Efficiently-Searchable and Deterministic Asymmetric Encryption

Mihir Bellare[1] Alexandra Boldyreva[2] Adam O'Neill[2]

[1]Dept. of Computer Science and Engineering, University of California at San Diego,
mihir@cse.ucsd.edu www.cse.ucsd.edu/~mihir/

[2]College of Computing, Georgia Institute of Technology,
{aboldyre,amoneill}@cc.gatech.edu www.cc.gatech.edu/{~aboldyre,~amoneill}

Abstract

Outsourcing data storage is a topic of emerging importance in database security. In this paper, we consider exact-match query functionality in the public-key setting. Solutions proposed in the database community lack clarity and proofs of security, while encryption-with-keyword-search schemes from the cryptographic community require linear search time (in database size) for each query, which is prohibitive. To bridge the gap, we introduce a new cryptographic primitive we call (asymmetric) efficiently-searchable encryption (ESE), which allows users to store encrypted data on a remote, untrusted server in such a way that the server can index the data and efficiently retrieve or update required parts of the data on request. We give an appropriate definition of security for ESE and several constructions that provably-achieve the definition, in the random oracle model, while providing various computation- and bandwidth-efficiency properties. As deterministic encryption implies ESE, the security definition and several constructions are also the first for asymmetric deterministic encryption schemes in general.

Keywords: asymmetric encryption, searchable encryption, deterministic encryption, database security.

1 Introduction

The Problem. Despite a continuous decrease of storage-hardware prices, the costs to store and manage data, providing availability, recoverability, security and regulatory compliance, are rapidly increasing. For many organizations, it is most cost-effective to outsource data to specialized off-site service providers [51]. Data is usually divided into records (aka. tuples) with attributes (aka. fields) and stored in a relational database. The service must provide at a minimum the following:

- **Query support**. Users should be able to actively retrieve a desired portion of the data on request.
- **Computation and communication efficiency**. The server should efficiently support indexing and query processing and provide data availability.

- **Security**. Very often users are concerned about the confidentiality of their data, as it may contain some sensitive financial, medical, or intellectual information. In a medical context, securing the data is actually required by law. In 2003, the U.S. Department of Health and Human Services issued the Privacy Rule to implement the requirement of the Health Insurance Portability and Accountability Act [1], whose Section 164.306 requires the health care organizations to "ensure the confidentiality, integrity, and availability of all electronic protected health information the covered entity creates, receives, maintains, or transmits." Even though the remote database service providers may employ strong security measures against outsider attacks, they themselves cannot always be trusted not to mistreat the data of their clients. Therefore, clients must appropriately protect sensitive information before outsourcing it to a remote service provider.[1]

The basic set-up constitutes the so-called outsourced database model (aka. Database-as-a-Service or DAS). Finding a good security-functionality tradeoff for DAS is a challenging research problem that has been recently receiving a great deal of attention in the database community [27, 44, 29, 28, 16, 17, 44, 4, 30, 3, 31, 37]. Notably, M. Kantracioglu and C. Clifton [31] recently pointed out that standard cryptographic (i.e., semantic-security-strength) security definitions are too strong to allow the server to perform any useful indexing on encrypted data, forcing it to scan the whole database on each query. Database practitioners do not seriously consider such protocols with search time linear in the database size because medium-size to very large databases, which can occupy up to several terabytes of space, do not fit in memory and having many disk accesses per query is prohibitively slow. This is perhaps why prior research did not produce any provably-secure solutions: practitioners required functionality for which the schemes satisfying the standard security notions were not suitable, and theoreticians did not see the need to look beyond these strong definitions.

Related Work. On the one hand, work in the database community focuses mostly on supporting flexible (i.e., SQL) queries and efficient and optimized query processing. They propose ad-hoc cryptographic schemes to index encrypted data in support of these tasks [44, 4, 27, 17, 29, 30, 28], which refer to such primitives as order-preserving hash functions and encryption or deterministic encryption without suggesting appropriate definitions of security or candidate constructions. Indeed, the drawbacks inherent in this approach cause [31] to call for a new direction for research on secure database servers aiming instead for "efficient encrypted database and query processing with *provable* security properties."

