathbf{g}', sk>, pk)$ for each $(sk, pk) \in GKS^*$;
(10)     $\mathbf{G}^{\mathbf{O}'}(S')$ make a query/response $(<\mathbf{g}', *>, pk)$ for for each $pk \in BKS^*$;
(11)     For every $(C, b) \in \mathit{KnownDecryptions}$, $\mathbf{D}^{\mathbf{O}'}(SK, C) = b$;
(12)     For every $(C, b) \in \mathit{KnownDecryptions}$, $\mathbf{D}^{\mathbf{O}'}(SK, C)$ makes no surprising
         query/responses.

(14)  Choose $(\mathcal{O}', S') \in_R \mathit{ValidEnvironments}$.
(15)  Let $(PK, SK') \leftarrow G^{\mathbf{O}'}(S')$.
(16)  Let $\mathcal{G}$ to be the set of query/responses made during $G^{\mathbf{O}'}(S')$.
(17)  Let $\mathcal{D}$ to be the set of all query/responses made during calls to $\mathbf{D}^{\mathbf{O}'}(SK', C)$ for each
      $(C, b) \in \mathit{KnownDecryptions}$.
(18)  Let $\mathbf{Env}'_n$ be the environment defined by $(\mathcal{O}', PK, SK', S')$.
(19)  Output $(\mathbf{Env}'_n, \mathcal{G}, \mathcal{D}, E_n^2)$
```

## L.2 A Clarification on Query/Response Notation

In Notn. 15, we formalized the notation for queries and responses to the sub-oracles $\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}$ or $\mathbf{w}$ of an oracle $\mathcal{O}$. However, there will now be many instances where we need to discuss queries to different oracle, as in $\textsc{Exp}_3$ above, and this leads to some ambiguity. Thus we attempt to clarify the differences between these queries and responses.

**Notation 56.** *Given two oracles $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u})$ and $\mathcal{O}' = (\mathbf{g}', \mathbf{e}', \mathbf{d}', \mathbf{w}', \mathbf{u}')$ then the for any $o \in \{\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w}, \mathbf{u}\}$ it's the case for every value $q$ and $r$, $(o, q)$ and $(o', q)$ are considered the same query and $(< o, q >, r)$ and $(< o', q >, r)$ are considered the same query/response. For example, if we had a set $S = \{(\mathbf{e}, pk, b, r)\}$, then it is the case that $(\mathbf{e}', pk, b, r) \in S$.*

## L.3 The Running Example for $\mathbf{Exp}_3$

We again return the running example which uses the PKEP $(\mathfrak{G}, \mathfrak{E}, \mathfrak{D})$ that was introduced on Page 27.

**Running Example 4.** *We remind the reader that in our running example, we have a public-/secret-key pair $(PK, SK)$ where $PK = (pk_0, ..., pk_6, S_8, S_9)$ and $SK = (sk_0, ..., sk_6, k_1, k_2)$ where $k_1 = \mathbf{e}(pk_6, 0, S_8)$ and $k_2 = \mathbf{e}(pk_6, 0, S_9)$. Further, at the end of Running Example 3 we assumed that after* $\textsc{Exp}_2$ *it would be the case that $GKS^* = \{(pk_i, sk_i) | 3 \le i \le 6\}$, and therefore $BKS^* = \{pk_0, pk_1, pk_2\}$.*

*In running* $\textsc{Exp}_3$ *we hope to find a new oracle $\mathcal{O}'$ that is consistent with all of the queries that have been made in* $\textsc{Exp}_1$ *and* $\textsc{Exp}_2$*, and relative to which there exists a seed $S'$ such that $(SK', PK) \leftarrow \mathfrak{G}^{\mathbf{O}'}(S')$. Let's consider an execution of* $\textsc{Exp}_3$*. On line 14 of* $\textsc{Exp}_3$ *an oracle $\mathcal{O}'$ and seed $S'$ are randomly selected so that they are consistent with all query/responses to $\mathcal{O}$ that were known to the adversary from* $\textsc{Exp}_1$ *and* $\textsc{Exp}_2$*, and which are consistent with all observations that the adversary has made that are potentially dependent on the responses to queries to $\mathcal{O}$. More specifically, we know that the oracle $\mathcal{O}'$ that is selected will have the property that $\mathbf{e}'(pk_6, 0, S_8) = k_1$ and $\mathbf{e}'(pk_6, 0, S_9) = k_2$, as these queries were made in* $\textsc{Exp}_1$*. Similarly, $\mathbf{g}'(sk_i) = pk_i$ for $3 \le i \le 6$ the queries $\mathbf{g}(sk_i) = pk_i$ for $3 \le i \le 6$ were made in* $\textsc{Exp}_2$ *(this is guaranteed by the presence of the corresponding key-pairs in the set $GKS^*$). It is also the case that there exists strings $sk'_0, sk'_1$ and $sk'_2$ relative to which $\mathbf{g}'(sk'_i) = pk_i$ for $i \le 3$; this is because there must exist a string $S'$ for which $\mathfrak{G}^{\mathbf{O}'}(S') = (SK', PK = (pk_0, ..., pk_6, S_8, S_9))$, and if such a $PK$ is output, such $sk'_i$ must exist. Let $S'$ be the seed that was chosen by* $\textsc{Exp}_3$*, and let $SK' = (sk'_0, sk'_1, sk'_2, sk'_3, .., sk'_6, k'_1, k'_2)$. We do point out that it is highly unlikely that $sk_0 = sk'_0$, $sk_1 = sk'_1$ or $sk'_2 = sk_2$, but that the rest of the secret-key $SK'$ will necessarily be identical to $SK$, as per the previous discussion.*

*Now let's consider how the key $SK'$ would perform in several different decryption tasks. By Assumption 23 (page 21), it will necessarily be the case that $\mathfrak{D}^{\mathbf{O}'}(SK', \mathfrak{E}^{\mathbf{O}'}(PK, M, R)) = M$ for any $M$ and $R$ of the appropriate length. It is not hard to see that it will necessarily be the case that if $\mathfrak{D}^{\mathbf{O}'}(SK', C' = RandCipher^{\mathbf{O}'}(PK, M, R, BKS^*, \mathcal{IQ})) = M$, for the same reason that its always the case that $\mathfrak{D}^{\mathbf{O}}(SK', C = RandCipher^{\mathbf{O}}(PK, M, R, BKS^*, \mathcal{IQ})) = M$: suppose $C' = (1, k_1, k_2, c'_1, \ldots, c'_5)$, then it will necessarily be the case that $\mathbf{d}'(sk_i, c'_i) = M$ for $3 \le i \le 5$, and therefore when $\mathfrak{D}$ calculates $Majority(\mathbf{d}'(sk'_1, c'_1), \ldots, \mathbf{d}'(sk'_5, c'_5))$, the result will necessarily be $M$. It is also not hard to see that it is* unlikely *that $\mathfrak{D}^{\mathbf{O}'}(SK', \mathfrak{E}^{\mathbf{O}}(PK, M, R)) = M$ for most $M$ and $R$. To see this, suppose that $C = (1, k_1, k_2, c_1, ..., c_5) = \mathfrak{E}^{\mathbf{O}}(PK, M, R)$ and note that $\mathfrak{D}^{\mathbf{O}'}(SK', C)$ will output $Majority(\mathbf{d}'(sk'_1, c_1), \ldots, \mathbf{d}'(sk'_5, c_5))$. Remembering that $c_i = \mathbf{e}(pk_i, M, r)$* **and not** *$c_i = \mathbf{e}'(pk_i, M, r)$ and that $\mathbf{e}'$ is effectively a length-tripling random one-to-one function, the probability that $c_i$ is in the image of $\mathbf{e}'(pk_i, \cdot, \cdot)$ is negligible, and so, by its definition, the output of $\mathbf{d}'(sk'_i, c_i) = \perp$, and therefore $\mathfrak{D}^{\mathbf{O}'}(SK', C)$ is likely to output $\perp$. Finally, consider $\mathfrak{D}^{\mathbf{O}}(SK', \mathfrak{E}^{\mathbf{O}}(PK, M, R))$: we note that we expect the output in this situation to be $M$. This might seem surprising as the key $SK'$ was generated relative to $\mathbf{O}'$, but is being used to decrypt relative to $\mathbf{O}$. The important observation to make is that $SK' = (sk'_0, ..., sk'_6, k'_1, k'_2)$ and $SK = (sk_0, .., sk_6, k_1, k_2)$ disagree only in that $sk_0 \ne sk'_0, sk_1 \ne sk'_1$ and $sk_2 \ne sk'_2$. Therefore, if $\mathfrak{E}^{\mathbf{O}}(PK, M, R) = C = (1, k_1, k_2, c_1, ..., c_6)$, then when $\mathfrak{D}$ computes $M_i = \mathbf{d}(sk'_i, c_i)$ for $1 \le ileq5$ it will be the case that $M_i = M$ for $3 \le i \le 5$ since $sk'_i = sk_i$ and $pk'_i = pk_i$ in those cases and thus $Majority(M_1, ..., M_5) = M$ independent of the values of $M_1$ and $M_2$. It should be notes that $M_1$ and $M_2$ are likely to be $\perp$ because it is very unlikely that $\mathbf{g}(sk'_1) = pk'_1$ or $\mathbf{g}(sk'_2) = pk'_2$.*

*Therefore, we see in our case that we cannot use $SK'$ to decrypt our challenge ciphertext relative to $\mathbf{O}'$, as this is unlikely to succeed. However, we can use $SK'$ to decrypt our challenge ciphertext relative to $\mathbf{O}$, even though $SK'$ was constructed relative to $\mathbf{O}'$. We point this out, only so later we can justify why finding $\mathcal{O}'$ at the end of* $\textsc{Exp}_3$ *is not sufficient, and that we need a further step.*

## L.4 Decrypting with $\mathcal{O}'$ and $SK'$

We will demonstrate that it is likely that given the newly selected oracle $\mathcal{O}'$ and secret-key $SK'$ , the execution of $\mathbf{D}^{\mathbf{O}'}(SK', \cdot)$ is likely to properly decrypt ciphertexts generated by a random execution of $\mathit{RandCipher}^{\mathbf{O}'}(PK, M, R, BKS^*, \mathcal{IQ})$, for randomly chosen $M$ and $R$. This is done by showing that, assuming that very likely event $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ occurred in the first two parts of the experiment, from the perspective of the adversary it is equally likely that it is interacting with the environment $\mathbf{Env}'$ as $\mathbf{Env}$, and thus if $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ is likely to properly decrypt ciphertexts of $\mathit{RandCipher}^{\mathbf{O}}(PK, M, R, BKS^*, \mathcal{IQ})$ then it is likely that that $\mathbf{D}^{\mathbf{O}'}(SK', \cdot)$ properly decrypts ciphertexts of $\mathit{RandCipher}^{\mathbf{O}'}(PK, M, R, BKS^*, \mathcal{IQ})$. The reason for this is that $\mathbf{Env}'_n$ is uniformly chosen from the set of all possible environments that are consistent with the adversary's observations, and from the adversary's perspective each environment in such a set is equally likely to be the one it has been interacting with.

**Notation 57.** *In order to be able to specify the random choices that are being made in a given experiment* $\textsc{Exp}_i$ *(resp. probabilistic algorithm Alg), we denote by* $\textsc{Exp}_{i,r}$ *(resp.* $Alg_r$*) an execution of* $\textsc{Exp}_i$ *(resp. Alg) where the string* $r$ *specifies the random bits that are used in the experiment (resp. algorithm).*

*In all of the experiments and algorithms where this notation is used the mapping from the random bits in the specified string to random choices in the experiments can be accomplished with standard procedures, and so we do not detail how this is done.*

**Lemma 58.** *For all sufficiently large* $n$*:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n)\\ r_1,r_2 \in_R \{0,1\}^* \\ E_n^1 \leftarrow \textsc{Exp}_{1,r_1}(\mathbf{Env}_n) \\ E_n^2 \leftarrow \textsc{Exp}_{2,r_2}(E_n^1) \\ E_n^3 \leftarrow \textsc{Exp}_3(E_n^2)}} \left[ \text{Event} \left( \bigwedge_{1\le i\le 5} \overline{\mathbf{BAD}'_i} \right) \text{ occurs in } \left\{ \begin{matrix} E_n'^1 \leftarrow \textsc{Exp}_{1,r_1}(\mathbf{Env}'_n) \\ E_n'^2 \leftarrow \textsc{Exp}_{2,r_2}(E_n'^1) \end{matrix} \right\} \middle| \bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i} \right] \ge 1 - \frac{3n^{2q}}{2^n},$$

*where the* $\{\mathbf{BAD}'_i\}_{1\le i\le 5}$ *events are the natural analogs of the events* $\{\mathbf{BAD}_i\}_{1\le i\le 5}$ *except they occur in* $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ *and* $\textsc{Exp}_{2,r_2}(E_n'^1)$*, as opposed to* $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ *and* $\textsc{Exp}_{2,r_2}(E_n'^1)$*.*

*Proof.* Observe that $\mathcal{O}'$ and $S'$ are chosen randomly in $\textsc{Exp}_3$ so that they are consistent with all of the observations of the oracle $\mathcal{O}$ and the seed $S$ that the adversary is able to observe or deduce during $\textsc{Exp}_1$ or $\textsc{Exp}_2$ because of its direct interaction with the oracle $\mathcal{O}$, the decryption oracle $\mathbf{D}^{\mathcal{O}}(SK, \cdot)$, and its knowledge of the public-key $PK$. They are also chosen assuming that no surprising queries were made by the decryption oracle in $\textsc{Exp}_2$, which is satisfied in this lemma by our conditioning on $\overline{\mathbf{BAD}_5}$.

It is easily observed that $\overline{\mathbf{BAD}'_1}$,$\overline{\mathbf{BAD}'_3}$ and $\overline{\mathbf{BAD}'_5}$ necessarily hold. This is because these events are witnessed by query/responses to $\mathcal{O}$ and the decryption oracle that the adversary makes. Therefore, if all of the queries and responses that occur in $\textsc{Exp}_{1,r_1}(\mathbf{Env}_n)$ and $\textsc{Exp}_{2,r_2}(E_n^1)$. occur in $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ and $\textsc{Exp}_{2,r_2}(E_n'^1)$, then the corresponding events are guaranteed to hold. This is not quite true for event $\overline{\mathbf{BAD}'_5}$, as the adversary does not have the queries and responses to $\mathcal{O}$ that were made by the decryption oracle, however the experiment assumes the likely case that no surprising responses were made during $\textsc{Exp}_2$.

- $\mathbf{BAD}'_1$ and $\mathbf{BAD}'_5$ denote the event that a surprising query/response occurred in $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ or $\textsc{Exp}_{2,r_2}(E_n'^1)$ respectively. $\overline{\mathbf{BAD}_1}'$ and $\overline{\mathbf{BAD}'_5}$ hold because there are no surprising queries in $\textsc{Exp}_{2,r_2}(E_n'^1)$ or $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ by the selection of $\mathcal{O}'$ and $S'$ in $\textsc{Exp}_3(E_n^2)$. In order to observe this, we note that the response each query to $\mathcal{O}$ that is made in either $\textsc{Exp}_{1,r_1}(\mathbf{Env}_n)$ or $\textsc{Exp}_{2,r_2}(E_n^1)$ *by the adversary, but not during an execution of* $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$ will have an identical response as the corresponding query to $\mathcal{O}'$ that is made during $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ or $\textsc{Exp}_{2,r_2}(E_n'^1)$ because of the design of $\textsc{Exp}_3$ on lines 3 and 6. Further, for $(C, b) \in \mathit{KnownDecryptions}$ it is the case that $\mathbf{D}^{\mathbf{O}}(SK, C) = \mathbf{D}^{\mathbf{O}'}(SK', C)$ by line 11, and therefore the executions of $\textsc{Exp}_{1,r_1}(\mathbf{Env}'_n)$ and $\textsc{Exp}_{2,r_2}(E_n'^1)$ will mirror those of $\textsc{Exp}_{1,r_1}(\mathbf{Env}_n)$ and $\textsc{Exp}_{2,r_2}(E_n^1)$, with the possible exception of queries to $\mathcal{O}$ that are made during the execution of $\mathbf{D}^{\mathbf{O}'}(SK', \cdot)$ Because $\mathbf{BAD}_5$ also holds, we have line 12 of $\textsc{Exp}_3$ to ensure $\mathbf{D}^{\mathbf{O}'}(SK', C)$ does not make any surprising query/responses for any $(C, b) \in \mathit{KnownDecryptions}$.

Because $\overline{\mathbf{BAD}_1}$ and $\overline{\mathbf{BAD}_5}$ hold, there were no surprising responses in the $\text{Exp}_{1,r_1}(\mathbf{Env}_n)$ or $\text{Exp}_{2,r_2}(E_n^1)$. Since the same responses were chosen to identical queries in $\text{Exp}_{1,r_1}(\mathbf{Env}'_n)$ and $\text{Exp}_{2,r_2}(E_n'^1)$ it is necessarily the case that $\overline{\mathbf{BAD}'_1}$ and $\overline{\mathbf{BAD}'_5}$ also hold.

- $\mathbf{BAD}'_3$ denotes the event that $\text{Exp}_{2,r_2}(E_n'^1)$ improperly retrieves a secret-key $\overline{sk}$ and halts on line 25 of $\text{Exp}_2$. $\overline{\mathbf{BAD}'_3}$ holds for similar reasons to $\overline{\mathbf{BAD}'_1}$ and $\overline{\mathbf{BAD}'_5}$. Because in $\text{Exp}_3$ we have that $\mathcal{O}'$ and $SK'$ are chosen to be consistent with *KnownQueries* and $\mathbf{D}^{\mathbf{O}'}(SK', C) = \mathbf{D}^{\mathbf{O}}(SK, C)$ for each $(C, b) \in$ *KnownDecryptions* and the fact that $\overline{\mathbf{BAD}_3}$ holds, $\text{Exp}_{2,r_2}(E_n'^2)$ will finish without halting on line 25.

It remains to show that with high probability $\overline{\mathbf{BAD}'_2}$, and $\overline{\mathbf{BAD}'_4}$ hold. These events are not witnesses by previous query/responses and are not ensured by the design of $\text{Exp}_3$, and therefore it is not the case that they immediately hold based on the results of $\text{Exp}_{1,r_1}(\mathbf{Env}_n)$ and $\text{Exp}_{2,r_2}(E_n^1)$. However, it is easy to see that they hold with the same probabilities as $\overline{\mathbf{BAD}_2}$ and $\overline{\mathbf{BAD}_4}$.

To observe that the stated probability bounds on $\mathbf{BAD}'_2$ and $\mathbf{BAD}'_4$ hold, note that during $\text{Exp}_{1,r_1}(\mathbf{Env}_n)$ and $\text{Exp}_{2,r_2}(E_n^1)$ the adversary learns very little information about $\mathcal{O}$ or $SK$. In fact, it learns only what is revealed to it through queries to $\mathcal{O}$ and the results of queries to the decryption oracle $\mathbf{D}^{\mathbf{O}}(SK, C)$. By its construction, the set *ValidEnvironments* represents the set of all possible environments that are consistent with the adversary's information about $\mathbf{Env}_n$ at the end of $\text{Exp}_{1,r_1}(\mathbf{Env}_n)$ and $\text{Exp}_{2,r_2}(E_n^1)$. Because $S^*$ was chosen uniformly at random from $\{0,1\}^n$ and $\mathcal{O}$ was constructed by the random process $\Upsilon$, from the adversary's perspective the probability that it has been interacting with any particular environment from *ValidEnvironments* is *equally likely.* Therefore, from the adversary's perspective, it has conducted the experiments $\text{Exp}_{2,r_2}$ and $\text{Exp}_{1,r_1}$ relative to a randomly chosen element $\mathbf{Env}_n \in$ *ValidEnvironments*. However, this is exactly what $\mathbf{Env}'_n$ represents, a randomly chosen element in *ValidEnvironments*. Therefore, the bounds on the probability of the events on $\mathbf{BAD}_2$ and $\mathbf{BAD}_4$ relative to $\mathbf{O}$ must hold relative for $\mathbf{BAD}'_2$ and $\mathbf{BAD}'_4$ relative to $\mathbf{O}'$. Therefore, the bound in Lemma 55 proves the lemma. □

Because we will be interested in cases where $\wedge_{1\le i\le 5}\overline{\mathbf{BAD}'_i}$ hold relative to the oracles output by $\text{Exp}_3$, independently of whether or not $\wedge_{1\le i\le 5}\overline{\mathbf{BAD}_i}$ hold relative to the original experiments $\text{Exp}_1$ and $\text{Exp}_2$ we define the event $\overline{\mathbf{BAD}_6}$ to denote it.

**Definition 59.** *For any n, given any experiment:*

$$\begin{aligned}
&r_1, r_2 \in_R \{0,1\}^* \\
&\mathbf{Env}_n \leftarrow Env(n) \\
&E_n^1 \leftarrow \text{Exp}_{1,r_1}(\mathbf{Env}_n) \\
&E_n^2 \leftarrow \text{Exp}_{2,r_2}(E_n^1) \\
&E_n^3 = (\mathbf{Env}'_n, \mathcal{G}, \mathcal{D}, \text{Exp}_2(\text{Exp}_1(\mathbf{Env}_n))) \leftarrow \text{Exp}_3(E_n^2),
\end{aligned}$$

*where* $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ *holds, define the event* $\mathbf{BAD}_6$ *to be that* $\bigwedge_{1\le i\le 5} \mathbf{BAD}'_i$ *occurs in in the execution of*

$$\begin{aligned}
E_n'^1 &\leftarrow \text{Exp}_{1,r_1}(\mathbf{Env}'_n) \\
E_n'^2 &\leftarrow \text{Exp}_{2,r_2}(E_n'^1)
\end{aligned}$$

*where the events* $\{\mathbf{BAD}'_i\}_{1\le i\le 5}$ *are the natural analogs of the events* $\{\mathbf{BAD}_i\}_{1\le i\le 5}$ *except they occur in the execution of* $\text{Exp}_{1,r_1}(\mathbf{Env}'_n)$ *and* $\text{Exp}_{2,r_2}(E_n'^1)$*, as opposed to* $\text{Exp}_{1,r_1}(\mathbf{Env}_n)$ *and* $\text{Exp}_{2,r_2}(E_n^1)$*.*

**Corollary 60 (of Lemma 58).** *For all sufficiently large $n$:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n) \\ r_1, r_2 \in_R \{0,1\}^* \\ E_n^1 \leftarrow \text{EXP}_{1,r_1}(\mathbf{Env}_n) \\ E_n^2 \leftarrow \text{EXP}_{2,r_2}(E_n^1) \\ E_n^3 \leftarrow \text{EXP}_3(E_n^2)}} \left[ \mathbf{BAD}_6 \middle| \bigwedge_{i \in \{1,3,5\}} \overline{\mathbf{BAD}_i} \right] \leq \frac{3n^{2q}}{2^n}$$

Assuming $\overline{\mathbf{BAD}_6}$ holds at the end of $\text{EXP}_3$, then it's likely that if ciphertexts are generated by *RandCipher* relative to the oracle $\mathcal{O}'$ produced, then it's the case that they will decrypt properly when given to the function $\mathbf{D}^{\mathbf{O}'}(SK', \cdot)$. The corollary below formalizes this.

**Corollary 61 (of Lemma 52 and Defn. 59 ).** *For all sufficiently large $n$ and all*

$$\mathbf{Env}_n \leftarrow Env(n),\ E_n^1 \leftarrow \text{EXP}_1(\mathbf{Env}_n),\ E_n^2 \leftarrow \text{EXP}_2(E_n^1),$$

$$E_n^3 = (\mathbf{Env}'_n = (\mathcal{O}', PK, SK', S'), \mathcal{G}, \mathcal{D}, E_n^2) \leftarrow \text{EXP}_3(E_n^2),$$

*where $\bigwedge_{i \in \{1,3,5,6\}} \overline{\mathbf{BAD}_i}$ holds,*

$$\Pr_{\substack{M \in \{0,1\}, R \in \{0,1\}^{n^{\rho_2}} \\ C \leftarrow RandCipher^{\mathbf{O}'}(PK, M, R, BKS^*, \mathcal{IQ})}} \left[ \mathbf{D}^{\mathbf{O}'}(SK', C) = M \right] \geq 1 - \frac{3n^{2q}}{n^{\alpha_6}}.$$

We have now shown how the adversary can generate a secret-key $SK'$ and oracle $\mathcal{O}'$ that it can use to execute $\mathbf{D}^{\mathbf{O}'}(SK', \cdot)$ in order to properly decrypt ciphertexts that are generated by calls to $RandCipher^{\mathbf{O}'}(PK, M, R, BKS, IQ)$. However, the goal of the adversary is to decrypt a challenge ciphertext generated by an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. In order to do this, we must construct a third oracle $\widehat{\mathbf{O}}$ that has the property that most of its responses to queries are identical to the equivalent query to $\mathbf{O}$. Further, we need to still ensure that relative to $\widehat{\mathbf{O}}$ it is the case that $\mathbf{G}^{\widehat{\mathbf{O}}}(S') = (SK', PK)$. We will show that this can be done, and the result is that it will be likely that $\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$, for randomly chosen $M$ and $R$.

In order to choose an oracle $\widehat{\mathbf{O}}$ we do the following: we choose $\mathbf{O}'$ and $S'$ as stated in $\text{EXP}_3$, and we extract from it all of the query/responses that are made during the execution of $\mathbf{G}^{\mathbf{O}'}(S') \rightarrow (SK', PK)$. We then "paste" these query responses into our original oracle $\mathbf{O}$, by forcing $\mathbf{O}$ to be consistent with these query/responses. Such changes can invalidate other parts of the oracle, for example by changing the one-to-one properties of $\mathbf{g}$ and $\mathbf{e}$ or the assurance that $\mathbf{d}$ will properly decrypt any of the encryptions made by $\mathbf{e}$. Such problems are corrected, and the resulting oracle is called $\widehat{\mathbf{O}}$.

Based on the above it would seem that all that would remain for the adversary would be to execute $\mathbf{D}^{\widehat{\mathbf{O}}}(SK, C)$ for the challenge ciphertext $C$ it is given. However, as previously alluded to, the adversary will have to simulate the responses of the queries to $\widehat{\mathbf{O}}$, and this will not be able to be done perfectly, so it will necessitate a new decryption algorithm. This is because as it is not necessarily the case $BKS^* = \emptyset$ and therefore there are secret-keys, $sk$, corresponding to public-keys in $BKS^*$ that were never retrieved in $\text{EXP}_2$. However, for each $pk \in BKS^*$, there must exist a value $sk'$ for which $\widehat{\mathbf{g}}(sk') = pk$, as the selection criteria for $S'$ forced the execution of $\mathbf{G}^{\mathbf{O}'}(S')$ to perform a query $\mathbf{g}'(*)$ with the response $pk$; it is queries to $\widehat{\mathbf{d}}(sk', \cdot)$ that are difficult to properly simulate.

In the next section we will show how to choose an oracle $\widehat{\mathbf{O}}$, and then we will show that it is in fact likely that $\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$. We will then spend the remainder of the chapter showing how the second part of the adversary, $A_2$, can decrypt $C$ with a new decryption algorithm while simulating $\widehat{\mathbf{O}}$, proving Theorem 26.

## L.5 How the Adversary Selects an Oracle $\widehat{\mathbf{O}}$ in $\widehat{\mathbf{Exp}}_3$

We would like the adversary to select an oracle $\widehat{\mathbf{O}}$ in a way that makes it is as similar to $\mathbf{O}$ as possible, but in a manner that ensures that it is randomly selected from the same distribution as $\mathbf{O}'$ is selected in $\text{EXP}_3$.

We show how this can be done and then show that in this case it is highly probable that $\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\hat{\mathbf{O}}}(PK, M, R)$.

We will modify $\textsc{Exp}_3$ to construct $\widehat{\textsc{Exp}_3}$, it will be $\widehat{\textsc{Exp}_3}$ that the adversary actually runs as the third part of the experiment. To construct $\widehat{\textsc{Exp}_3}$, $\textsc{Exp}_3$ is modified so that after it selects an environment $\mathbf{Env}'_n = (\mathcal{O}', PK, SK', S')$ on line 18 (Page 48), rather than outputting it the experiment will select a new environment $\widehat{\mathbf{Env}}_n = (\widehat{\mathcal{O}}, PK, SK', S')$ where $\widehat{\mathcal{O}}$ is uniformly chosen from the set of all oracles

$$\{\tilde{\mathcal{O}} | \tilde{\mathcal{O}} \leftarrow \Upsilon \text{ and } \tilde{\mathcal{O}} \text{ is consistent with all of the query/responses in } \mathit{KnownQueries}, \mathcal{G} \text{ and } \mathcal{D}\}.$$

The distributions on the outputs of $\widehat{\textsc{Exp}_3}(E_n^2)$ and $\textsc{Exp}_3(E_n^2)$ are identical. This is observed by noting that the only information specified about $\mathcal{O}'$ in $\textsc{Exp}_3$ is that it is consistent with the query/responses in *KnownQueries*, $\mathcal{G}$ and $\mathcal{D}$. Therefore, since $\mathcal{O}'$ was chosen uniformly at random from the set of all oracles that were consistent with these choices, each is equally likely to be the oracle selected in $\textsc{Exp}_3$. We are simply randomly choosing a new oracle from this distribution.

The reason we select a new oracle $\widehat{\mathcal{O}}$, is that we will choose $\widehat{\mathcal{O}}$ in a manner that makes it mostly consistent with $\mathcal{O}$ (i.e. most responses are equivalent for equivalent queries). This may seem contradictory, but remember that $\mathcal{O}$ itself was chosen randomly, and we do not need pairwise independence between the random selection of $\mathcal{O}$ and $\widehat{\mathcal{O}}$; therefore, we can use the "unused" randomness from the selection of $\mathcal{O}$ in $Env(n)$ and use it to help select $\widehat{\mathcal{O}}$. However, we need to also ensure that $\widehat{\mathcal{O}}$ is consistent enough with $\mathcal{O}'$ that it's still the case that $\mathbf{G}^{\hat{\mathbf{O}}}(S') = (SK', PK)$, and so the adversary has a secret-key corresponding to $PK$ relative to this new oracle.

In the next section we introduce the random process **ConsolidateOrac** that describes how $\widehat{\textsc{Exp}_3}$ randomly chooses an oracle $\widehat{\mathcal{O}}$ from the set described above, but with the property that it is as consistent as possible with the oracle $\mathcal{O}$ in $\mathbf{Env}_n$, while maintaining the necessary consistency with $\mathcal{O}'$. As its name suggests, **ConsolidateOrac** effectively consolidates the two oracles into one. **ConsolidateOrac** is constructed assuming that $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ held in $\textsc{Exp}_2$ and $\textsc{Exp}_1$ before $\textsc{Exp}_3$ was called, but this scenario is the likely one and the only one in which we will be interested.

## L.6 ConsolidateOrac: Choosing the Oracle $\widehat{\mathcal{O}}$ in $\widehat{\mathbf{Exp}}_3$

We describe the random-process **ConsolidateOrac** that generates an oracle $\widehat{\mathcal{O}}$ that is highly consistent with the oracle $\mathcal{O}$ in $\mathbf{Env}_n$. It takes as input the oracle $\mathcal{O}$ and the sets *KnownQueries*, $\mathcal{G}$ and $\mathcal{D}$ that are generated in $\textsc{Exp}_3$ during the experiment an experiment $\mathbf{Env}_n \leftarrow Env(n)$, $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$, $E_n^2 \leftarrow \textsc{Exp}_2(E_n^1)$, and $\textsc{Exp}_3(E_n^2)$, conditioned on $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ holding.

We call the process **ConsolidateOrac** to denote that it is consolidating the oracles $\mathcal{O}$ along with the oracle query/response pairs in $\mathcal{G}$ and $\mathcal{D}$ that were found by making queries to $\mathcal{O}'$ in $\textsc{Exp}_3(E_n^2)$. The intuition behind **ConsolidateOrac** is now described. **ConsolidateOrac** starts by making the oracle $\widehat{\mathcal{O}}$ consistent with all of the query/responses in *KnownQueries*, $\mathcal{G}$ and $\mathcal{D}$. This ensures that it is consistent with the observed properties of $\mathcal{O}$ from $\textsc{Exp}_1$ and $\textsc{Exp}_2$, and that $\mathbf{G}^{\hat{\mathbf{O}}}(S') = (SK', PK)$. Once this has been done, it attempts to make as many of the remaining responses in $\widehat{\mathcal{O}}$ consistent with those in $\mathcal{O}$, but this won't always be possible due to the forced consistency with query/responses in *KnownQueries*, $\mathcal{G}$ and $\mathcal{D}$. In such cases, the remaining queries' responses are set randomly, but in a method that ensures the randomly set responses are consistent with the responses previously set in the construction of $\widehat{\mathcal{O}}$.

We note we will not describe how to generate the sub-oracle $\widehat{\mathbf{w}}$, as it will be unnecessary.

```
**ConsolidateOrac**$(\mathcal{O}, \mathit{KnownQueries}, \mathcal{G}, \mathcal{D})$
(1)  For each $m \in \mathbb{N}$ we perform the following experiment to generate $\widehat{\mathcal{O}}$.

(2)  $\widehat{\mathbf{g}}$: Let $\mathit{Dom} = \{0,1\}^m$ and $\mathit{Rng} = \{0,1\}^{3m}$.
(3)      For each $sk \in \mathit{Dom}$ where for which there is a query/response $(< \mathbf{g}, sk >, pk)$ in
         $\mathcal{G} \cup \mathcal{D} \cup \mathit{KnownQueries}$: set $\widehat{\mathbf{g}}(sk) \leftarrow pk$, remove $pk$ from $\mathit{Rng}$, and remove $sk$ from
         $\mathit{Dom}$.
(4)      For each $sk \in \mathit{Dom}$ for which $\mathbf{g}(sk) \in \mathit{Rng}$, set $\widehat{\mathbf{g}}(sk) \leftarrow \mathbf{g}(sk)$, remove $sk$ from $\mathit{Dom}$
         and remove $pk$ from $\mathit{Rng}$.
(5)      Choose a random one-to-one function $h : \mathit{Dom} \rightarrow \mathit{Rng}$, and for each $sk \in \mathit{Dom}$ set
         $\widehat{\mathbf{g}}(sk) \leftarrow h(sk)$.

(6)  $\widehat{\mathbf{e}}$: For each $pk \in \{0,1\}^{3m}$:
(7)      Let $\mathit{Dom} = \{0,1\} \times \{0,1\}^m$ and $\mathit{Rng} = \{0,1\}^{3m}$.
(8)      For each query/response $(< \mathbf{e}, pk, b, r >, c)$ in $\mathcal{G} \cup \mathit{KnownQueries}$ set $\widehat{\mathbf{e}}(pk, b, r) \leftarrow c$
         and remove $(b, r)$ from $\mathit{Dom}$ and $c$ from $\mathit{Rng}$.
(9)      For each query/response $(< \mathbf{w}, pk, * >, (c_1, c_2, ..., c_m))$ made in $\text{E}{\scriptstyle\text{XP}}_2$:
(10)         For each $c_i$ in the response that is also in the set $\mathit{Rng}$, if there exists a $(b, r) \in \mathit{Dom}$
             such that $\mathbf{e}(pk, b, r) = c_i$, then set $\widehat{\mathbf{e}}(pk, b, r) \leftarrow c_i$.
(11)         For every remaining $c_i$ in the response:
(12)             Let $b_i = \mathbf{d}(\widehat{\mathbf{g}}^{-1}(pk), c_i)$ and select uniformly at random $r_i \in \{r | (b_i, r) \in \mathit{Dom}\}$.
(13)             Set $\widehat{\mathbf{e}}(pk, b_i, r_i) \leftarrow c_i$ and remove $(b_i, r_i)$ from $\mathit{Dom}$ and $c_i$ from $\mathit{Rng}$.
(14)
(15)     For each remaining $(b, r) \in \mathit{Dom}$ if $\mathbf{e}(pk, b, r) \in \mathit{Rng}$ then set $\widehat{\mathbf{e}}(pk, b, r) \leftarrow \mathbf{e}(pk, b, r)$,
         and remove $(b, r)$ from $\mathit{Dom}$ and $\mathbf{e}(pk, b, r)$ from $\mathit{Rng}$.
(16)     Remove $c$ from $\mathit{Rng}$ if there was a query/response $(< \mathbf{d}, \widehat{\mathbf{g}}^{-1}(pk), c >, \perp) \in$
         $\mathit{KnownQueries} \cup \mathcal{D}$ or a query/response $(< \mathbf{u}, pk, c >, \perp) \in \mathit{KnownQueries}$.
(17)     Remove $c$ from $\mathit{Rng}$ if there was a query $(< \mathbf{d}, \widehat{\mathbf{g}}^{-1}(pk), c >, \perp)$ performed in $\text{E}{\scriptstyle\text{XP}}_1$ or
         $\text{E}{\scriptstyle\text{XP}}_2$ excluding those queries that were made during a call to $\mathbf{D^O}(SK, *)$.
(18)     Choose a random one-to-one function $h : \mathit{Dom} \rightarrow \mathit{Rng}$, and for each $(b, r) \in \mathit{Dom}$ set
         $\widehat{\mathbf{e}}(pk, b, r) \leftarrow h(b, r)$.

(19) $\widehat{\mathbf{d}}$: For each $sk \in \{0,1\}^m, c \in \{0,1\}^{3m}$ and $b \in \{0,1\}$
(20)     Set $\widehat{\mathbf{d}}(sk, c) \leftarrow b$ if there exists an $r \in \{0,1\}^m$ such that $\widehat{\mathbf{e}}(\widehat{\mathbf{g}}(sk), b, r) = c$; and
         otherwise set $\widehat{\mathbf{d}}(sk, c) \leftarrow \perp$.

(21) $\widehat{\mathbf{u}}$ : For each $pk \in \{0,1\}^{3m}$ and $c \in \{0,1\}^{3m}$:
(22)     Set $\widehat{\mathbf{u}}(pk, c) \leftarrow \top$ if there exists an $sk \in \{0,1\}^m, b \in \{0,1\}$ and $r \in \{0,1\}^m$ such that
         $\widehat{\mathbf{g}}(sk) = pk$ and $\widehat{\mathbf{e}}(pk, b, r) = c$; otherwise, set $\widehat{\mathbf{u}}(pk, c) \leftarrow \perp$.
```

We have not yet made clear how the adversary will actually simulate access to such an oracle $\widehat{\mathcal{O}}$ while only making a polynomial number of queries to $\mathcal{O}$. This will be described in detail later in Section N.1, but as previously mentioned the adversary will be unable to simulate $\widehat{\mathcal{O}}$ in all cases. However, it will do a sufficiently good simulation that it is likely that the adversary will be decrypt its challenge ciphertext with the aid of an alternate decryption algorithm.

We now formalize $\widehat{\text{Exp}}_3$, the modified version of $\text{Exp}_3$.

```
$\widehat{\text{Exp}}_3(E_n^2)$
(1)    $E_n^3 = (\mathbf{Env}'_n, \mathcal{G}, \mathcal{D}, E_n^2) \leftarrow \text{Exp}_3(E_n^2)$
(2)    Let $\mathbf{Env}'_n = (\mathcal{O}', PK, SK', S')$
(3)    $\widehat{\mathcal{O}} \leftarrow \mathbf{ConsolidateOrac}(\mathcal{O}, KnownQueries, \mathcal{G}, \mathcal{D})$
(4)    Let $\widehat{\mathbf{Env}}_n = (\widehat{\mathcal{O}}, PK, SK', S')$.
(5)    Output $\widehat{E}_n^3 = (\widehat{\mathbf{Env}}_n, \mathcal{G}, \mathcal{D}, E_n^2)$
```

We stress the following key observation: the distributions on $\widehat{\mathbf{Env}}_n$ between the following two experiments is identical.

**Observation 62.** *The distributions on $\widehat{E}_n^3$ and $E_n^3$ that are produced in the following two experiments are identical when the inclusion of $\mathbf{Env}_n$ is suppressed* [15]*:*

**First Experiment**

$$\mathbf{Env}_n \leftarrow Env(n),\ E_n^1 \leftarrow \text{Exp}_1(\mathbf{Env}_n),\ E_n^2 \leftarrow \text{Exp}_2(E_n^1),\ E_n^3 = (\mathbf{Env}'_n, \mathcal{G}, \mathcal{D}, E_n^2) \leftarrow \text{Exp}_3(E_n^2),$$

*where $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ holds.*

**Second Experiment**

$$\mathbf{Env}_n \leftarrow Env(n),\ E_n^1 \leftarrow \text{Exp}_1(\mathbf{Env}_n),\ E_n^2 \leftarrow \text{Exp}_2(E_n^1),\ \widehat{E}_n^3 = (\widehat{\mathbf{Env}}_n, \mathcal{G}, \mathcal{D}, E_n^2) \leftarrow \widehat{\text{Exp}}_3(E_n^2),$$

*where $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ holds.*

*Further, because of this observation, we can meaningfully discuss the event $\mathbf{BAD}_6$ relative to the second experiment above. The observation implies that the bound given in Corollary 60 on the probability of the event $\mathbf{BAD}_6$ in the first experiment also hold for the corresponding events in the second experiment.*

For the remainder of the chapter, we will assume that Corollary 60 applies relative to the second experiment above.