On the other, research done by cryptographers in the area of database security targets strong security goals and provides provably-secure constructions. Most works, starting with a paper by D. Song, D. Wagner and A. Perrig [52] in the symmetric-key setting and one by Boneh et al. [11] in the asymmetric setting, focus on the better-defined subproblem of secure keyword search on encrypted data [11, 26, 23, 12, 2, 6]. In essence, the schemes allow a server, just when given some secret information, to test whether a ciphertext (e.g., an encrypted email) contains particular keywords, without revealing the keywords or any other information about the message. But using them to answer queries asking to find records containing a particular keyword essentially requires testing each record in the database one-by-one, which, as explained above, is prohibitive. While symmetric-setting indexing methods suggested in [52] instead require scanning a list of possible

[1] There are additional security concerns that we do not address, such as whether the server correctly responds to queries. We note that the server is either trusted in this regard or some special measures [50] can be employed.

keywords on each query, the problem here is that the size of such an index would often be on the order of the size of the database, e.g., for patient ID numbers in medical records.

Obfuscated databases, studied more recently in [42], allow public access to data for those who know for what exactly they search (i.e., a phone number for Bob) while preventing mass-harvesting (getting all email addresses). The solutions of [42] asymmetrically distinguish between obfuscated, searchable attributes (e.g., phone numbers) and possibly-encrypted data attributes (e.g., email addresses), in that one cannot later search for an email address and retrieve the corresponding phone number. In our setting, however, one typically wants various attributes to be simultaneously searchable and decryptable, and in any case these protocols also require an entire database scan on each query.

Works on secure multiparty computation [53, 25], private information retrieval [14, 35, 19], searching on streaming data [43], oblivious RAM [24] and secure data mining and statistical databases [47, 15, 32, 13, 34, 18, 10] also study related but fundamentally different problems in which a server stores data that is usually not encrypted, and its privacy shall be protected as the privacy of the users querying the data.

Our Contributions. Our results address exact-match queries, where queries ask to retrieve records containing given attribute values, as well as queries that ask to update such records or insert new records altogether for applications that require them. Exact-match is a basic operation that happens to be particularly frequent in existing databases because it is the basis of a more complex operation called equijoin. We focus on the asymmetric (aka. public-key) setting, where querying the database requires only the public key, so, in a medical context, for example, nurses, clerical workers, etc. can update sensitive attributes in medical records stored in a remote database by encrypting data under the public key for the intended physician and submitting it appropriately to the server. They can similarly retrieve records for physicians, but, without the secret key to decrypt certain parts, only a patient's physician can ultimately view his or her full record.

In Section 3, we define what it means for (asymmetric) encryption to support this additional functionality, a primitive we call *efficently-searchable encryption* (ESE) (Definition 3.1). Informally, there should exist two extra functions: one that takes the public key and a plaintext, and one that takes the public key and any ciphertext for the same plaintext, such that their output strings (called the tag) agree, and for all distinct messages, their tags under a given public key agree only with exceedingly small probability (taken over the coin tosses of the key-generation algorithm). Presence of the tags thus ensures that search mechanisms used in retrievals, updates, or inserts on encrypted databases will be essentially the same as on unencrypted ones (this is what "efficiently" refers to), which is appealing to implementors. For example, the server can index records according to ESE-encrypted attributes via their tags using a tree-based index such as a B-tree, allowing logarithmic search time when a tag or new record is submitted.

It turns out that finding a suitable security definition for ESE is not straightforward. The adversary, given a challenge ciphertext, can always compute the tag of the underlying message and compare it with the tag of candidate plaintexts. Furthermore, we would like to capture the intuition that the adversary cannot compute any useful information about the message given the ciphertext, but clearly the the tag *is* some useful information. Thus we make two relaxations in our security definition of *privacy against chosen-plaintext* (resp. *-ciphertext*) attacks (*priv-cpa* [resp. *-cca*]) (Definition 3.4), as compared to the standard indistinguishability-based ind-cpa or -cca notion. For one, we consider message spaces that have "enough entropy;" lack of security otherwise is unavoidable with the given functionality requirement. Note that non-sensitive or non-searchable

parts of records would still be encrypted normally or left in the clear, as appropriate, so do fall under any such constraint. To make the definition achievable, we also do not allow the useful information the adversary needs to compute to depend on the public key, which we argue is fine in practice.