As previously mentioned, the reason for introducing $\widehat{\mathcal{O}}$ is to show that it is likely that $\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ for randomly selected $M$ and $R$, and then show we can use $SK'$ to decrypt relative to it. In the next lemma we show that encryptions relative to $\mathbf{O}$ and $\widehat{\mathbf{O}}$ are likely to be the same. We note that intuitively this should be the case, as $\mathcal{O}$ and $\widehat{\mathcal{O}}$ will have identical responses to most queries, and, by design, it is highly probable that they have identical responses on queries that are commonly made during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$, for randomly chosen $M$ and $R$, as the oracles will be consistent on the set $\mathcal{E}$ which contains all of the likely query/responses that were made during an execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ for randomly chosen $M$ and $R$.

**Lemma 63.**

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n)\\ E_n^1 \leftarrow \text{Exp}_1(\mathbf{Env}_n)\\ E_n^2 \leftarrow \text{Exp}_2(E_n^1)\\ \widehat{E}_n^3 \leftarrow \widehat{\text{Exp}}_3(E_n^2), M\in\{0,1\}, R\in\{0,1\}^{n^{\rho_2}}}} [\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R) | \bigwedge_{i\in\{1,2,3,5\}} \overline{\mathbf{BAD}_i}] \geq 1 - \frac{4n^{5q+2\alpha_4+s}}{n^{\alpha_1 - 2}}$$

*Proof.* We bound the probability that there is a query made to $\mathbf{O}$ during the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ whose response differs from that of $\widehat{\mathbf{O}}$. We will bound the probability that a given query made by $\mathbf{E}$ has

[15] We note that both $\widehat{E}_n^3$ and $E_n^3$ contain $E_n^2$ which contains $E_n^1$, which contains $\mathbf{Env}_n$, and it is this variable that is excluded when we discuss the equality of the distributions. Clearly, if $\mathbf{Env}_n$ is included the distributions are not identical as on average $\widehat{\mathcal{O}}$ and $\mathcal{O}$ in $\widehat{E}_n^3$ will be far more correlated than $\mathcal{O}'$ and $\mathcal{O}$ in $E_n^3$

different responses relative to oracles $\mathbf{O}$ and $\widehat{\mathbf{O}}$. Next, we apply the union-bound to get a bound on the probability that any of the $n^q$ queries made by $\mathbf{E}$ have different responses from oracles $\mathbf{O}$ and $\widehat{\mathbf{O}}$.

The proof of this lemma is based on two premises: first, that $\mathbf{O}$ and $\widehat{\mathbf{O}}$ have the same responses for queries that are likely to be made during the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ (because $\overline{\mathbf{BAD}_2}$ holds); second, there are only a relatively small number of queries whose responses are not equivalent between $\mathbf{O}$ and $\widehat{\mathbf{O}}$, and these queries are not likely to be made. Therefore, most of this argument amounts to counting the number of queries whose responses from the two oracles are different, and showing that such queries are unlikely.

First, let's consider the number of queries whose responses differ between $\mathbf{O}$ and $\widehat{\mathbf{O}}$. In order to do so, we need to look at how $\widehat{\mathcal{O}}$ was constructed by the call **ConsolidateOrac**$(\mathcal{O}, \mathit{KnownQueries}, \mathcal{G}, \mathcal{D})$ in $\widehat{\text{Exp}_3}$.

We begin by considering queries that have differing responses between $\mathbf{g}$ and $\widehat{\mathbf{g}}$. Any query whose corresponding query/response is in $\mathcal{G}$ or $\mathcal{D}$ has the potential for differing responses $\mathbf{g}$ and $\widehat{\mathbf{g}}$, as $\widehat{\mathbf{g}}$'s responses are assigned in line 3 of **ConsolidateOrac**. We note that assigning responses $\widehat{\mathbf{g}}(sk) \leftarrow pk$ that correspond with query/responses $(<\mathbf{g}, sk>, pk) \in \mathit{KnownQueries}$ ensures consistent responses between $\mathbf{g}$ and $\widehat{\mathbf{g}}$, as all query/responses in $\mathit{KnownQueries}$ are consistent with $\mathcal{O}$.

$|\mathcal{G}| = n^q$ by Assumption 20 and inspection of $\text{Exp}_2$ shows that $|\mathcal{D}|$ is less than:

$$\begin{aligned} |KS| \cdot (\zeta(|KS|, n^q, 1) \cdot n^{2\alpha_4} \cdot n^q + n^s \cdot n^{2\alpha_4} \cdot n^q) &\leq n^q(2n^{3q+2\alpha_4} + n^{s+q+2\alpha_4}) \\ &= 2n^{4q+2\alpha_4} + n^{s+2q+2\alpha_4}, \end{aligned}$$

where the inequality follows from the fact that $\overline{\mathbf{BAD}_1}$ holds, and thus that $|KS| \leq n^q$. Clearly any query to $\widehat{\mathbf{g}}$ whose response was set on line 4 of **ConsolidateOrac** will have the same response as the corresponding query to $\mathbf{g}$. Finally, consider the return values for those queries that are defined on line 5 of **ConsolidateOrac**. The responses of $\widehat{\mathbf{g}}$ to such queries will almost surely be inconsistent with those of $\mathbf{g}$, and thus we need to bound the number of such queries. We observe that there can be at most one such query for each query/response pair in $\mathcal{G} \cup \mathcal{D}$. This is because the only way that the response to $\widehat{\mathbf{g}}(sk^*)$ is set on line 5 is if $\mathbf{g}(sk^*) = c^*$ and there exists a query/response $(<\mathbf{g}, sk'>, c^*)$ for $sk' \neq sk^*$. Therefore, there can be at most $2n^{4q+2\alpha_4} + n^{s+2q+2\alpha_4} + n^q$ queries whose responses differ between $\mathbf{g}$ and $\widehat{\mathbf{g}}$ due to line 5. Further, none of the queries is in $\mathcal{IQ}$ and therefore, since $\overline{\mathbf{BAD}_2}$ holds, the probability of any such query is less than $1/n^{\alpha_1 - 2}$. Therefore, the probability that there is a query made to $\mathbf{g}$ that has response that is different than the same query made to $\widehat{\mathbf{g}}$ is less than $\frac{2n^{4q+2\alpha_4} + n^{s+2q+2\alpha_4} + n^q}{n^{\alpha_1 - 2}}$.

Next, consider queries that have differing responses between $\mathbf{e}$ and $\widehat{\mathbf{e}}$. Any query whose corresponding query/response is in $\mathcal{G}$ has the potential for different responses from the two sub-oracles $\mathbf{e}$ and $\widehat{\mathbf{e}}$, as these values are assigned in line 8 of **ConsolidateOrac**. There are at most $n^q$ such queries as $\overline{\mathbf{BAD}_1}$ holds. The only other places in **ConsolidateOrac** where a query to $\widehat{\mathbf{e}}$ is assigned a return value that differs from $\mathbf{e}$ is on lines 13 and 18. Consider each of these case separately.

In the first case, on line 13 of **ConsolidateOrac** there can be at most $n^q$ queries to $\widehat{\mathbf{e}}$ whose responses are set to be different than the corresponding response of $\mathbf{e}$. For, if on line 13 the value of $\widehat{\mathbf{e}}(pk, b_i, r_i)$ is set to $c_i$, then for the pair $(b, r)$ for which $\mathbf{e}(pk, b, r) = c_i$, it is necessarily the case that the response to $\widehat{\mathbf{e}}(pk, b, r)$ was already assigned a value other than $c_i$ on line 8 of **ConsolidateOrac**.

In the second case, on line 18 there can be at most $2n^q$ queries to $\widehat{\mathbf{e}}$ whose responses are set to be different than those of $\mathbf{e}$, as any inconsistency in response that is formed on this line is a direct consequence of a previous inconsistency in response on either lines 8 or 13, and there are at most $2n^q$ such inconsistencies. Further, none of these queries is in $\mathcal{IQ}$ and therefore, since $\overline{\mathbf{BAD}_2}$ holds, the probability of any such query is made during the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ is less than $1/n^{\alpha_1 - 2}$. Therefore, the probability that there is a query made to $\mathbf{e}$ that has response that is different that the same query made to $\widehat{\mathbf{e}}$ is less than $\frac{4n^q}{n^{\alpha_1 - 2}}$

Finally, consider queries to $\widehat{\mathbf{d}}$ that have different responses to the same queries in $\mathbf{d}$. There are two very different ways that differing responses can occur between $\mathbf{d}$ and $\widehat{\mathbf{d}}$: because of differences between $\mathbf{g}$ and $\widehat{\mathbf{g}}$, and because of differences between $\mathbf{e}$ and $\widehat{\mathbf{e}}$. We will consider two mutually exclusive sub-cases for a query $(\widehat{\mathbf{d}}, sk, c)$ where $\widehat{\mathbf{d}}(sk, c) \neq \mathbf{d}(sk, c)$. The first is when $\mathbf{g}(sk) = \widehat{\mathbf{g}}(sk)$ and the second is when $\mathbf{g}(sk) \neq \widehat{\mathbf{g}}(sk)$. We bound the probability of each of these case to also be less than $\frac{4n^q}{n^{\alpha_1 - 2}}$, and therefore the probability of the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ making a query to $\mathbf{d}$ whose response is different than that of the equivalent query to $\widehat{\mathbf{d}}$ is less than $\frac{4n^q}{n^{\alpha_1 - 2}}$.

Before, proving the bound on the probability of different responses for corresponding queries to $\mathbf{d}$ and $\widehat{\mathbf{d}}$ we note that given these bounds, we can bound the probability that $\mathbf{E}^{\mathbf{O}}(PK, M, R) \neq \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$. The probability that any query being made to $\mathbf{O}$ during the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ has differing responses between $\mathbf{O}$ and $\widehat{\mathbf{O}}$ is less than $\max\left\{\frac{2n^{4q+2\alpha_4}+n^{s+2q+2\alpha_4}+n^q}{n^{\alpha_1-2}}, \frac{4n^q}{n^{\alpha_1-2}}\right\}$, which is equal to $\frac{2n^{4q+2\alpha_4}+n^{s+2q+2\alpha_4}+n^q}{n^{\alpha_1-2}}$ for all sufficiently large $n$. There are at most $n^q$ queries made during the execution $\mathbf{E}^{\mathbf{O}}(PK, M, R)$. Therefore, for all sufficiently large $n$ we can bound the probability that $\mathbf{E}^{\mathbf{O}}(PK, M, R) \neq \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ to be less that

$$(n^q)\left(\frac{2n^{4q+2\alpha_4}+n^{s+2q+2\alpha_4}+n^q}{n^{\alpha_1-2}}\right) \leq \frac{4n^{5q+2\alpha_4+s}}{n^{\alpha_1-2}}.$$

It remains to show the derivation of the bounds for the two mutually exclusive sub-cases for a query $(\widehat{\mathbf{d}}, sk, c)$ where $\widehat{\mathbf{d}}(sk, c) \neq \mathbf{d}(sk, c)$. First, where $\mathbf{g}(sk) = \widehat{\mathbf{g}}(sk)$ and second where $\mathbf{g}(sk) \neq \widehat{\mathbf{g}}(sk)$. These bounds are derived below.

**Case $\mathbf{g}(sk) = \widehat{\mathbf{g}}(sk)$:** Let $\mathbf{g}(sk) = pk$. The fact that $\widehat{\mathbf{d}}(sk, c) \neq \mathbf{d}(sk, c)$ implies that there exists a $b^*, r^*$ such that $\mathbf{e}(pk, b^*, r^*) = c$ but $\widehat{\mathbf{e}}(pk, b^*, r^*) \neq c$ or vice-versa. Therefore, we can use the upper-bound of $4n^q$ that was previously established on the number of different responses for equivalent queries to $\mathbf{e}$ and $\widehat{\mathbf{e}}$. Further, we know that the probability of the query $(\mathbf{d}, sk, c)$ was less than $1/n^{\alpha_1-2}$ as it could not be in $\mathcal{IQ}$ if the responses differ between $\mathbf{d}$ and $\widehat{\mathbf{d}}$ and because $\overline{\mathbf{BAD}_2}$ holds. Therefore, the probability of such a query being made is less than $\frac{4n^q}{n^{\alpha_1-2}}$.

**Case $\mathbf{g}(sk) \neq \widehat{\mathbf{g}}(sk)$:** In this case we cannot simply count on the fact that the probability of such a query $(\mathbf{d}, sk, c)$ is low, and the number of queries where $\mathbf{d}(sk, \cdot) \neq \widehat{\mathbf{d}}(sk, \cdot)$ is low, as in all of the previous cases. This is because the number of queries on which $\mathbf{d}(sk, \cdot)$ and $\widehat{\mathbf{d}}(sk, \cdot)$ have differing responses is almost surely very large (exponential in $|sk|$). This makes this case significantly more involved than the others. We will consider two mutually exclusive sub-cases: first when $\mathbf{d}(sk, c) \neq \perp$ and when $\mathbf{d}(sk, c) = \perp$. We show that that the probabilities of these two types of query/responses can be bound by $\frac{4n^q}{n^{\alpha_1-2}}$ and $\frac{1}{2n^{\alpha_1-2}}$ respectively, so the probability of this case is bound by $\frac{4n^q}{n^{\alpha_1-2}}$, the maximum of the two bounds for all sufficiently large $n$. Below we show the derivation of these two bounds.

**Sub-case: $\mathbf{d}(sk, c) \neq \perp$.** Again we will sub-divide the argument into two cases: first, we consider the possibility that $\mathbf{G}^{\mathbf{O}}(S^*)$ made a query $(< \mathbf{e}, \mathbf{g}(sk), *, * >, c)$ (remembering that $S^*$ is the seed in $\mathbf{Env}_n$). The probability of this case is less than $\frac{n^q}{n^{\alpha_1-2}}$. This is because the probability of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ making any such query $\mathbf{d}(sk, c)$ is less than $1/n^{\alpha_1-2}$, because $\overline{\mathbf{BAD}_2}$ holds and there are at most $n^q$ queries to $\mathbf{e}$ that are made during the execution of $\mathbf{G}^{\mathbf{O}}(S^*)$.
The second case is when there was no query/response $(< \mathbf{e}, \mathbf{g}(sk), *, * >, c)$ made during the execution of $\mathbf{G}^{\mathbf{O}}(S^*)$. During the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ there was no query/response $(< \mathbf{e}, \mathbf{g}(sk), *, * >, c)$ prior to the query $\mathbf{d}(sk, c)$; this is because Assumption 22 ensures that $\mathbf{E}$ never queries $\mathbf{d}(sk, \cdot)$ if it already has access to its response due to a previous query to $\mathbf{e}(\mathbf{g}(sk), \cdot, \cdot)$. Thus, since there was no query $(< \mathbf{e}, \mathbf{g}(sk), *, * >, c)$ made during $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ nor during $\mathbf{G}^{\mathbf{O}}(S^*)$ the fact that the response to the query $\mathbf{d}(sk, c)$ is not $\perp$ is very improbable.
The probability of making such a query/response is low because $\mathbf{e}(\mathbf{g}(sk), \cdot, \cdot)$ is a random one-to-one function whose range is exponentially larger than its domain ( because of the random selection of $\mathcal{O} \leftarrow \Upsilon$ in the $Env(n)$), and therefore the probability of finding a string $c$ in the co-domain of $\mathbf{e}(\mathbf{g}(sk), \cdot, \cdot)$ is very unlikely.
It is easy to observe that if our experiment was simply

$$\begin{gathered}\mathbf{Env}_n \leftarrow Env(n) \\ E_n^1 \leftarrow \text{EXP}_1(\mathbf{Env}_n) \\ E_n^2 \leftarrow \text{EXP}_2(E_n^1) \\ \widehat{E}_n^3 \leftarrow \widehat{\text{EXP}}_3(E_n^2), M \in \{0,1\}, R \in \{0,1\}^{n^{\rho_2}}\end{gathered}$$

the probability that the execution of $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ would make such a query/response $\mathbf{d}(sk, c) \neq \perp$ would no more than $\frac{n^{\alpha_0+1}-2n^q}{n^{3\alpha_0}-2n^q}$: we subtract $n^q$ to allow for the information that is revealed about the function $\mathbf{e}(\mathbf{g}(sk), \cdot, \cdot)$ during $\mathbf{G}^{\mathbf{O}}(S^*)$ and we subtract another $n^q$ for further information that is revealed during any queries made in $\mathbf{E}^{\mathbf{O}}(PK, M, R)$ prior to the query $(\mathbf{d}, sk, c)$). However, our experiment is conditioned on $\wedge_{i \in \{1,2,3,5\}} \overline{\mathbf{BAD}_i}$ holding in $\text{EXP}_1$ and $\text{EXP}_2$, so this effects the probability. By dividing by a lower-bound on probability of these event we get an upper-bound on the probability of the event. This probability is less than

$$\frac{\left(\frac{n^{\alpha_0+1}-n^q}{n^{3\alpha_0}-n^q}\right)}{\Pr[\wedge_{i\in\{1,2,3,5\}}\overline{\mathbf{BAD}_i}]} \leq \frac{2n^{\alpha_0+1}-2n^q}{n^{3\alpha_0}-n^q},$$

where the inequality holds for all sufficiently large $n$, as for all sufficiently large $n$ its definitely the case that $\Pr[\wedge_{i\in\{1,2,3,5\}}\overline{\mathbf{BAD}_i}] \geq \Pr[\wedge_{1\leq i\leq 5}\overline{\mathbf{BAD}_i}] \geq \frac{1}{2}$ for all sufficiently large $n$, where the last inequality follows from Lemmas 54 and 55.

**Sub-case: $\mathbf{d}(sk, c) = \perp$.** This is very much a dual to the previous sub-case. We will consider two cases: first we consider the case that there is a query/response $(<\widehat{\mathbf{e}}, *, *>, c) \in \mathcal{G} \cup \mathcal{D}$, and next we consider the alternative case where no such query/response exists.

In the case that there is a query/response $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, *>, c)$ in the set $\mathcal{G}$, then we know the probability of the query $(\mathbf{d}, sk, c)$ is at most $\frac{n^q}{n^{\alpha_1-2}}$, since the probability of the query $(\mathbf{d}, sk, c)$ is at most $n^{\alpha_1-2}$ because $\mathbf{BAD}_2$ holds, and since $|\mathcal{G}| = n^q$.

In contrast, consider the case that there is no query/response $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, *>, c) \in \mathcal{G}$. In this case, let $b'$ and $r'$ be the strings for which $\widehat{\mathbf{e}}(\widehat{\mathbf{g}}(sk), b', r') = c$. There was no query $(\mathbf{e}, \widehat{\mathbf{g}}, b', r')$ in either $\textsc{Exp}_1$ or during $\textsc{Exp}_2$, as otherwise such a query would be in *KnownQueries* which would imply that $\mathbf{d}(sk, c) = \widehat{\mathbf{d}}(sk, c)$ contradicting the current case. Therefore, such a query is unlikely, as it requires the adversary to choose a string $c$ that is in the range of the one-to-one length tripling function $\widehat{\mathbf{e}}(\widehat{\mathbf{g}}, \cdot, \cdot)$. We note that this function was not chosen uniformly at random, as it is constructed by a call to **ConsolidateOrac**$(\mathcal{O}, \mathit{KnownQueries}, \mathcal{G}, \mathcal{D})$. However, by noting that there was no query/response $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, *>, c)$ in the set $\mathcal{G} \cup \mathit{KnownQueries}$ we know that line 8 of **ConsolidateOrac** did not set the value of $\widehat{\mathbf{e}}(pk, b', r')$. Similarly, there could not have been any queries to $\mathbf{w}(\widehat{\mathbf{g}}(sk), \cdot)$ in $\textsc{Exp}_2$, for if there were the fact that $\overline{4}$ holds would imply that $\mathbf{g}(sk) = \widehat{\mathbf{g}}(sk)$, contradicting the conditions for the current case. This implies that line 9 through 13 of **ConsolidateOrac** did not have an effect on the value of $\widehat{\mathbf{e}}(\widehat{\mathbf{g}}(sk), b^*, r^*)$. Therefore, the value of $\widehat{\mathbf{e}}(\widehat{\mathbf{g}}(sk), b^*, r^*)$ was set on either line 15 or line 18. We will consider each possibility separately. If the value was set on line 15, then the probability of finding the string $c$ is less than $\frac{n^{\alpha_1-2}-|\mathit{KnownQueries}|}{3n^{\alpha_1-2}-|\mathit{KnownQueries}|} \leq 1/2n^{\alpha_1-2}$. We note this, by seeing that the $\mathbf{e}(\widehat{\mathbf{g}}(sk), \cdot, \cdot)$ was randomly chosen by the random process $\Upsilon$. However, in the worst-case we can assume that $|\mathit{KnownQueries}|$ input-output pairings of $\mathbf{e}(\widehat{\mathbf{g}}(sk), \cdot, \cdot)$ have been revealed in the experiment.

Alternatively, if the value was set on line 18, then the probability of finding such a string $c$ is less than $1/2n^{\alpha_1-2}$. This can be seen by noting that during the execution of line 18 it will be the case that $|\mathit{Dom}| \geq 1$ and $|\mathit{Rng}| \geq 2n^{\alpha_1-2}$. Therefore, since both mutually exclusive cases bound the probability of a query/response $\mathbf{d}(sk, c) = \perp$ when $\widehat{\mathbf{d}}(sk, c) \neq \perp$, we can bound the probability of this sub-case to be less than $1/2n^{\alpha_1-2}$.

□

**Corollary 64.**

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n)\\ E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)\\ E_n^2 \leftarrow \textsc{Exp}_2(E_n^1)\\ \widehat{E}_n^3 \leftarrow \widehat{\textsc{Exp}}_3(E_n^2), M\in\{0,1\}, R\in\{0,1\}^{n^{\rho_2}}}}[\mathbf{E}^{\mathbf{O}}(PK, M, R) = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R) | \bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}] \geq 1 - \frac{n^{3\alpha_4}}{n^{\alpha_1-2}}$$

*Proof.* Follows from the bounds in previous lemma (63) and the fact that for all sufficiently large $n$ we can bound $\Pr[\overline{\mathbf{BAD}_2} | \bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}]$ to be at least $1 - 3n^{2q}/2^n$, which follows directly from Lemma 55. Using these bounds we note that:

$$\begin{array}{c}\Pr[\mathbf{E}^{\mathbf{O}}(PK,M,R)=\mathbf{E}^{\widehat{\mathbf{O}}}(PK,M,R)|\bigwedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}]\\ \geq\\ \Pr[\mathbf{E}^{\mathbf{O}}(PK,M,R)=\mathbf{E}^{\widehat{\mathbf{O}}}(PK,M,R)|\bigwedge_{i\in\{1,2,3,5\}}\overline{\mathbf{BAD}_i}]\cdot\Pr[\overline{\mathbf{BAD}_2}|\bigwedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}]\\ \geq(1-\frac{4n^{5q+2\alpha_4+s}}{n^{\alpha_1-2}})(1-3n^{2q}/2^n)\geq 1-\frac{n^{3\alpha_4}}{n^{\alpha_1-2}}\end{array}$$

where the last inequality holds for all sufficiently large $n$. □

## M $A_1$: The First Part of the CCA#1 Adversary

The first part of the adversary, $A_1$, can now be properly described. Given the definition of CCA#1 security for a PKEP, the adversary $A_1$ will be given a random oracle $\mathcal{O}\leftarrow\Upsilon$; a public-key $PK$, corresponding to the secret-key $SK$, generated by a call to $\mathbf{G}^{\mathbf{O}}(S^*)$ for a randomly chosen $S^*\in_R\{0,1\}^n$; and access to the decryption oracle $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK,\cdot)$. $A_1$ will then consecutively execute $\textsc{Exp}_1,\textsc{Exp}_2$ and $\widehat{\textsc{Exp}}_3$. It will simulate $\mathbf{Env}_n$, by using the oracle $\mathcal{O}$, the public-key $PK$ and the decryption oracle $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK,\cdot)$. It will then be the case that with very high probability that $\bigwedge_{i\in\{1,3,5,6\}}\overline{\mathbf{BAD}_i}$ holds. This will imply that if $\mathbf{E}^{\mathbf{O}}(PK,M,R)=C$ is the challenge ciphertext that will be given to $A_2$, the second part of the adversary $A_2$, it will necessarily be the case that $\mathbf{E}^{\widehat{\mathbf{O}}}(PK,M,R)=C$ with high probability. What remains to show is that $A_2$ can somehow decrypt $C$, in those cases when $\mathbf{E}^{\widehat{\mathbf{O}}}(PK,M,R)=C$. This is the focus of the next section.

## N $A_2$: The Second Part of the CCA#1 Adversary

### N.1 How the Adversary Simulates $\widehat{\mathcal{O}}$

We would like the adversary to be able to execute $\mathbf{D}^{\widehat{\mathbf{O}}}(SK',\cdot)$. Unfortunately, as has been suggested several times, the perfect simulation of $\widehat{\mathbf{O}}$ cannot be done without an exponential number of queries to $\mathcal{O}$, which is not permitted. Therefore, we will have the adversary perform an imperfect simulation of $\widehat{\mathcal{O}}$, and show that the simulation is correct for a large fraction of the possible queries to $\widehat{\mathcal{O}}$. Further, this simulation can be done without the need to make a large number of queries to $\mathcal{O}$. We will isolate those queries to $\widehat{\mathcal{O}}$ whose responses are improperly simulated, and then construct an alternate decryption algorithm, $\widehat{\mathbf{D}}$, that does a good job of decrypting relative to the oracle $\widehat{\mathbf{O}}$, and that is unlikely to make and of the queries whose responses are simulated improperly.

Below we describe how $A_2$, the second part of the adversary, simulates the response of an oracle query to $\widehat{\mathbf{O}}$. We stress that it is assumed that $A_1$, the first part of the adversary, has already performed $\textsc{Exp}_1$, $\textsc{Exp}_2$ and $\widehat{\textsc{Exp}}_3$, and further that $\bigwedge_{i\in\{1,3,5\}}$ held during those experiments, and we note that this is the likely case, as was previously discussed in Section M.

Because the algorithm $\widehat{\mathbf{D}}$, which is the only algorithm that $A_2$ will ever execute, only ever queries $\widehat{\mathbf{g}}$, $\widehat{\mathbf{d}}$ and $\widehat{\mathbf{u}}$, we will only explain how to simulate these queries. The simulation of the oracle $\widehat{\mathcal{O}}$ that the adversary performs is based on the following idea: if the adversary wants to simulate the response to a query to $\widehat{\mathcal{O}}$ and the query can be answered with knowledge from query/responses that are stored in $\mathcal{G}$, $\mathit{KnownQueries}$ and $\mathcal{D}$, then the adversary responds using that knowledge without making any query to $\mathcal{O}$; otherwise, the adversary assumes that the responses of $\mathcal{O}$ and $\widehat{\mathcal{O}}$ are equivalent, and queries $\mathcal{O}$ and uses the response as the corresponding response to the query to $\widehat{\mathcal{O}}$. Specifically, the simulation works as follows:

**On the query** $(\widehat{\mathbf{g}},sk)$**:** if there is a query/response $(<\mathbf{g},sk>,c)\in\mathcal{G}\cup\mathit{KnownQueries}\cup\mathcal{D}$, then the simulation responds with $c$, otherwise it responds with $\mathbf{g}(sk)$.

**On the query** $(\widehat{\mathbf{d}},sk,c)$**:** if there has been a query/response $(<\mathbf{d},sk,c>,b)\in\mathcal{G}\cup\mathit{KnownQueries}\cup\mathcal{D}$, then respond with $b$. Alternatively, if $(<\mathbf{g},sk>,pk)$ and $(<\mathbf{e},pk,b,*>,c)$ are in $\mathcal{G}\cup\mathit{KnownQueries}\cup\mathcal{D}$

then respond with $b$. Otherwise, respond with $\mathbf{d}(sk, c)$.

**On the query** $(\widehat{\mathbf{u}}, pk, c)$**:** if there has been a pair of query/responses $(<\mathbf{g}, sk>, pk), (<\mathbf{d}, sk, c>, b) \in \mathcal{G} \cup KnownQueries \cup \mathcal{D}$, then if $b \in \{0, 1\}$ respond with $\top$ and otherwise $\bot$. Alternatively, if $(<\mathbf{g}, sk>, pk)$ *and* $(<\mathbf{e}, pk, b, *>, c)$ are in $\mathcal{G} \cup KnownQueries \cup \mathcal{D}$ then respond with $\top$. Otherwise, respond with$\mathbf{u}(pk, c)$.

## N.2 How the Adversary's Simulation can go Wrong

As was previously mentioned, this simulation is not perfect; below we list ways in which the simulation can go wrong. It's easy to observe from the description of the simulation in the previous section, and the description of **ConsolidateOrac** that the list enumerates all of the ways in which the simulation can go wrong.

**On a query** $(\widehat{\mathbf{g}}, sk)$**,** the simulated response could be incorrect if there is a query $sk$ where the value $\widehat{\mathbf{g}}(sk)$ was randomly defined on line 5 of **ConsolidateOrac**.

**On a query** $(\widehat{\mathbf{d}}, sk, c)$**,** the simulated response could be incorrect if any of the following hold:

1. there is a query/response $(<\mathbf{g}, sk>, pk) \in \mathcal{G} \cup \mathcal{D}$ for some $pk$ and $(<\mathbf{g}, sk>, pk) \notin KnownQueries$ and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ in the set $KnownQueries \cup \mathcal{D} \cup \mathcal{G}$.
2. $\widehat{\mathbf{g}}(sk)$ was defined on line 5 of **ConsolidateOrac** , and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ in the set $KnownQueries \cup \mathcal{D} \cup \mathcal{G}$.
3. $(<\mathbf{g}, sk>, pk) \in KnownQueries$ for some $pk$, but $(<\mathbf{e}, pk, *, *>, c)$ was defined on line 18 of **ConsolidateOrac**, and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ in the set $KnownQueries \cup \mathcal{D} \cup \mathcal{G}$.

**On a query** $(\widehat{\mathbf{u}}, pk, c)$**,** the simulated response could be incorrect if:

1. there is a query/response $(<\mathbf{g}, sk>, pk) \in \mathcal{G} \cup \mathcal{D}$ for some $pk$ and there is no query/response $(<\mathbf{g}, sk>, pk)$ in $KnownQueries$ and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ or $(<\mathbf{u}, pk, c>, *)$ in the set $KnownQueries \cup \mathcal{D} \cup \mathcal{G}$.
2. $\widehat{\mathbf{g}}(sk) = pk$ was defined on line 5 of **ConsolidateOrac** and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ or $(<\mathbf{u}, pk, c>, *)$ in the set $KnownQueries \cup \mathcal{D} \cup \mathcal{G}$.
3. $(<\mathbf{g}, sk>, pk) \in KnownQueries$ for some $sk$, but $(<\mathbf{e}, pk, *, *>, c)$ was defined on line 18 of **ConsolidateOrac**, and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ or $(<\mathbf{u}, pk, c>, *)$ in the set $KnownQueries \cup \mathcal{D} \cup \mathcal{G}$.

If the adversary could simply run $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C)$ and our simulation was correct with high probability, it could break the alleged CCA#1 security of $(\mathbf{G}, \mathbf{E}, \mathbf{D})$. Unfortunately things are not so simple. Later in the chapter, we will show that it is unlikely that an execution of $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C)$ will make any queries to $\widehat{\mathbf{g}}$ where the simulation could go wrong. This is because all of the queries to $\widehat{\mathbf{g}}$ for which the simulated response can be wrong are extremely unlikely to be made during an execution of $\mathbf{D}$ in the first place. Unfortunately, this is not the case with queries to $\widehat{\mathbf{d}}$.

In Section N.2 we demonstrated three ways in which the simulation could go wrong when responding to queries to $\widehat{\mathbf{d}}$. It is unlikely that a query will be made during the execution of $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C)$ that corresponds to reasons 2 or 3, as such queries are effectively hard to find. The problem is that queries that correspond to reason 1 are not necessarily unlikely. This is basically the problem that results from the fact that we cannot guarantee that $BKS^* = \emptyset$ at the end of the second experiment, and so there are public-keys $pk$ that are embedded into $PK$ for which the corresponding secret key $sk = \mathbf{g}^{-1}(pk)$ was never retrieved. This means that the constructed oracle $\widehat{\mathbf{O}}$ is not consistent with $\mathbf{g}(sk) = pk$, and therefore there is some value $sk'$ for which $\widehat{\mathbf{g}}(sk') = pk$, and this value $sk'$ may have been embedded into $SK'$. This causes problems as this implies that is might be quite likely that $\widehat{\mathbf{d}}(sk', \cdot)$ is queried during an execution of $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C)$, but the simulated response provided by the corresponding query to $\mathbf{d}(sk', \cdot)$ is almost surely incorrect.

Fortunately, it is exactly these problematic queries that we have shown are in some sense unessential. The reason is the following: consider a query $(\widehat{\mathbf{d}}, sk', c)$ that is made during $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C)$, where $C = \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ for randomly chosen $M$ and $R$, then you would expect that $c$ was found by making a specific query $(\widehat{\mathbf{e}}, pk, b, r)$ during the execution of $\mathbf{E}$, and that $\mathbf{D}$ expects to retrieve $b$. However, during the hybridization in $\textsc{Exp}_2$ we showed that if we replaced the response to the query $(\widehat{\mathbf{e}}, pk, b, r)$ with the response $\widehat{\mathbf{e}}(pk, b', r')$ for randomly chosen $b'$ and $r'$, then the resulting ciphertext was likely to decrypt properly. Therefore, the response to the query $(\widehat{\mathbf{d}}, sk', c)$ cannot be that useful to the outcome of $\mathbf{D}$, we may as well return a random bit.

Based on the above intuition we will demonstrate that there exists an alternative decryption algorithm that with high probability only makes oracle queries that the adversary is likely to simulate the response to correctly. In order to do this, the constructed decryption algorithm will never makes queries $(\widehat{\mathbf{d}}, sk, *)$ in those cases where $\widehat{\mathbf{g}}(sk)$ is in the set $BKS^*$ retrieved by $\textsc{Exp}_2$. We will call our alternate decryption algorithm $\widehat{\mathbf{D}}$.

We remind the reader that each query $(\widehat{\mathbf{e}}, pk, b, r)$ where $pk \in BKS^*$ and $(\widehat{\mathbf{e}}, pk, b, r) \notin \mathcal{IQ}$ that is made during an execution of $RandCipher^{\widehat{\mathbf{O}}}(PK, *, *, BKS^*, \mathcal{IQ})$ is responded to with a replacement response $\widehat{\mathbf{e}}(pk, b', r')$ for randomly chosen $b'$ and $r'$. Thus, if $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', \cdot)$ is able to properly decrypt the ciphertexts generated by executions of $RandCipher^{\widehat{\mathbf{O}}}(PK, *, *, BKS^*, \mathcal{IQ})$, then the values to which any of the replacement responses generated by $RandCipher$ are decrypted during $\mathbf{D}^{\widehat{\mathbf{O}}}(SK', \cdot)$ should, probabilistically, not effect the outcome very much, as $\mathbf{D}$ could simply simulate decrypting the replacement responses by flipping a coin. Note that because the replacement responses represent encryptions of random-bits, this should be an effective simulation. We will show that this is the case, and construct a new algorithm $\overline{\mathbf{D}}$ that simply flips a coin instead of querying $\widehat{\mathbf{d}}(sk, \cdot)$ to find the response to the decryption replacement responses.

In order for the algorithm $\overline{\mathbf{D}}$ to distinguish between a ciphertext produced by an execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ versus one produced by an execution of $RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ})$, it would need to be able to distinguish between responses of replacement queries in $RandCipher$ and the actual query responses in $\mathbf{E}$. Because $\widehat{\mathbf{O}}$ was chosen mostly randomly and because $RandCipher$ does not replace likely queries (which are contained in the set $\mathcal{IQ}$), it is the case that there is little chance that $\overline{\mathbf{D}}$ could distinguish these cases. Therefore, we are able to show that $\overline{\mathbf{D}}$ does well at decrypting ciphertexts generated by $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$. In the next section we formalize the intuitive argument that was just presented.

## N.3 Decrypting With Limited Access to the Sub-oracle $\widehat{\mathbf{d}}$

Our goal is to produce a modified version of $\mathbf{D}$ named $\widehat{\mathbf{D}}$ that has the property that an execution of $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}}, \widehat{\mathbf{d}}}(SK', C)$ does not need to make any queries of the form form $\widehat{\mathbf{d}}(sk, *)$ in the case where $\widehat{\mathbf{g}}(sk) \in BKS$ (i.e. the queries whose responses are reliably simulatable by the adversary)

In this section we introduce an intermediate decryption algorithm $\overline{\mathbf{D}}$, that decrypts ciphertexts generated by $RandCipher$ in our experiment. It will takes as an input, in addition to a ciphertext generated by $RandCipher$ and a secret-key, the set of all replacement query/responses that were made during said execution of $RandCipher$. $\overline{\mathbf{D}}$ will then simulate an execution $\mathbf{D}$. If during the simulation of $\mathbf{D}$ a query $(\widehat{\mathbf{d}}, sk, c)$ is made for which the query/response $(< \widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, * >, c)$ is in the set of replacement query/responses, then a random bit is selected as the response and the oracle is not queries. We show that it is likely that ciphertexts produced by $RandCipher$ decrypt properly when $\overline{\mathbf{D}}$ is used to decrypt. Next, we show there is a simple modification to $\overline{\mathbf{D}}$ that produces the final decryption algorithm $\widehat{\mathbf{D}}$. We then show that $\widehat{\mathbf{D}}$ does a good job of decrypting ciphertexts produced by $RandCipher$ and it does not need as an input the extra set describing replacement responses that was required by $\overline{\mathbf{D}}$. Further, it does not query $\mathbf{d}(sk, \cdot)$ in those cases when $\mathbf{g}(sk) \in BKS$ and thus where the simulation of the oracle is likely to be incorrect. Finally, we show that $\widehat{\mathbf{D}}$ not only does a good job of decrypting ciphertexts produced by $RandCipher$, but it also does well decrypting ciphertexts produced by $\mathbf{E}$.

We will only show that the intermediate decryption algorithm $\overline{\mathbf{D}}$ does an acceptable job at properly decrypting ciphertexts produced by $RandCipher$ in the highly probable case that the randomly selected replacement query/responses that $RandCipher$ produces are not query/responses that have been previously made during $\mathbf{Env}_n$,$\textsc{Exp}_1$, $\textsc{Exp}_2$ or $\widehat{\textsc{Exp}}_3$ nor do they share the same random bits that were used to make any query/responses that were made during $\mathbf{Env}_n$,$\textsc{Exp}_1$, $\textsc{Exp}_2$ or $\widehat{\textsc{Exp}}_3$. We define this precisely below.

**Definition 65.** *For any $n$, given an experiment $\mathbf{Env}_n \leftarrow Env(n)$, $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$, $E_n^2 \leftarrow \textsc{Exp}_2(E_n^1)$, $\widehat{E}_n^3 \leftarrow \widehat{\textsc{Exp}}_3(E_n^2)$ where $\bigwedge_{i \in \{1,3,5\}} \overline{\mathbf{BAD}_i}$ holds, then given an experiment where $M \in \{0,1\}$ and $R \in$*

$\{0,1\}^{n^{\rho_2}}$ *are chosen uniformly at random and* $C \leftarrow \mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ})$ *is randomly computed we denote by* $\mathbf{BAD}_7$ *the event that during the execution of RandCipher a* replacement query/response $(< \widehat{\mathbf{e}}, pk_i, b_i, r_i >, c_i)$ *was made such that there is a query* $(\mathbf{e}, pk_i, *, r_i) \in \mathit{KnownQueries} \cup \mathcal{G}$ *or a query* $(\widehat{\mathbf{e}}, pk_i, *, r_i)$ *has previously been made during the execution of RandCipher.*

The point of this definition is the following: consider an execution of $\mathit{RandCipher}(PK, M, R, BKS, \mathcal{IQ})$ in which $\overline{\mathbf{BAD}_7}$ holds. Suppose the adversary is given a replacement response $c$ that was generated by the execution, then given all the prior knowledge that the adversary has about $\widehat{\mathbf{O}}$ the probability that $c$ represent an encryption under $\widehat{\mathbf{e}}$ of the bit 1 (or 0) is exactly 1/2.

**Claim 66.** *For all sufficiently large $n$ and every experiment:* $\mathbf{Env}_n \leftarrow \mathit{Env}(n)$, $E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n)$, $E_n^2 \leftarrow \textsc{Exp}_2(E_n^1)$, $\widehat{E}_n^3 \leftarrow \widehat{\textsc{Exp}}_3(E_n^2)$ *where* $\bigwedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}$ *holds:*

$$\Pr_{\substack{M\in\{0,1\}, R\in\{0,1\}^{n^{\rho_2}} \\ \mathit{RandCipher}(PK,M,R,BKS^*,\mathcal{IQ})}} [\mathbf{BAD}_7] \leq \frac{2n^{3\alpha_1}}{n^{\alpha_0}}.$$

Intuitively, the claim is true because there have not been that many queries made to the sub-oracle $\mathbf{e}$, and therefore the probability that a randomly chosen replacement query to the sub-oracle will match a previous query is quite low. The proof formalizes this intuition.

*Proof.* Because $\overline{\mathbf{BAD}_1}$ holds, $|BKS^*| \leq n^q$, and therefore there were at most $n^q$ different public-keys, $pk_i$, that were used in the different *replacement queries* made during the execution of *RandCipher*. For each such $pk_i$ in $BKS^*$ there are $2n^q$ replacement queries of the form $(\mathbf{e}, pk_i, *, *)$ that are made during $\mathit{RandCipher}(PK, M, R, BKS^*, \mathcal{IQ})$, and the random bits used in each of these queries were chosen independently of the other such queries. Therefore, we can bound the probability of a single *replacement query* satisfying the conditions in the definition of $\mathbf{BAD}_7$, and then use the union bound to bound the probability that *RandCipher* makes any such replacement queries.

We consider an arbitrary replacement query $(\mathbf{e}, pk_i, b_j, r_j)$ that is made during *RandCipher*. Since $pk_i \in BKS^*$, we know that $|pk_i| \geq 3\alpha_0 \log n$ (we remember that $BKS^* \subseteq KS$, and by the construction of $KS$ in $\textsc{Exp}_1$ this inequality is guaranteed). Further, by inspection of $\textsc{Exp}_1$ and $\textsc{Exp}_2$ we can bound the number of queries of the form $(\mathbf{e}, pk_i, *, *)$ and $(\mathbf{d}, sk_i, c)$: there are at most $v_1 = n^q$ such queries in $\mathbf{Env}_n$; at most $v_2 = n^{2\alpha_1} \cdot n^q$ such queries in $\textsc{Exp}_1$; at most $v_3 = n^q \cdot \left(2n^{2q} \cdot \left((n^{2\alpha_4} \cdot 2n^q) + n^s \cdot (n^{2\alpha_4} \cdot 2n^q)\right)\right)$ such queries in $\textsc{Exp}_2$; and finally at most $v_4 = 2n^q$ such queries in *RandCipher*. Let $v = \sum_{k=1}^{4} v_k$. The probability that the randomly chosen $r_j$ would match the value $r$ for any of these $v$ queries of the form $(\mathbf{e}, pk_i, *, r)$ is at most $v/n^{\alpha_0}$.

By applying the union bound, we conclude that the probability of $\mathbf{BAD}_7$ is no more than $(2n^q v)/n^{\alpha_0}$. Simple algebra and the relations on $q, \alpha_1, \alpha_4$ and $s$ (Section I.2), show that $(2n^q v) \leq 2n^{3\alpha_1}$ for all sufficiently large $n$. □

**Lemma 67.** *For all sufficiently large $n$, and all experiments*

$$\mathbf{Env}_n \leftarrow \mathit{Env}(n),\ E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n),\ E_n^2 \leftarrow \textsc{Exp}_2(E_n^1),\ \widehat{E}_n^3 \leftarrow \widehat{\textsc{Exp}}_3(E_n^2),$$

*where* $\bigwedge_{i\in\{1,3,5,6\}} \overline{\mathbf{BAD}_i}$ *holds:*

$$\Pr_{\substack{M\in\{0,1\}, R\in\{0,1\}^{n^{\rho_2}} \\ C\leftarrow \mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK,M,R,BKS^*,\mathcal{IQ})}} \left[\mathbf{D}^{\widehat{\mathbf{O}}}(SK', C) = M\right] \geq 1 - \frac{3n^{2q}}{n^{\alpha_6}}.$$

*Proof.* This follows directly from Observation 62 and Corollary 61. □

**Corollary 68.** *For all sufficiently large $n$, and all experiments*

$$\mathbf{Env}_n \leftarrow \mathit{Env}(n),\ E_n^1 \leftarrow \textsc{Exp}_1(\mathbf{Env}_n),\ E_n^2 \leftarrow \textsc{Exp}_2(E_n^1),\ \widehat{E}_n^3 \leftarrow \widehat{\textsc{Exp}}_3(E_n^2),$$

*where* $\bigwedge_{i\in\{1,3,5,6\}} \overline{\mathbf{BAD}_i}$ *holds:*

$$\Pr_{\substack{M\in\{0,1\},R\in\{0,1\}^{n^{\rho_2}}\\ C\leftarrow RandCipher^{\mathbf{O}'}(PK,M,R,BKS^*,\mathcal{IQ})}}\left[\mathbf{D}^{\widehat{\mathbf{O}}}(SK',C)=M\,\middle|\,\overline{\mathbf{BAD}_7}\right]\geq 1-\frac{4n^{2q}}{n^{\alpha_6}}.$$

*Proof.* A bound of $1-\frac{3n^{2q}}{n^{\alpha_6}}-\frac{2n^{3\alpha_1}}{n^{\alpha_0}}$ follows directly from an application of the bounds on $\overline{\mathbf{BAD}_7}$ established in Claim 66 to the bound on proper decryptions from Lemma 67. For all sufficiently large $n$, $1-\frac{3n^{2q}}{n^{\alpha_6}}-\frac{2n^{3\alpha_1}}{n^{\alpha_0}}\geq 1-\frac{4n^{2q}}{n^{\alpha_6}}$.

□

## N.4 The Intermediate Decryption Algorithm: $\overline{\mathbf{D}}$

We now formally define the modified decryption algorithm $\overline{\mathbf{D}}$.

```
$\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK,C,\mathcal{Q}_C)$
Simulate the execution of $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK,C)$
    On oracle query $(\widehat{\mathbf{g}},sk)$ reply with $\widehat{\mathbf{g}}(sk)$
    On oracle query $(\widehat{\mathbf{d}},sk,c)$
        If $(<\widehat{\mathbf{e}},\widehat{\mathbf{g}}(sk),*,*>,c)\in\mathcal{Q}_C$ reply with $b'_c\in_R\{0,1\}$
        otherwise reply with $\widehat{\mathbf{d}}(sk,c)$
Output the result of simulation
```

It takes as input a set $\mathcal{Q}_C$ that is intended to contain the set of replacement query/responses that are made by the execution of *RandCipher* that generated the ciphertext $C$.

We would like to show that the algorithm $\overline{\mathbf{D}}$ does well at properly decrypting ciphertexts generated by *RandCipher*. However, this is not sufficient, as the algorithm will be interacting with the oracle $\widehat{\mathcal{O}}$, and as previously mentioned, the adversary simulates responses to this oracle imperfectly. Therefore, we need to show that it not only does a good job at decrypting, but that its execution can be properly simulated in those cases.

**Lemma 69.** *For all sufficiently large n:*

$$\Pr_{\substack{\mathbf{Env}_n\leftarrow Env(n)\\ E_n^1\leftarrow \mathrm{EXP}_1(\mathbf{Env}_n)\\ E_n^2\leftarrow \mathrm{EXP}_2(E_n^1)\\ \widehat{E}_n^3\leftarrow \widehat{\mathrm{EXP}}_3(E_n^2),M\in\{0,1\},R\in\{0,1\}^{n^{\rho_2}}\\ C\leftarrow RandCipher^{\widehat{\mathbf{O}}}(PK,M,R,BKS,\mathcal{IQ})}}[\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK',C,\mathcal{Q}_C)=M \textit{ and is properly simulated}|\bigwedge_{i\in\{1,3,5,7\}}\overline{\mathbf{BAD}_i}]\geq 1-\frac{6n^{2q}}{n^{\alpha_6}},$$

*where* $\mathcal{Q}_C$ *denotes the set of* replacement queries/responses *made by RandCipher.*

*Proof.* We will first show that it is likely that $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK',C,\mathcal{Q}_C)=M$, and next show that is also likely that all of the responses to the oracle queries made during the execution of $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK',C,\mathcal{Q}_C)=M$ are properly simulated. Combining these two bounds will prove the claim.

As a thought experiment consider the scenario where before the execution of $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK',C,\mathcal{Q}_C)$ for every $(<\widehat{\mathbf{e}},pk,b_c,r_c>,c)\in\mathcal{Q}_C$ a bit $b'_c\in_R\{0,1\}$ is chosen uniformly at random as the intended simulated response to the query $(\widehat{\mathbf{d}},sk,c)$, should such a query be made during the execution of $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK',C)$. Clearly, this does not modify the experiment, as we are simply modifying it so that these random choices are made before the execution of $\overline{\mathbf{D}}$ as opposed to during the execution. Next, consider an oracle $\mathcal{O}'$ that differs from the oracle $\mathcal{O}$, contained in $\mathbf{Env}_n$, in that the response to each query $(\widehat{\mathbf{e}},\widehat{pk},b_c,r_c)\in\mathcal{Q}_C$ is swapped with the response to $(\widehat{pk}, 1-b_c, r_c)$ in the case that $b_c \neq b'_c$. There are further differences in $\mathbf{d}'$, $\mathbf{w}'$ and $\mathbf{u}'$ that are necessary to make them consistent with such swapping [16], but the changes are the ones that are intuitively necessary to make the oracle consistent with the proposed changes to $\widehat{\mathbf{e}}$, and so we will not discuss them further. Because we have conditioned on $\overline{\mathbf{BAD_7}}$[17], if $Env(n)$ had selected oracle $\mathcal{O}'$ instead of $\mathcal{O}$, then the original experiment would have resulted in the exact same outcome (modulo the fact that the modified oracle was chosen), as no oracle responses in the experiment would have different. Also because $\overline{\mathbf{BAD_7}}$ holds, from the adversary's point of view it is as likely that $\mathcal{O}'$ was chosen in $Env(n)$ as it is that $\mathcal{O}$ was chosen.

The fact that the adversary cannot distinguish between the two scenarios is observed by noting that in order for the two oracles, $\mathcal{O}$ and $\mathcal{O}'$, to have differing responses to a query $(\mathbf{e}, pk, b, r)$ it is a necessary condition that both $(\mathbf{e}, pk, b, r)$ and $(\mathbf{e}, pk, 1-b, r)$ were never queried during $\textsc{Exp}_1, \textsc{Exp}_2$ and $\widehat{\textsc{Exp}}_3$ in the original experiment. Further, since $\overline{\mathbf{BAD_1}} \wedge \overline{\mathbf{BAD_5}}$ holds there are no queries in the experiment that had surprising responses, and therefore we know that the queries $(\mathbf{d}, \mathbf{g}^{-1}(pk), \mathbf{e}(pk, *, r))$ and $(\mathbf{u}, pk, \mathbf{e}(pk, *, r))$ have not been made. Also because of the design of the experiment, it's the case that $pk \in BKS^*$, and for any string $pk \in BKS^*$ there have been no queries of the form $(\mathbf{w}, pk, *)$ made in the experiment.

Because of the indistinguishability previously mentioned, we imagine a slightly modified version of the originally performed experiment. In this modified version the oracle $\mathcal{O}'$ was chosen during $Env(n)$ instead of $\mathcal{O}$, and $\widehat{\mathcal{O}}'$ is output by $\widehat{\textsc{Exp}}_3$ instead of $\widehat{\mathcal{O}}$. The new experiment defined in our thought experiment is denoted as follows:

$$\begin{array}{c}\mathbf{Env}'_n \leftarrow Env(n)\\ E'^1_n \leftarrow \textsc{Exp}_1(\mathbf{Env}'_n)\\ E'^2_n \leftarrow \textsc{Exp}_2(E'^1_n)\\ \widehat{E}'^3_n \leftarrow \widehat{\textsc{Exp}}_3(E'^2_n), M \in \{0,1\}, R \in \{0,1\}^{n^{\rho_2}}\\ C \leftarrow \mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)\end{array},$$

where $\mathbf{Env}'_n$ is identical to $\mathbf{Env}_n$ except that it replaces $\mathcal{O}$ with $\mathcal{O}'$, and $E'^1_n, E'^2_n$ and $\widehat{E}'^3_n$ are respectively identical to $E^1_n, E^2_n$ and $\widehat{E}^3_n$ except that $\mathbf{Env}'_n$ replaces $\mathbf{Env}_n$ in each of these variables.

It can now be observed that the random execution of $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$ from the original experiment will always have an identical output to the execution of $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK', C)$ in the modified experiment, as they can be viewed as the same execution[18]. Therefore, we can bound the probability that $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C) = M$ by bounding the probability that $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK', C) = M$. This is done by bounding the probability that $\mathbf{BAD_6}$ holds in the recast experiment, conditioning on it not occurring, and then using the result of Corollary 68 to bound the probability that $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK', C) = M$.

What is the probability that $\mathbf{BAD_6}$ holds in our modified experiment where oracle $\mathcal{O}'$ is selected in $Env(n)$ instead of $\mathcal{O}$? We note that the bound computed in Corollary 60 for $\mathbf{BAD_6}$ applies here because we can assume that we originally selected the oracle $\mathcal{O}'$ randomly in $Env(n)$. This is observed by imagining that in the original experiment, for each triple of strings $pk$, $b$ and $r$ the response to the query $(\mathbf{e}, pk, b, r)$ is not immediately fixed, and could be either $\mathbf{e}(pk, b, r)$ or $\mathbf{e}(pk, 1-b, r)$. The response is only fixed when a query that is dependent on the result is made. At that point, a coin is flipped to decide if the response to the queries $(\mathbf{e}, pk, b, r)$ and $(\mathbf{e}, pk, 1-b, r)$ should respectively be $\mathbf{e}(pk, b, r)$ and $\mathbf{e}(pk, 1-b, r)$ or vice-verse. This would clearly not change the distribution from which the oracle $\mathcal{O}$ was drawn, as we are just putting off a random decision in the oracles selection until it is needed. Since we have conditioned the random choices made in $RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS, \mathcal{IQ})$ on $\overline{\mathbf{BAD_7}}$, this is essentially what we're doing in $\overline{\mathbf{D}}$: randomly "flipping" the responses of the oracles on queries $\mathbf{e}(pk, b, r)$ and $\mathbf{e}(pk, 1-b, r)$ for some queries where it is guaranteed the "flipping" will result in an oracle from an identical distribution as the original was selected

[16]Formally we denote this change as follows: for each $(<\widehat{\mathbf{e}}, pk_c, b_c, r_c>, c) \in \mathcal{Q}_C$ if $b_c \neq b'_c$ we modify $\mathbf{e}'$ to be consistent with $(<\mathbf{e}', pk_c, b'_c, r_c>, c)$ and $(<\mathbf{e}', pk_c, b_c, r_c>, c')$ where $c' = \mathbf{e}(pk_c, b'_c, r_c)$. Next, we make any modifications to $\mathbf{d}'$ and $\mathbf{w}'$ that are necessary to accommodate the change to $\mathbf{e}'$. These are done in the obvious manner: if $\widehat{\mathbf{g}}^{-1}(pk)$ is well defined, set $\mathbf{d}(\mathbf{g}^{-1}(\widehat{\mathbf{g}}^{-1}(pk), c) \leftarrow b'_c$ and $\mathbf{d}(\mathbf{g}^{-1}(\widehat{\mathbf{g}}^{-1}(pk), c) \leftarrow b_c$; similarly, for any query $(\mathbf{w}, pk, *)$ with a response $(*, \cdots, *, c, *, \cdots, *, *)$ replace the value $c$ with $c'$, and vice-versa.

[17]We remind the reader that this conditioning effects the random choices made by *RandCipher* and not the random choices made by $Env(n), \textsc{Exp}_1, \textsc{Exp}_2$ or $\widehat{\textsc{Exp}}_3$

[18]The randomly simulated responses in $\overline{\mathbf{D}}$ are replaced with the same responses in the execution of $\mathbf{D}$ except the responses now correspond to correct responses from the oracle $\widehat{\mathcal{O}}'$.

from. The upper bound on the probability of event $\mathbf{BAD}_6$ in Corollary 60 is $\frac{(n^{2q}+1)}{2^n}$. Therefore, for all sufficiently large $n$ with probability at least

$$\begin{aligned}(1-\frac{n^{2q+1}}{2^n})(1-\frac{4n^{2q}}{n^{\alpha_6}}) &\geq (1-\frac{n^{2q+1}}{2^n}-\frac{4n^{2q}}{n^{\alpha_6}}) \\ &\geq 1-\frac{5n^{2q}}{n^{\alpha_6}}\end{aligned}$$

we have that $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)=M$ and $\overline{\mathbf{BAD}_6}$ holds.

**Bounding the Probability that the Simulation is Correct**

Maintaining our condition that $\overline{\mathbf{BAD}_6}$ holds, we turn our attention to bounding the probability that all of the simulated responses of queries to $\mathcal{O}'$ are correct. This involves a case-by-case analysis that shows that it is unlikely that any query to $\widehat{\mathbf{d}}$ or $\widehat{\mathbf{g}}$ is made in which the simulation is incorrect.

We note that the specific execution of $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK',C,\mathcal{Q}_C)$ in our experiment and the execution of $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)$ make almost exactly the same queries to the sub-oracles. The only exception is when $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)$ makes a query $(\mathbf{d},sk,c)$ in which $(<\mathbf{g},sk>,pk)\in\mathcal{G}$, $pk\in BKS^*$, $(<\mathbf{g},sk>,pk)\notin \mathit{KnownQueries}$ and $(<\mathbf{e},pk,*,*>,c)\in\mathcal{Q}_C$: in this case $\overline{\mathbf{D}}$ responds with a random bit, as opposed to making the query to the oracle, but it is specifically on these queries where we have modified the responses of $\widehat{\mathbf{d}}'$ so that they correspond to the randomly chosen bits of the execution of $\overline{\mathbf{D}}$. Therefore, it suffices to bound the probability that there are any queries are made during the execution of $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)$ whose responses would be improperly simulated by the adversary, with the exception that we can ignore the previously mentioned case. Our analysis will separately consider queries to $\widehat{\mathbf{g}}'$ and $\widehat{\mathbf{d}}'$.

**Bounding the Probability that $\overline{\mathbf{D}}$ Makes Queries to $\widehat{\mathbf{g}}'$ that are Improperly Simulated**.

If $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)$ makes a query $(\widehat{\mathbf{g}}',sk)$, then in order for the simulated reply to be incorrect it is necessary that $\widehat{\mathbf{g}}'(sk)\neq\mathbf{g}(sk)$. Fortunately, the probability that the simulated reply is incorrect is small, because there are very few strings $sk'$ for which $\mathbf{g}(sk')\neq\widehat{\mathbf{g}}'(sk')$, and it's quite unlikely that the execution of $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)$ can produce such a string, as they are distributed in a random fashion.

This is observed by noting that $\widehat{\mathbf{g}}'=\widehat{\mathbf{g}}$, as no changes were made to $\widehat{\mathbf{g}}$ in the modification of the experiment that was just described; therefore, the question of incorrect simulation comes down to the probability that $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}'}(SK',C)$ makes a query $(\widehat{\mathbf{g}},sk)$ where $\widehat{\mathbf{g}}(sk)\neq\mathbf{g}(sk)$. Such an event would imply the value $\widehat{\mathbf{g}}(sk)$ was assigned on line 5 of **ConsolidateOrac**[19].

In order for such an event to occur, it must be the case that the query/response $(<\mathbf{g},sk>,\mathbf{g}(sk))\notin$ *KnownQueries*, yet for some $\widetilde{sk}\neq sk$ it is the case that $(<\widehat{\mathbf{g}},\widetilde{sk}>,\mathbf{g}(sk))\in\mathcal{D}\cup\mathcal{G}$. In such a case, by the construction of $\widehat{\mathcal{O}}$, we have that $\widehat{\mathbf{g}}(\widetilde{sk})\leftarrow\mathbf{g}(sk)$, and there the value of $\widehat{\mathbf{g}}(sk)$ is assigned on line 5 of **ConsolidateOrac**. We will call such an event a *collision*, and say both that $sk$ was **involved in a collision** and that $\widetilde{sk}$ **caused a collision**. We bound the probability that the execution of $\mathbf{D}$ makes any query $(\widehat{\mathbf{d}},sk,*)$ where $sk$ is *involved in a collision.*

By the design of $\widehat{\text{Exp}_3}$ there are at most $|\mathcal{D}\cup\mathcal{G}|$ strings that can *cause a collision*, and therefore at most that many strings that can be *involved in a collision.* We know that $|\mathcal{G}|\leq n^q$, and it can easily be shown that $|\mathcal{D}|\leq(2n^{4q}\cdot n^{2\alpha_4}+n^s\cdot n^{2\alpha_4}\cdot n^{2q})$. Therefore, for all sufficiently large $n$ there are at most $3n^{3\alpha_4}$ strings that can be *involved in a collision.* We bound the probability of finding any such string on a given query.

If a string $sk$ is *involved in a collision* in $\widehat{\text{Exp}_3}$ then the only information that is known about it is that it is in the set $S=\{0,1\}^{|sk|}\setminus\{\overline{sk}|(<\mathbf{g},\overline{sk}>,*)\notin(\mathit{KnownQueries}\cup\mathcal{D}\cup\mathcal{G})\}$. This is the case because for any string $sk'$ for which there is a query/response of the form $(<\mathbf{g},sk'>,pk')\in\mathit{KnownQueries}\cup\mathcal{D}\cup\mathcal{G}$ it's the case that $\widehat{\mathbf{g}}(sk')=\mathbf{g}(sk')=pk'$, by the design of $\widehat{\text{Exp}_3}$, and at the end of $\text{Exp}_3$, no other queries have been made to $\mathbf{g}$. As $\mathbf{g}$ is a randomly chosen one-to-one function, there is no preference that one string in the set $S$ would be involved in a collision over another. Therefore, the probability of making a query $(\widehat{\mathbf{g}},sk)$ after $\widehat{\text{Exp}_3}$, where $sk$ is involved in a collision is no more than $\frac{|\mathcal{D}\cup\mathcal{G}|}{|S|}$. We wish to derive a bound on the probability that any of the queries of the form $(\widehat{\mathbf{g}},sk)$ made by $\mathbf{D}$ or $\mathbf{E}$ (which provides an input to $\overline{\mathbf{D}}$) is involved in a collision, and since these algorithms each make no more than $n^q$ queries, the probability is

[19]It was noted in Section N.2 that this is the only possibility for an incorrect simulated response to $\widehat{\mathbf{g}}$

less than $\frac{|\mathcal{D}\cup\mathcal{G}|}{|S|-2n^q}$, where the summand $2n^q$ accounts for information that might be learned about $\mathbf{g}$ through queries in $\mathbf{D}$ and $\mathbf{E}$. We have a bound on $|\mathcal{D}\cup\mathcal{G}|$, and by counting the number of queries of the form $(\mathbf{g},*)$ in both $\textsc{Exp}_1$ and $\textsc{Exp}_2$, it can be shown that the number of queries to $\mathbf{g}$ in *KnownQueries* is less than $(n^{2\alpha_1}\cdot n^q)+(2n^{4q}\cdot n^{2\alpha_4}+n^s\cdot n^{2\alpha_4}\cdot n^{2q})\leq 2n^{3\alpha_1}$, where the last inequality holds for all sufficiently large $n$.

Therefore, the probability that any query that $\mathbf{D}$ or $\mathbf{E}$ made finds a value *sk* that was involved in a collision as described would be no more than

$$\begin{aligned}\frac{|\mathcal{D}\cup\mathcal{G}|}{|S|-2n^q} &\leq \frac{3n^{3\alpha_4}}{|\{0,1\}^{|sk|}|-|\mathit{KnownQueries}\cup\mathcal{D}\cup\mathcal{G}|-2n^q}\\ &\leq \frac{3n^{3\alpha_4}}{n^{\alpha_0}-3n^{3\alpha_4}-2n^{3\alpha_1}-2n^q}\\ &\leq \frac{3n^{3\alpha_4}}{n^{\alpha_0}-3n^{3\alpha_1}},\end{aligned}$$

where all the inequalities hold for sufficiently large $n$. The second inequality follows from remembering that $|sk|\geq\alpha_0\cdot\log n$, as there is no possibility of an incorrect simulated response on a *small query*.

The bound we have computed is true, given every query/response that has been made to the oracle $\mathcal{O}$ in $\textsc{Exp}_1$, $\textsc{Exp}_2$ and $\widehat{\textsc{Exp}}_3$, but we are running our experiment subject to $\wedge_{i\in\{1,3,5,6\}}\overline{\mathbf{BAD}_i}$. Because, we have made use of all of the known query/responses from the experiment in the argument that derived this bound, it holds given the condition of $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$, as these events depend only on the results of query/responses in the experiment. However, our bound currently does not take into account the effects of conditioning on $\overline{\mathbf{BAD}_6}$, and therefore we need to compensate for any effect this might have. Making use of the following basic fact from probability

$$\Pr[A|B\wedge C]=\frac{\Pr[A\wedge B|C]}{\Pr[B|C]}\leq\frac{\Pr[A|C]}{\Pr[B|C]},$$

it suffices to get a bound on the event $\overline{\mathbf{BAD}_6}$ given that $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$ hold, and such a bound is provided by Corollary 60. This bound implies that for all sufficiently large $n$: $\Pr[\overline{\mathbf{BAD}_6}|\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}]\geq\frac{1}{2}$. Using this lower bound to take into account the effects of the conditioning on $\overline{\mathbf{BAD}_6}$, we conclude the probability that $\mathbf{D}$ makes a query $(\mathbf{d},sk)$ whose response will be improperly simulated is less than $\frac{6n^{3\alpha_4}}{n^{\alpha_0}-3n^{3\alpha_1}}$.

**Bounding the Probability that $\overline{\mathbf{D}}$ Makes Queries to $\widehat{\mathbf{d}}'$ that are Improperly Simulated.**

We now bound the probability that $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK',C)$ makes a query $(\widehat{\mathbf{d}}',sk,c)$ that has an incorrect simulated reply. Unlike the bound of such queries to $\widehat{\mathbf{g}}'$, we must consider several different cases that cover the possible ways in which the simulated oracle responses can go wrong. These cases correspond to the ways that simulated responses can be incorrect that are listed in Section N.2. We consider these cases below.

**Case 1. $\widehat{\mathbf{g}}(sk)$ was defined on line 5 of ConsolidateOrac.** The probability of this case can be bounded by using the same argument that is used for bounding the probability of improperly simulating a response query to $\widehat{\mathbf{g}}$. Observe that in order to make a query $\widehat{\mathbf{d}}'(sk,c)$ whose response will be improperly simulated in this this case, a string *sk* must be found for which $\widehat{\mathbf{g}}(sk)$ was defined on line 5. It is this probability that previously was bound to be less than $\frac{6n^{3\alpha_4}}{n^{\alpha_0}-3n^{3\alpha_1}}$, and therefore, the bound on this case is also $\frac{6n^{3\alpha_4}}{n^{\alpha_0}-3n^{3\alpha_1}}$.

**Case 2. $(<\mathbf{g},sk>,*)\in\mathcal{G}$, $\mathbf{g}(sk)\neq\widehat{\mathbf{g}}'(sk)$, $(<\mathbf{g},sk>,*)\notin$ *KnownQueries* and $(<\mathbf{d},sk>,c)\notin$ *KnownQueries*.**
There are several reasons why such queries are either unlikely, or unlikely to be simulated incorrectly. We consider several mutually exclusive sub-cases.

**Case 2.1 $(<\mathbf{g},sk>,pk)\in\mathcal{G}$, $pk\in BKS^*$, $(<\mathbf{g},sk>,pk)\notin$ *KnownQueries*, $\mathbf{g}(sk)\neq\widehat{\mathbf{g}}(sk)$ and there is no query/response $(<\widehat{\mathbf{e}},pk,*,*>,c)$ in $\mathcal{Q}_C$.**
Intuitively, the most likely response to the query $(\widehat{\mathbf{d}}',sk,c)$ in this case is $\perp$, as there is no query/response $(<\widehat{\mathbf{e}},pk,*,*>,c)$ in $\mathcal{Q}_C$ nor in $\mathcal{G}$; and, as has previously been mentioned, it is hard to find strings $c$ for which $\mathbf{d}(sk,c)\neq\perp$ without making such a query. If there is an error in the simulated response then there are only two possibilities: $\mathbf{d}(sk,c)\neq\perp$ or $\widehat{\mathbf{d}}(sk,c)\neq\perp$. Both

cases are unlikely. The analysis of each case is nearly identical, and therefore we will only explicitly give it for the first.

The probability of finding a string $c$ that is the image under $\widehat{\mathbf{e}}'(pk,\cdot,\cdot)$ of a pair of strings $b$ and $r$, when the query $(\widehat{\mathbf{e}}', pk, b, r)$ has not been made in the experiment is less than $\frac{2\cdot 2n^{3\alpha_1}}{n^{\alpha_0}-2n^{3\alpha_1}}$: the value of $2n^{3\alpha_1}$ in the numerator is an upper-bound on the number of elements in the domain whose images have been revealed by queries to $\widehat{\mathbf{e}}'(pk,\cdot,\cdot)$; and the denominator corresponds to the number elements in the co-domain of $\widehat{\mathbf{e}}'(pk,\cdot,\cdot)$ that might conceivably have pre-images, but for which none are known. This bound can be derived by noting that $\widehat{\mathbf{e}}'(pk,\cdot,\cdot)$ is a one-to-one function of the form $\{0,1\}\times\{0,1\}^{|sk|}\to\{0,1\}^{3|sk|}$ that was effectively chosen uniformly at random from the set of all such functions; the number of elements in the domain of $\mathbf{e}(pk,\cdot,\cdot)$ for which the corresponding image is known can be shown to have an upper-bound of less than $2n^{3\alpha_1}$. This is shown by counting the number of queries of the form $(\mathbf{e}, pk, *, *)$ that could possibly be made in $\text{Exp}_1$ and $\text{Exp}_2$. To obtain a lower bound on the number of elements in the co-domain that might have a pre-image we note that initially there were at least a possible $n^{\alpha_0}$ such values (remembering $|sk|\geq\alpha_0\cdot\log n$, as otherwise the simulated response would necessarily be correct). Next, we perform an over-estimate and assume that each query to $\widehat{\mathbf{e}}'(pk,\cdot,\cdot)$, $\widehat{\mathbf{d}}'(sk,\cdot)$ or $\widehat{\mathbf{u}}'(pk,\cdot)$ reveals a pre-image/image pairing $((b',r'),\widehat{\mathbf{e}}'(pk,b',r'))$ of the function $\widehat{\mathbf{e}}'(pk,\cdot,\cdot)$. It is easy to show there are at most $2n^{3\alpha_1}$ such queries in $\text{Exp}_1$, $\text{Exp}_2$ and $\widehat{\text{Exp}}_3$.

Because our bound takes into account the revealed values of the oracle during $\text{Exp}_1$, and $\text{Exp}_2$, it holds even when given our conditioning on $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$, as these events follow directly from the specific values of the oracle that were revealed in $\text{Exp}_1$ and $\text{Exp}_2$. However, we still need to take into account the fact that our argument conditions on $\overline{\mathbf{BAD}_6}$. Therefore, we bound the probability of these events to be at least 1/2 using the same argument that is used earlier in the proof of this lemma, when we obtained a bound on the chance of making a query to $\widehat{\mathbf{g}}$ whose simulated response would be incorrect.

**Case 2.2** $(<\mathbf{g}, sk>, pk)\in\mathcal{G}$**,** $pk\in \mathit{BKS}^*$**,** $(<\mathbf{g}, sk>, pk)\notin \mathit{KnownQueries}$ **and** $(<\mathbf{e}, pk, *, *>, c)\in\mathcal{Q}_C$**.**

These are the queries that are excluded from our analysis as $\overline{\mathbf{D}}$ will not actually make these queries, as previously mentioned. Therefore, the probability of error in this case is 0.

**Case 2.3** $(<\mathbf{g}, sk>, pk)\in\mathcal{G}\setminus \mathit{GKS}^*$ **and** $pk\notin \mathit{BKS}^*$

The probability of such a query being performed is low. If during $\text{Exp}_1$ there was a query/response $(<\mathbf{e}, pk, *, *>, c')$, then this case is not even possible: such a query/response would either require $pk\in KS$ (where $KS=(\mathit{GKS}^*\cup \mathit{BKS}^*)$), had the equality $\mathbf{u}(pk,c')=\top$ held on line 7 of $\text{Exp}_1$; and otherwise, if $\mathbf{u}(pk,c')=\bot$, then $\mathbf{g}^{-1}(pk)$ is undefined, also implying the case is not possible.

The probability that there is a query $(\mathbf{e}, pk, *, *)$ made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$ for a value $pk\notin\{pk'|(\mathbf{e}, pk', *, *)\in\mathcal{E}\}$ is no more than $1/n^{\alpha_1-2}$, as our analysis is already conditioning on $\overline{\mathbf{BAD}_6}$ and this implies that $\overline{\mathbf{BAD}_2}$ holds relative to the oracle $\widehat{\mathcal{O}}'$ that was chosen in our modified experiment. Since there are at most $n^q$ possible values for which $(<\mathbf{g}, *>, pk)\in\mathcal{G}$, the probability of this case is less than $n^q/n^{\alpha_1-2}$. Alternatively, if there was no query/response $(<\mathbf{e}, pk, *, *>, c)$ made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$, we can bound the probability of this case using the same argument and analysis that was presented in sub-case 2.1, which is based on the principle that it is difficult to find a string $c$ for which $\widehat{\mathbf{d}}'(sk,c)\neq\bot$ without making a query/response of the form $(<\widehat{\mathbf{e}}', pk, *, *>, c)$. The analysis in sub-case 2.1 bounds the probability of this case to be less than $\frac{4n^{3\alpha_1}}{n^{\alpha_0}-3n^{3\alpha_1}}$. For all sufficiently large $n$, $\frac{4n^{3\alpha_1}}{n^{\alpha_0}-3n^{3\alpha_1}}\leq\frac{n^q}{n^{\alpha_1-2}}$, so $\frac{n^q}{n^{\alpha_1-2}}$ bounds the probability of this sub-case.

We can bound the probability of Case 2 by taking the maximum of the bounds for the three mutually disjoint cases that were presented: $\max\{\frac{n^q}{n^{\alpha_1-2}}, 0, \frac{4n^{3\alpha_1}}{n^{\alpha_0}-2n^{2\alpha_1}}\}=\frac{n^q}{n^{\alpha_1-2}}$ for all sufficiently large $n$.

**Case 3.** $(<\widehat{\mathbf{g}}', sk>, pk)\in\mathcal{D}$ **there is no query/response** $(<\mathbf{g}, sk>, *)\in(\mathcal{G}\cup \mathit{KnownQueries})$**,** $\mathbf{g}(sk)\neq\widehat{\mathbf{g}}'(sk)$**, and** $(<\mathbf{d}, sk>, c)\notin \mathit{KnownQueries}$

Again, we can consider two familiar cases: whether or not there was a query/response $(<\mathbf{e}, pk, *, *>, c)$ that was made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$. If no such query/response was made, then the probability of this case is less than $\frac{4n^{3\alpha_1}}{n^{\alpha_0}-3n^{3\alpha_1}}$, as per sub-case 2.1.

The case in which a query of the form $(<\widehat{\mathbf{e}}', pk, *, *>, c)$ was made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$ is also unlikely. First, note that the probability of making any query of the form $(\mathbf{e}, pk, *, *)$ during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$ is less than $1/n^{\alpha_1-2}$ because of our conditioning on $\overline{\mathbf{BAD}_6}$ and $\overline{\mathbf{BAD}_1}$. The condition $\overline{\mathbf{BAD}_1}$ implies that there are no surprising queries to $\mathbf{d}$ made in $\textsc{Exp}_1$, and since –by definition of the current case– there is no query $(<\widehat{\mathbf{g}}', sk>, pk)$ in $\mathcal{G}$ there can be no queries of the form $(\mathbf{e}, pk, *, *)$ in the set $\mathcal{E}$ computed in $\textsc{Exp}_1$. Therefore, since $\mathbf{BAD}_6$ holds, we know that $\mathbf{BAD}_2$ effectively holds in our modified experiment, and therefore the the probability that $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$ makes a query/response of the form $<\widehat{\mathbf{e}}', pk, *, *>, c)$ is less than $\frac{1}{n^{\alpha_1-2}}$.

The probability of this case is therefore less than $\max\left\{\frac{4n^{3\alpha_1}}{n^{\alpha_0}-3n^{3\alpha_1}}, \frac{1}{n^{\alpha_1-2}}\right\} = \frac{1}{n^{\alpha_1-2}}$.

**Case 4.** $(<\mathbf{g}, sk>, pk) \in \mathit{KnownQueries}$ **for some** $pk$**, but** $(<\mathbf{e}, pk, *, *>, c)$ **was defined on line 18 of ConsolidateOrac.**

Again, we can consider two familiar cases: whether or not there was a query $(<\mathbf{e}, pk, *, *>, c)$ that was made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$. If no such query/response was made, then the probability of this case is less than $\frac{4n^{3\alpha_1}}{n^{\alpha_0}-3n^{3\alpha_1}}$, as per sub-case 2.1.

We bound the probability of the the alternate case in which such a query/response $(<\mathbf{e}, pk, *, *>, c)$ is made. We use an argument similar to the one that showed it was improbable that responses of queries to $\widehat{\mathbf{g}}'$ are improperly simulated. We begin by noting that the values of $\widehat{\mathbf{e}}(pk, *, *)$ that were defined on line 18 of **ConsolidateOrac** are few in number and randomly distributed, making them very unlikely to be queried during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$. The probability of an improper response in this case comes down to the probability that $\mathbf{E}^{\widehat{\mathbf{O}}'}(PK, M, R)$ makes a query/response $(<\widehat{\mathbf{e}}, pk, b, r>, c)$ where the value of $\widehat{\mathbf{e}}(\widehat{\mathbf{g}}(sk), b, r)$ was assigned on line 18 of **ConsolidateOrac**. In order for this to occur it must be the case that there is no pair of strings, $(b, r)$, such that $(<\mathbf{e}, pk, b, r>, c) \in \mathit{KnownQueries}$, but that for some $(\tilde{b}, \tilde{r}) \neq (b, r)$ it is the case that $(<\widehat{\mathbf{e}}, pk, \tilde{b}, \tilde{r}>, c) \in \mathcal{G}$ (remembering that there are no query/response $(<\mathbf{e}, *, *, *>, *) \in \mathcal{D}$, as $\mathbf{D}$ does not query the decryption sub-oracle). In this case $\widehat{\mathbf{e}}(pk, b, r)$ is assigned on line 18. This means that $(b, r)$ could be any pair of values in the set $\{0,1\} \times \{0,1\}^{|sk|} \setminus \{(b', r')|(<\mathbf{e}, pk, b', r'>, *) \in \mathit{KnownQueries} \cup \mathcal{G} \cup \mathcal{D}\}$.

We now remark to the reader that the remainder of the analysis for this case is similar to the case where we bounded the probability of an improperly simulated response of a query to $\widehat{\mathbf{g}}'$. Because of the random process $\Upsilon$ by which $\mathcal{O}$ was chosen, the probability of finding such a pair, $(b, r)$, is less than $\frac{n^q}{n^{\alpha_0}-2n^{3\alpha_1}}$. This bound takes into account all previous query/responses to the oracle, and thus holds when conditioning on $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$, as these events rely only upon the results of queries made during $\textsc{Exp}_1$ and $\textsc{Exp}_2$. Our bound does not take into account conditioning on $\overline{\mathbf{BAD}_6}$. By Corollary60, for all sufficiently large $n$: $\Pr[\overline{\mathbf{BAD}_6}| \wedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}] \geq \frac{1}{2}$. We multiply our previous bound by 2, thereby taking into account the effects of the conditioning on $\overline{\mathbf{BAD}_6}$ and arrive at the bound of: $\frac{2n^q}{n^{\alpha_0}-2n^{3\alpha_1}}$ for this case.

The probability of making a query for which the simulation will be incorrect is bound to be less than the maximum of all of the previously discussed cases:$\max\{\frac{6n^{3\alpha_4}}{n^{\alpha_0}-3n^{3\alpha_1}}, \frac{n^q}{n^{\alpha_1-2}}, \frac{1}{n^{\alpha_1-2}}, \frac{2n^q}{n^{\alpha_0}-2n^{3\alpha_1}}\} = \frac{n^q}{n^{\alpha_1-2}}$. There are at most $n^q$ queries that are made during the execution of $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}'}(SK', C)$ and therefore, the probability of an error is less than $\frac{n^{2q}}{n^{\alpha_1-2}}$. Therefore, with probability at least $1 - \frac{n^{2q}}{n^{\alpha_1-2}}$, we have a successful simulation, given that $\overline{\mathbf{BAD}_6}$ holds.

Combining this bound with the bounds on the likelihood that $\mathbf{D}^{\widehat{\mathbf{g}}',\widehat{\mathbf{d}}'}(SK', C) = M$ and that $\overline{\mathbf{BAD}_6}$ holds, we have that we have both a successfully encryption and simulation with probability at least: $1 - \frac{n^{2q}}{n^{\alpha_1-2}} - \frac{5n^{2q}}{n^{\alpha_6}} \geq 1 - \frac{6n^{2q}}{n^{\alpha_6}}$, where the inequality holds for all sufficiently large $n$. □

## N.5 The Final Decryption Algorithm: $\widehat{\mathbf{D}}$

The problem with the algorithm $\overline{\mathbf{D}}$ is that in order to use it to decrypt ciphertexts generated by the call to *RandCipher* it requires as an input the set $\mathcal{Q}_C$ which contains the replacement query/responses that were made during the execution of *RandCipher*. The challenge ciphertext that the adversary is attempting to decrypt will be generated by an execution of $\mathbf{E}$, in which there is no notion of replacement query/responses — and, even if such a corresponding notion existed, the adversary would not have access to them — so we need to develop an algorithm that does not require access to a set $\mathcal{Q}_C$.

We modify the algorithm $\overline{\mathbf{D}}$ to produce an algorithm $\widehat{\mathbf{D}}$ that will, with high probability, also decrypt ciphertexts generated by *RandCipher* and be simulatable by the adversary, but it will have the benefit of not requiring a set of replacement query/responses as an input. The intuition behind the design of the algorithm $\widehat{\mathbf{D}}$ is as follows: in the experiment described in Lemma 69 it is very unlikely that $\overline{\mathbf{D}}^{\widehat{\mathbf{O}}}(SK', C, \mathcal{Q}_C)$ will make a decryption query $(\widehat{\mathbf{d}}, sk, c)$ that has a response in $\{0,1\}$ in those cases where the values $sk$ and $c$ have not been encoded into either $SK'$, $C$ or both. Therefore, for any query $(\mathbf{d}, sk, c)$ where $\mathbf{g}(sk) \in BKS^*$ if we know both that there is no query response $(<\mathbf{e}, \mathbf{g}(sk), *, *>, c) \in \mathcal{IQ}$ and that $\widehat{\mathbf{d}}(sk, c) \in \{0,1\}$, then with high probability the string $c$ was a replacement response during the execution of *RandCipher*, and therefore there would be a corresponding query in $Q_C$. In $\widehat{\mathbf{D}}$ we assume this to be the case and reply with a random coin flip. This algorithm is formalized below.

```
$\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK, C, BKS^*, \mathcal{IQ}^*)$
(1)   Simulate the execution of $\mathbf{D}^{\mathbf{g},\mathbf{d}}(SK, C)$
(2)       On oracle query $(\widehat{\mathbf{g}}, sk)$ reply with $\widehat{\mathbf{g}}(sk)$
(3)       On oracle query $(\widehat{\mathbf{d}}, sk, c)$
(4)           If there exists a $b$ and $r$ such that $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), b, r>, c) \in \mathcal{IQ}^*$ reply with $\widehat{\mathbf{d}}(sk, c)$
(5)           If $\widehat{\mathbf{g}}(sk) \notin BKS^*$ reply with $\widehat{\mathbf{d}}(sk, c)$
(6)           If $\widehat{\mathbf{u}}(\widehat{\mathbf{g}}(sk), c) = \top$ then reply with $b_c \in_R \{0,1\}$
(7)           Reply with $\perp$
(8)   Output the result of the simulation
```

We will execute $\widehat{\mathbf{D}}$ with the set $BKS^*$ that is constructed by $\text{Exp}_2$ in our experiment. The set $\mathcal{IQ}^* = \mathcal{IQ} \cup \mathcal{G}$ where $\mathcal{IQ}$ was generated in $\text{Exp}_2$ and the set $\mathcal{G}$ generated in $\text{Exp}_3$. We will show that $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*)$, will function in an almost equivalent manner to $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, Q_C)$ when used in the experiment conducted in the experiments we have been discussing (specifically, the experiment stated in Lemma 69). The problem is that $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*)$ could make a decryption query $(\widehat{\mathbf{d}}, sk, c)$ whose response is not $\perp$ and for which there has never been a corresponding encryption query/response $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, *>, c)$, and in this case the execution might not be equivalent to $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, Q_C)$. Fortunately, the probability of such an event is low. We formally define this notion below and bound the probability of the event.

We first remind the reader of the notation introduced in Notn. 57, which allows us to denote by $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK, C, BKS, \mathcal{IQ} \cup \mathcal{D})$ (resp. $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK, C, \mathcal{Q}_C)$), the execution of where the string $r \in \{0,1\}^{n^q}$ specifies the random bits to be used during the computation of $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK, C, BKS, \mathcal{IQ} \cup \mathcal{D})$ (resp. $\overline{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK, C, \mathcal{Q}_C)$). Specifically, we let the $i$th bit of $r$ represent the response to the $i$th request for a random bit $b_c$ in $\widehat{\mathbf{D}}$ (resp. $b'_c$ in $\overline{\mathbf{D}}$).

**Lemma 70.** *For all sufficiently large n:*

$$\Pr[\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M \textit{ and is properly simulated} | \bigwedge_{i \in \{1,3,5,7\}} \overline{\mathbf{BAD}_i}] \geq 1 - \frac{8n^{2q}}{n^{\alpha_6}},$$

*where the experiment is*

$$\begin{array}{c} \mathbf{Env}_n \leftarrow Env(n) \\ E_n^1 \leftarrow \text{Exp}_1(\mathbf{Env}_n) \\ E_n^2 \leftarrow \text{Exp}_2(E_n^1) \\ \widehat{E}_n^3 \leftarrow \widehat{\text{Exp}}_3(E_n^2), M \in \{0,1\}, R \in \{0,1\}^{n^{\rho_2}}, r \in \{0,1\}^{n^q} \\ C \leftarrow RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ}) \end{array},$$

*and* $\mathcal{IQ}^* = \mathcal{IQ} \cup \mathcal{G}$.

*Proof.* We will prove this result, by comparing the execution of $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS, \mathcal{IQ})$ in the experiment with a corresponding execution of $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$, where $\mathcal{Q}_C$ is the set of *replacement query/responses* made by the execution of *RandCipher*. We remind the reader that, by Lemma 69, the probability that both $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C) = M$ and that the execution is properly simulated by the adversary is at least $1 - \frac{6n^{2q}}{n^{\alpha_6}}$. We will assume that if either $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C) \neq M$ or the execution of $\overline{\mathbf{D}}$ is improperly simulated then either $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}) \neq M$ or the execution of $\widehat{\mathbf{D}}$ is improperly simulated.

First, let us look at the probability that the executions of $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ})$ and $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$ will diverge, and show that it is small. Both algorithms simulate the execution of $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C)$ and modify the responses to some of the oracle queries. Neither algorithm modifies the responses of queries to $\widehat{\mathbf{g}}$, so there is no chance the simulated executions can diverge on such a query. The situation is substantially different with queries to $\widehat{\mathbf{d}}$. Let us consider the two simulations' responses to a query $(\widehat{\mathbf{d}}, sk, c)$. We first note that if $\widehat{\mathbf{g}}(sk) \notin BKS^*$ then the two algorithms necessarily provide the same simulated reply to the query, for if there is no query/response $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, *>, c) \in \mathcal{Q}_C$ then $\overline{\mathbf{D}}$ responds with $\widehat{\mathbf{d}}(sk, c)$. Let $pk = \widehat{\mathbf{g}}(sk)$. By the definitions of *RandCipher* and the set $\mathcal{Q}_C$, if $pk \notin BKS^*$ then there cannot be a query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$, and therefore in this case the simulated responses of $\overline{\mathbf{D}}$ and $\widehat{\mathbf{D}}$ will always be equivalent. Similarly, if there was a query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{IQ}^*$, then, because $\overline{\mathbf{BAD}_7}$ holds, there can be no query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$ and therefore $\overline{\mathbf{D}}$ would reply to the query $(\widehat{\mathbf{d}}, sk, c)$ with the response $\widehat{\mathbf{d}}(sk, c)$. There are only two remaining possibilities: firstly, that there is no query $(<\widehat{\mathbf{e}}, pk, *, *>, c)$ in the set $\mathcal{IQ}^* \cup \mathcal{Q}_C$, and, secondly, that there is a query $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$. In the first case, $\overline{\mathbf{D}}$ will respond with $\widehat{\mathbf{d}}(sk, c)$ but there are two possibilities for $\widehat{\mathbf{D}}$'s response: it will respond with a random coin-flip in the case that $\widehat{\mathbf{u}}(pk, c) = \top$, and otherwise when $\widehat{\mathbf{u}}(pk, c) = \bot$ it responds with $\bot$. We will show that it is likely that $\widehat{\mathbf{d}}(sk, c) = \bot$, and therefore that $\widehat{\mathbf{u}}(pk, c) = \bot$. This implies that it is likely that both $\overline{\mathbf{D}}$ and $\widehat{\mathbf{D}}$ simulate a reply of $\bot$.

In the second case, in which there is a query $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$, both executions respond with a coin-flip; since both executions share the same random string $r$ — assuming that the two executions have not diverged, and thus that all previous queries in the executions have had the same simulated responses — both $\overline{\mathbf{D}}$ and $\widehat{\mathbf{D}}$ will provide the same simulated response.

Based on the previous case analysis, it suffices to bound the probability that any query $(\widehat{\mathbf{d}}, sk, c)$ is made during the simulation of $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C)$ where $\widehat{\mathbf{d}}(sk, c) \neq \bot$, $pk \in BKS^*$ and there is no query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{IQ}^* \cup \mathcal{Q}_C$. That such queries should be unlikely is not surprising, as for any string $sk$ it is very hard to find a string $c$ for which $\widehat{\mathbf{d}}(sk, c) \neq \bot$ without making a query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c)$. We note that the query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c)$ was not made during the generation of $\mathbf{G}^{\widehat{\mathbf{O}}}(S') \rightarrow (PK, SK')$ due to the fact that there is no query/response of the form $(<\widehat{\mathbf{e}}, pk, *, *>, c)$ in the set $\mathcal{IQ}^* \supset \mathcal{G}$. Further, no such query/response was made during the execution of $RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ}) \rightarrow C$, as $\overline{\mathbf{BAD}_7}$ holds and there is no query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$. Since these executions account for generating the only two inputs to $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C)$, there is only a very small probability that execution of $\mathbf{D}$ makes a query $(\widehat{\mathbf{d}}, sk, c)$ for which $\widehat{\mathbf{d}}(sk, c) \neq \bot$.

We will call a query $(\widehat{\mathbf{d}}, sk, c)$ *informative* if there has been no previous query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c)$ that was made during either the execution of $\mathbf{G}^{\widehat{\mathbf{O}}}(S') \rightarrow (PK, SK')$ or $RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ}) \rightarrow C$, where $pk = \widehat{\mathbf{g}}(sk)$, and $\widehat{\mathbf{d}}(sk, c) \neq \bot$ [20]. At the end of the proof of this lemma we present Claim 71, and it bounds the probability that $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$ makes an informative query in the experiment to be less than $\frac{2}{n^{\alpha_0}}$. Therefore, with probability $1 - \frac{2}{n^{\alpha_0}}$ we have that $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C) = \widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*)$. Combining this with the fact that, by Lemma 69, with probability at least $1 - \frac{6n^{2q}}{n^{\alpha_6}}$ both $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C) = M$ and its execution is properly simu-

[20] We note the notion of an informative query is similar, but not identical to the notion of a surprising response, which was presented in Defn. 29.

lated by the adversary, we conclude that with probability at least $1 - \frac{6n^{2q}}{n^{\alpha_6}} - \frac{2}{n^{\alpha_0}}$ it is the case that $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M$. This bound is greater than $1 - \frac{7n^{2q}}{n^{\alpha_6}}$ for all sufficiently large $n$.

Unfortunately, without additional argument, we cannot also conclude that the execution of $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*)$ will be properly simulated by the adversary, as there are queries made to $\widehat{\mathbf{u}}$ during the execution of $\widehat{\mathbf{D}}$ that may not be properly simulated. Assuming that we are in the case just described where all of the following hold: $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C) = M$, its execution is properly simulated by the adversary and all of the simulated query/responses in the two executions correspond, then any query to $\widehat{\mathbf{d}}$ or $\widehat{\mathbf{g}}$ made by $\widehat{\mathbf{D}}$ will, assuming all queries to $\widehat{\mathbf{u}}$ are properly simulated, have necessarily been made by $\overline{\mathbf{D}}$, and therefore, the execution of $\widehat{\mathbf{D}}$ will be properly simulated. Therefore, we show that with high probability every query of the form $(\widehat{\mathbf{u}}, \widehat{\mathbf{g}}(sk), c)$ made in $\widehat{\mathbf{D}}$ on line 6 is properly simulated [21]. If a query $(\widehat{\mathbf{d}}, sk, c)$ was made by $\overline{\mathbf{D}}$ and was properly simulated, where $\widehat{\mathbf{g}}(sk) = pk$, then the query $(\widehat{\mathbf{u}}, pk, c)$ will have its response properly simulated. This can be seen by noting the symmetry in the definitions of $\widehat{\mathbf{u}}$ and $\widehat{\mathbf{d}}$ in **ConsolidateOrac** and the symmetry in the adversary's simulation of the response to such queries, as is presented in Section N.1. The symmetry in definitions imply that if the response to the query $(\widehat{\mathbf{d}}, sk, c)$ is properly simulated, then so is the response to the query $(\widehat{\mathbf{u}}, pk, c)$. Therefore, it suffices to show that the response to the query $(\widehat{\mathbf{u}}, pk, c)$ is simulated properly in the case where there is a query $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$, for it is only these queries to $\widehat{\mathbf{u}}$ for which $\overline{\mathbf{D}}$ will not have made a corresponding query $(\widehat{\mathbf{d}}, sk, c)$, and therefore there is no guarantee that these responses will be properly simulated.

Given that $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{Q}_C$ we know that $pk \in BKS^*$. Let $b$ and $r$ be the values for which $(<\widehat{\mathbf{e}}, pk, b, r>, c) \in \mathcal{Q}_C$. Because $\overline{\mathbf{BAD}_7}$ holds, there have been no previous query/responses $(<\widehat{\mathbf{e}}, pk, b, r>, c)$ in $KnownQueries \cup \mathcal{G}$. Because $\widehat{\mathbf{e}}(pk, b, r) = c$, it is the case that $\widehat{\mathbf{u}}(pk, c) = \top$, and therefore there can only be an error if the simulated response to the query $(\widehat{\mathbf{u}}, pk, c)$ is $\perp$; such a simulated response can only occur if $\mathbf{u}(pk, c) = \perp$, and this can only be the case if there are no values $b'$ and $r'$ for which $\mathbf{e}(pk, b', r') = c$. In particular this implies that $\mathbf{e}(pk, b, r) \neq \widehat{\mathbf{e}}(pk, b, r) = c$. The only way for this to happen is if the value of $\widehat{\mathbf{e}}(pk, b', r')$ was set on line 8 or line 18 of **ConsolidateOrac**.

If the value of $\widehat{\mathbf{e}}(pk, b', r')$ was set on line 8, then the simulated response to $\widehat{\mathbf{u}}(pk, c)$ is guaranteed to be correct by the definition of the simulation. Therefore, it suffices to bound the probability that the value of $\widehat{\mathbf{e}}(pk, b', r')$ was set on line 18. We bound the probability of this case by bounding the probability that *RandCipher* make any such query in the experiment, which it must do if such a query is to be in the set $\mathcal{Q}_C$.

For a given value $pk \in BKS^*$, the execution of *RandCipher* makes at most $2n^q$ replacements queries of the form $(<\widehat{\mathbf{e}}, pk, *, *>, c)$, and for each such query $(\widehat{\mathbf{e}}, pk, b, r)$ that is made, the strings $b \in \{0,1\}$ and $r \in \{0,1\}^{|sk|}$ are chosen uniformly at random conditioned on the event $\overline{\mathbf{BAD}_7}$. This condition means that $r$ is selected so that there is no query $(\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, r)$ in the set $KnownQueries \cup \mathcal{G}$, nor has such a query previously been made during the current execution of *RandCipher*. Observe that for sufficiently large $n$, the value $n^{3\alpha_1}$ is a large overestimate on the number of queries that are disallowed due to our conditioning on $\overline{\mathbf{BAD}_7}$. We also know that the total number of pairs $(b, r)$ for which the value of $\widehat{\mathbf{e}}(pk, b, r)$ is set on line 18 can easily be overestimated by $n^{3\alpha_1}$. Therefore, the probability that a given replacement query, made by *RandCipher*, has a response whose value was assigned on line 18 of **ConsolidateOrac** is less than $\frac{2n^{3\alpha_1}}{n^{\alpha_0} - 2n^{3\alpha_1}}$. As previously mentioned, there are at most $2n^q$ queries made for a given key $pk \in BKS^*$, and there are at most $n^q$ such keys in $BKS^*$. Therefore, by the union-bound the probability of an incorrect simulation is less than $\frac{4n^{3(\alpha_1+q)}}{n^{\alpha_0} - 2n^{3\alpha_1}}$. Therefore, it is highly likely that that the simulation of the responses of queries to $\widehat{\mathbf{u}}$ will be correct. Combining this bound with the previously derived bound on proper decryptions, we get that the probability that $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M$ and that its execution is properly simulated by the adversary is at least

$$1 - \frac{7n^{2q}}{n^{\alpha_6}} - \frac{4n^{3(\alpha_1+q)}}{n^{\alpha_0} - 2n^{3\alpha_1}} \geq 1 - \frac{8n^{2q}}{n^{\alpha_6}}$$

.

It remains to prove the claim that bounds the probability that an *informative* query is made during the execution of $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$. We remind the reader that a response to the query $(\widehat{\mathbf{d}}, sk, c)$ is *informative*

[21] Observe that that the query $(\widehat{\mathbf{g}}, sk)$ embedded in the query $(\widehat{\mathbf{u}}, \widehat{\mathbf{g}}(sk), c)$ will always be simulated correctly because the query/response $(<\widehat{\mathbf{g}}, sk>, \widehat{\mathbf{g}}(sk)) \in \mathcal{G}$ as $\widehat{\mathbf{g}}(sk) \in BKS^*$ by virtue of line 5 of $\widehat{\mathbf{D}}$, and all such queries are properly simulated.

if there has previously been no query/response of the form $(<\widehat{\mathbf{e}}, \widehat{\mathbf{g}}(sk), *, *>, c)$ that was made during either the execution of $\mathbf{G}^{\widehat{\mathbf{O}}}(S') \rightarrow (PK, SK')$ or $\mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ}) \rightarrow C$ in the experiment.

**Claim 71.** *The probability that in the experiment of Lemma 70 that* $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$ *performs an informative query/response is less than* $\frac{2}{n^{\alpha_0}}$.

*Proof.* Intuitively there are only two ways for $\overline{\mathbf{D}}$ to make an informative query: firstly, it might be that $sk$ and $c$ are encoded into $\overline{\mathbf{D}}$'s inputs, $SK'$ and $C$; or, secondly, $\overline{\mathbf{D}}$ could attempt to find such values itself. However, since $\mathbf{G}$ does not call $\mathbf{d}$, by Assumption 21, we don't need to worry about the input $SK$ encoding any strings that would form informative queries. In order for $sk$ and $c$ to be encoded into $C$, the execution of *RandCipher* needs to make an informative query. We will first bound the probability that $\mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, \mathcal{Q}_C)$ makes an *informative decryption query* to be less than $\frac{n^q}{n^{\alpha_0}}$ and will argue there is a similar bound on the probability that $\overline{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}}}(SK', C, \mathcal{Q}_C)$ makes such a query given that *RandCipher* did not. We begin with former bound.

Consider an execution of $\mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, \mathcal{Q}_C)$ where the $i$th query that it performs is $(\widehat{\mathbf{d}}, sk, c)$ where the string $pk = \widehat{\mathbf{g}}(sk)$ is in $BKS^*$, there is no query/response $(<\mathbf{e}, pk, *, *>, c) \in \mathcal{IQ}^*$, and there is no query/response $(<\mathbf{e}, pk, *, *>, c)$ that has previously been made during the prior $i-1$ queries made during the execution of *RandCipher*.

Given that the the function $\widehat{\mathbf{e}}(pk, \cdot, \cdot)$ is randomly selected from the set of all possible functions of the form $\{0,1\} \times \{0,1\}^{|sk|} \rightarrow \{0,1\}^{3|sk|}$, the probability that there exists strings $b$ and $r$ such that $\widehat{\mathbf{e}}(pk, b, r) = c$ is less than $\frac{2^{|sk|+1}}{2^{3|sk|} - 3n^{2\alpha_1}}$: we have subtracted $3n^{2\alpha_1}$ in the denominator to compensate for information about $\widehat{\mathbf{e}}(pk, \cdot, \cdot)$ that may have been learned, via direct queries to the oracle, either in the experiment of Lemma 70 or during the earlier $i-1$ queries of *RandCipher*. Since $pk = \widehat{\mathbf{g}}(sk) \in BKS^*$ we know that $|sk| \geq \alpha_0 \cdot \log n$, and therefore this bound is less than $\frac{n^{\alpha_0+1}}{n^{3\alpha_0} - 3n^{2\alpha_1}}$ for all sufficiently large $n$. Using the union-bound to account for the fact that *RandCipher* could perform $n^q$ different queries to $\mathbf{d}$, we see that the probability that any such informative query is made during the execution of $\mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, \mathcal{Q}_C)$ is less than $\frac{n^{\alpha_0+q+1}}{n^{3\alpha_0} - 3n^{2\alpha_1}} \leq \frac{1}{n^{\alpha_0}}$, where the inequality holds for all sufficiently large $n$.

Next, we bound the probability that $\mathbf{D}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, \mathcal{Q}_C)$ makes an *informative query* given that that $C \leftarrow \mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, \mathcal{Q}_C)$ has made no *informative* queries. Using an almost identical argument to that just presented we can bound the probability to being less than $\frac{1}{n^{\alpha_0}}$. Given the similarity and redundancy of the arguments, we will not present it here. Therefore, we can bound the probability of an *informative* query to be less than $\frac{2}{n^{\alpha_0}}$ □

□

## N.6 Using $\widehat{\mathbf{D}}$ to Decrypt the Challenge Ciphertext

The previous lemma shows that the adversary can use $\widehat{\mathbf{D}}$ to properly decrypt ciphertexts generated by *RandCipher* with high probability. Our final lemma shows that the adversary can also use $\widehat{\mathbf{D}}$ to decrypt ciphertexts generated by $\mathbf{E}$, and thus the adversary will likely be able to decrypt its challenge ciphertext.

The intuition as to why $\widehat{\mathbf{D}}$ should do a good job decrypting ciphertexts generated by $\mathbf{E}$ is that $\widehat{\mathbf{D}}$ cannot effectively distinguish ciphertexts generated by $\widehat{\mathbf{D}}$ and *RandCipher*, and therefore it cannot do a better job decrypting ciphertexts generated from one algorithm vs. the other. The distinction between an execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ and an execution of $\mathit{RandCipher}^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ})$ is that *RandCipher* simulates the execution of $\mathbf{E}$ but it replaces the response of the query/responses $(<\widehat{\mathbf{e}}, pk, b, r>, c)$ that are both not in $\mathcal{IQ}$ and for which $pk \in BKS^*$, with $c' = \widehat{\mathbf{e}}(pk, b', r')$ for randomly chosen $b'$ and $r'$. Since neither the query $(\widehat{\mathbf{e}}, pk, b, r)$ nor $(\widehat{\mathbf{e}}, pk, b', r')$ is common (the former because it's not in $\mathcal{IQ}$ and the latter because $b'$ and $r'$ are chosen randomly), it's not likely that either has been made before at any point in any of the experiments. Assuming this is the case, from the perspective of both the adversary and the decryption algorithm the response to each query is effectively a random string, and the strings would not be effectively distinguishable by any party with limited queries to $\widehat{\mathbf{O}}$. Therefore, the decryption algorithm cannot decrypt substantially better in one scenario versus the other, or it would imply an ability to effectively distinguish the responses of the oracle on previously unknown queries.

This intuition is formalized in the proof of the following lemma.

**Lemma 72.** *For all sufficiently large n:*

$$\Pr_{\substack{\mathbf{Env}_n \leftarrow Env(n)\\ E_n^1 \leftarrow \mathrm{EXP}_1(\mathbf{Env}_n)\\ E_n^2 \leftarrow \mathrm{EXP}_2(E_n^1)\\ \widehat{E}_n^3 \leftarrow \widehat{\mathrm{EXP}}_3(E_n^2), M \in \{0,1\}\\ R \in \{0,1\}^{\rho_2}, C \leftarrow \mathbf{E}^{\widehat{\mathbf{O}}}(PK,M,R)}} [\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M \text{ and is properly simulated}] | \bigwedge_{i \in \{1,3,5\}} \overline{\mathbf{BAD}_i}] \geq 1 - \frac{9n^{2q}}{n^{\alpha_6}},$$

*where* $\mathcal{IQ}^* = \mathcal{IQ} \cup \mathcal{G}$

*Proof.* The only difference between the experiments stated in this lemma and that of Lemma 70 is that in Lemma 70 the ciphertext $C$ is the result of an execution of $RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ})$ conditioned on the event $\overline{\mathbf{BAD}_7}$, but in this lemma the ciphertext $C$ is the result of an execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ with no conditioning.

We show that with high probability we can view the output of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ as the output of a random execution of $RandCipher^{\widehat{\mathbf{O}}'}(PK, M, R, BKS^*, \mathcal{IQ})$ conditioned on $\overline{\mathbf{BAD}_7}$. Here $\widehat{\mathbf{O}}'$ is an oracle that is nearly identical to $\widehat{\mathbf{O}}$, but from the perspective of $\widehat{\mathbf{D}}$ is indistinguishable from $\widehat{\mathbf{O}}$. This will imply that it is likely that $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M$.

First, we remind the reader of the experiment from Lemma 70 which showed that $\widehat{\mathbf{D}}$ did a good job decrypting ciphertexts generated by *RandCipher*, which we restate below:

$$\substack{\mathbf{Env}_n \leftarrow Env(n)\\ E_n^1 \leftarrow \mathrm{EXP}_1(\mathbf{Env}_n)\\ E_n^2 \leftarrow \mathrm{EXP}_2(E_n^1)\\ \widehat{E}_n^3 \leftarrow \widehat{\mathrm{EXP}}_3(E_n^2), M \in \{0,1\}, R \in \{0,1\}^{n^{\rho_2}}, r \in \{0,1\}^{n^q}\\ C \leftarrow RandCipher^{\widehat{\mathbf{O}}}(PK,M,R,BKS^*,\mathcal{IQ})},$$

given that $\bigwedge_{i \in \{1,3,5,7\}} \overline{\mathbf{BAD}_i}$ holds. Remember that Lemma 70 shows that it is likely that as a result of the above experiment that both $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M$ and that such an execution is properly simulated.

Consider a modified execution of $RandCipher^{\widehat{\mathbf{O}}}(PK, M, R, BKS^*, \mathcal{IQ})$ in the above experiment where for each $pk \in BKS^*$ and each replacement query $(\widehat{\mathbf{e}}, pk, b_i, r_i)$ that is made during its execution we replace the query's response with the value $\widehat{\mathbf{e}}(pk, b_i', r_i')$ where $b_i'$ and $r_i'$ are chosen subject to two constraints: first, that the query $(\widehat{\mathbf{e}}, pk, b_i', r_i')$ was not been made in the experiment, and second that the query was not previously made in the current, modified execution of *RandCipher*. Let $C'$ be the ciphertext that results from this modified execution. The probability that $\widehat{\mathbf{D}}_r^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C', BKS^*, \mathcal{IQ}^*) = M$ and that such an execution is properly simulated is at least $1 - \frac{8n^{2q}}{n^{\alpha_6}}$, as was the case in Lemma 70. This is the case because from the perspective of $\widehat{\mathbf{D}}$ the distribution on ciphertexts is identical. It does not query $\widehat{\mathbf{e}}$, and therefore it has no ability to distinguish between the two sequences of responses. Further, the probability of each sequence of responses is identical. This is because, although we have changed the queries, the responses are effectively randomized due to the random selection of the oracle. Note that queries to $\widehat{\mathbf{u}}$ cannot help the adversary to distinguish between the two sequences because they only confirm whether or not ciphertexts have valid decryptions, and so $\widehat{\mathbf{u}}(pk, \widehat{\mathbf{e}}(pk, b_i, r_i) = \widehat{\mathbf{e}}(pk, b_i', r_i') = \top$, for each replacement response $\widehat{\mathbf{e}}(pk, b_i, r_i)$ and its corresponding response $\widehat{\mathbf{e}}(pk, b_i', r_i')$. Further, queries of the form $(\widehat{\mathbf{d}}, sk, c)$ are not made by $\widehat{\mathbf{D}}$ in the case that $\widehat{\mathbf{g}}(sk) = pk \in BKS^*$ and there is no query/response $(<\widehat{\mathbf{e}}, pk, *, *>, c) \in \mathcal{IQ}^*$; both of these properties are satisfied for each replacement query and its modified response.

The latter property is satisfied in the original experiment due to the fact that $\overline{\mathbf{BAD}_7}$ holds, and the constraints on the modifications to the experiment ensure the property is satisfied in the modified experiment. Further, since all of the queries whose responses are being modified are both distinct and never previously queried in the original experiment, there is an equal likelihood that the oracle selected is one where for each $i$ the response to the query $(\widehat{\mathbf{e}}, pk, b_i, r_i)$ is $\widehat{\mathbf{e}}(pk, b'_i, r'_i)$ as opposed to $\widehat{\mathbf{e}}(pk, b_i, r_i)$. Therefore, the probability that $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C', BKS^*, \mathcal{IQ}^*) = M$ in the modified experiment is the same as in the original experiment, as any discrepancy would result in a distinguisher between the sub-oracles $\mathbf{O}$ and $\mathbf{O}'$; given our discussion and the fact that each sub-oracle is equally likely, this is not possible. Similarly, the probability that the simulation of $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C', BKS^*, \mathcal{IQ}^*)$ is correct is the same as the probability that the simulation of $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*)$ is correct for similar reasons.

Therefore, in order to prove the lemma it suffices to show that with high probability, for each $pk \in BKS^*$ it is the case that each of the queries $(\mathbf{e}, pk, b, r) \notin \mathcal{IQ}^*$ that is made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ has not previously been queried in the experiment and is distinct. We will consider the probability in two mutually exclusive cases: when $\mathbf{BAD}_6$ holds and when it does not. We remember that $\overline{\mathbf{BAD}_6}$ tells us that $\mathcal{IQ}$ contains all of the queries that are likely to be made during an execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ with probability greater than $\frac{1}{n^{\alpha_1-2}}$. We will assume that when $\mathbf{BAD}_6$ holds there will be some query $(\mathbf{e}, pk, b', r') \notin \mathcal{IQ}^*$ that is made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ that was previously queried in the experiment, and therefore that $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C', BKS^*, \mathcal{IQ}^*) \neq M$. However, in the case that $\overline{\mathbf{BAD}_6}$ holds, then we know that the probability that any of the first $i$ queries of the form $(\widehat{\mathbf{e}}, pk, b, r) \notin \mathcal{IQ}^*$ for which $pk \in BKS^*$ have the property that they're distinct and have not previously occurred in the experiment is at least $(1 - \frac{n^{3\alpha_4}}{n^{\alpha_1-2}})$, where $n^{3\alpha_4}$ is a gross overestimate on the number of possible queries that have been made to $\widehat{\mathbf{e}}(pk, \cdot, \cdot)$ during $\textsc{Exp}_1, \textsc{Exp}_2$ and $\widehat{\textsc{Exp}}_3$ and that are also not contained in $\mathcal{IQ}^*$. Therefore, we can bound the probability that all of the first $i$ queries are distinct and have not previously occurred to be at least $(1 - \frac{n^{3\alpha_4}}{n^{\alpha_1-2}})^i$. Since there are at most $n^q$ such queries that are made during the execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$, this probability is at least:

$$\begin{aligned}(1 - \frac{n^{3\alpha_4}}{n^{\alpha_1-2}})^{n^q} &\geq 1 - \frac{n^{3\alpha_4+q}}{n^{\alpha_1-2}}. &(2)\\ &\geq 1 - \frac{n^{4\alpha_4}}{n^{\alpha_1-2}} &(3)\\ & &(4)\end{aligned}$$

Therefore, for all sufficiently large $n$, the probability that the execution of $\mathbf{E}^{\widehat{\mathbf{O}}}(PK, M, R)$ satisfies all of the required conditions is at least

$$\begin{aligned}(1 - \frac{n^{4\alpha_4}}{n^{\alpha_1-2}}) \cdot \Pr[\overline{\mathbf{BAD}_6} | \wedge_{i\in\{1,3,5\}} \overline{\mathbf{BAD}_i}] &\geq 1 - \frac{n^{4\alpha_4}}{n^{\alpha_1-2}} - \frac{3n^{2q}}{2^n} \text{ (by Corollary 60 )}\\ &\geq 1 - \frac{n^{5\alpha_4}}{n^{\alpha_1-2}} \text{ (for all sufficiently large } n)\end{aligned}$$

Therefore, by the union bound, for all sufficiently large $n$, with probability at least

$$1 - \frac{n^{5\alpha_4}}{n^{\alpha_1-2}} - \frac{8n^{2q}}{n^{\alpha_6}} \geq 1 - \frac{9n^{2q}}{n^{\alpha_6}}$$

we have that $\widehat{\mathbf{D}}^{\widehat{\mathbf{O}}}(SK', C, \mathcal{IQ}^*) = M$ and that the adversary properly simulates the computation. □

## O Putting it all Together

We finally show how to put together the lemmas from this chapter in order to prove our main theorem.

**Theorem. 26.**
*There exists a CCA#1 adversary $A = (A_1, A_2)$ for which it's the case that for all sufficiently large $n$:*

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

*Proof.* We note that the distribution that $\mathcal{O}$, $PK$ and $SK$ are chosen from is identical to the one generated by $Env(n)$. Therefore, our adversary $A_1$ will treat $\mathcal{O}$ and $PK$ as if they had been output by $Env(n)$ and perform $\textsc{Exp}_1$. We note that $\textsc{Exp}_1$ makes no use of $SK$, and therefore the adversary can easily perform the experiment. Let $E_n^1 = (\mathbf{Env}_n, KS, \mathcal{IQ}, \mathcal{E}, SQ)$ be the result of $\textsc{Exp}_1$. Next, $A_1$ performs $\textsc{Exp}_2(E_n^1)$. Note that while $\textsc{Exp}_2$ does make use of the value $SK$, which $A_1$ does not have access to, it only makes use of the value in computing $\mathbf{D}^{\mathbf{O}}(SK, \cdot)$, and the adversary can use the decryption oracle in order to compute these values. Let $E_n^2 = (\tau^*, BKS^*, GKS^*, E_n^1)$ be the result of $\textsc{Exp}_2$. If $\textsc{Exp}_1$ or $\textsc{Exp}_2$ were to halt without completing, which is an unexpected case, then $A_1$ will output a special symbol that directs $A_2$ to output a bit that is chosen uniformly at random. Assuming that $\textsc{Exp}_1$ and $\textsc{Exp}_2$ terminated in the expected fashion, $A_1$ performs $\widehat{\textsc{Exp}}_3(E_n^2)$. Let $E_n^3 = (\mathbf{Env}', \mathcal{G}, \mathcal{D}, E_n^2)$ where $\mathbf{Env}'_n = (\mathcal{O}', PK, SK', S')$, and $A_1$ outputs $E_n^3$.

The second part of the adversary, $A_2$, will now compute $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C, BKS^*, \mathcal{IQ}^*) = M'$ and output the result, where $\mathcal{IQ}^* = \mathcal{IQ} \cup \mathcal{G}$. We remind the reader that $A_2$ cannot necessarily perform this computation perfectly because it has to simulate the responses of oracle queries to $\widehat{\mathbf{g}}, \widehat{\mathbf{d}}$ and $\widehat{\mathbf{u}}$. However, using many of the lemmas that are proven in this chapter, we show that it's likely that the simulated responses are correct and that $M' = M$.

We begin by showing that it is likely that $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$ holds in the experiments that $A_1$ performs. We will assume the worst case: if any of the events $\overline{\mathbf{BAD}_1}$,$\overline{\mathbf{BAD}_3}$ or $\overline{\mathbf{BAD}_5}$ fails to hold then it is necessarily the case that the output of $A_2$ is not $M$. We first note that by Lemma 54 it's the case that $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$ holds with probability at least $1 - \frac{1}{3n^{\alpha_0}}$. Next, we would like to show that it is likely that $C' = C \leftarrow \mathbf{E}^{\mathbf{O}'}(PK, M, R)$. Corollary 64 tells us that given that $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$ holds, this is the case with probability at least $1 - \frac{n^{3\alpha_4}}{n^{\alpha_1 - 2}}$; and Lemma 72 tells us that, also assuming that $\wedge_{i\in\{1,3,5\}}\overline{\mathbf{BAD}_i}$ holds, the probability that both $\widehat{\mathbf{D}}^{\widehat{\mathbf{g}},\widehat{\mathbf{d}},\widehat{\mathbf{u}}}(SK', C', \mathcal{IQ}^*) = M$ and that the execution of $\widehat{\mathbf{D}}$ is properly simulated is at least $1 - \frac{9n^{2q}}{n^{\alpha_6}}$. Therefore, $A_2$ outputs $M$ with probability at least

$$1 - \frac{1}{3n^{\alpha_0}} - \frac{n^{3\alpha_4}}{n^{\alpha_1-2}} - \frac{9n^{2q}}{n^{\alpha_6}} \geq 1 - \frac{1}{n},$$

where the inequality holds for all sufficiently large $n$. □

## P Conclusions, Open Questions and Future Work

The main result of this chapter shows that there is no black-box proof of security for any black-box construction of a CCA#1 secure PKEP from a semantically secure PKEP, if the constructed encryption algorithm does not query the black-box decryption algorithm. The model used in this chapter permits each party to use an arbitrary amount of time to perform computation. This includes the adversary, and may seem to be unrealistic. However, if one looks at the proof used in the experiment, the only point in which the adversary uses its unlimited computational ability is in the selection of the oracle and seed pair $(\mathcal{O}', S')$ in lines 5 through 14 of $\textsc{Exp}_3$. However, it is easy to see that the selection of $(\mathcal{O}', S')$ is really just the random selection of an *NP*-witness from the set *ValidEnvironments*. At first glance, it may seem like this is not the case, as we're selecting an infinitely large oracle $\mathcal{O}'$, but of course we only need to select the oracle with queries up to size $n^s$ and thus it is finite. Yet, we're still left with an exponentially large description. A quick review of $\textsc{Exp}_3$, $\widehat{\textsc{Exp}}_3$, $\widehat{\mathbf{D}}$ and the simulation of the oracle $\widehat{\mathcal{O}}$ shows that the only part of $\mathcal{O}'$ that we actually make use of are those query/responses that are contained in the sets $\mathcal{D}$ and $\mathcal{G}$ output by $\textsc{Exp}_3$, and they are polynomial in size. Therefore, if we could efficiently and uniformly at random select an *NP*-witness from an *NP*-set, we could execute the description of our adversary in probabilistic, polynomial time. A result by Bellare, Goldreich and Petrank [5], shows that with access to an *NP*-oracle, a probabilistic polynomial time adversary could perform all of the experiments and algorithms in this chapter. This implies that finding a black-box proof of CCA#1 security for a construction ruled out in this chapter (i.e. those ruled out in Theorem 26), would imply a separation of $P$ and $NP$. This is the same principle that is behind Impagliazzo' and Rudich's results in [20], which showed that any black-box construction of a key-exchange primitive from a one-way function would imply that $P \neq NP$.

It is still an open question as to whether or not it can be shown that there is no black-box construction of a CCA#1 secure PKEP from a semantically secure PKEP, but we feel the restriction presented in this chapter is neither obviously necessary and not likely to be made use of in many reasonable constructions, and therefore the result presents serious evidence there may not be any such constructions.