In Section 4, we propose and analyze several ESE schemes, the high-level ideas for which, namely using the hash of the message as the tag or some form of deterministic encryption (for which the tag and ciphertext coincide), are discussed informally in the database literature. First we show that the former construction, which we call "hash-and-encrypt" (Construction 3.2), yields a priv-cpa (resp. priv-cca) scheme in the random-oracle (RO) model [7] if the underlying encryption scheme is ind-cpa (resp. ind-cca) (Theorem 4.1). Then we propose a general "encrypt-with-hash" deterministic ESE construction with greater bandwidth- and computation-efficiency (Construction 4.2), replacing the randomness used in encryption by a standard scheme with the hash of the message. We show this instead achieves priv-cca in the RO model assuming only that the underlying encryption scheme is ind-cpa and satisfies a slight additional property (Theorem 4.3). Moreover, any ind-cpa scheme can be easily modified to achieve this additional property, which is needed only to achieve priv-cca (vs. priv-cpa) in the construction based on ind-cpa security of the underlying scheme, but in practice this is not even necessary since practical ind-cpa schemes already posses it. Our last construction is a deterministic, length-preserving ESE scheme based on RSA-OAEP [8, 20], which we call RSA-DOAEP (Construction 4.4). Here we show that RSA-DOAEP is priv-cpa in the RO model assuming RSA is one-way (Theorem 4.5). Note that the first two constructions can be used to efficiently encrypt messages of various lengths by making the underlying scheme a hybrid encryption scheme. For RSA-DOAEP, we also show how to efficiently encrypt messages of arbitrary length without making use of any hybrid scheme.

Furthermore, in Section 5 we show that in addition to providing authentication of data, analogous to in the standard (i.e., non-ESE) asymmetric setting [5], digital signatures can actually boost security of efficiently-searchable encryption when used in an "encrypt-then-sign" fashion (Theorem 5.2). In particular, RSA-DOAEP in fact achieves priv-cca anyway when combined with a secure digital signature scheme using this construction.

We stress that there is nothing "application-specific" about our deterministic ESE schemes. In addition to helping researchers and developers working on securing outsourced databases, we expect our work would be of independent interest as providing the first definitions and constructions for asymmetric deterministic encryption schemes, which can be used more generally whenever messages to encrypt contain "enough entropy," such as with symmetric keys.[2] Since cryptographic schemes are proven secure assuming a source truly random bits whereas computers are in actuality deterministic, implementing "randomness" generation in practice is a tricky process that can end up compromising security. Using a deterministic scheme instead is therefore desirable whenever possible in terms of security guarantee.

2 Preliminaries

We denote by $\{0,1\}^*$ the set of all binary strings of finite length. We will refer to members of $\{0,1\}^*$ as strings. If X is a string then $|X|$ denotes its length in bits and if X, Y are strings then $X \parallel Y$ denotes the concatenation of X and Y. If S is a set then $X \xleftarrow{\$} S$ denotes that X is

[2]We do not imply, however, that priv-cpa or priv-cca asymmetric deterministic schemes can be securely used in all known applications, e.g., hybrid encryption.

selected uniformly at random from S. For convinience, for any $k \in N$ we write $X_1, X_2, \ldots, X_k \xleftarrow{\$} S$ as shorthand for $X_1 \xleftarrow{\$} S, X_2 \xleftarrow{\$} S, \ldots, X_n \xleftarrow{\$} S$. "RPT" (resp. "PT") stands for "randomized, polynomial-time," (resp. "polynomial-time") and "RPTA" (resp. "PTA") for "RPT algorithm" (resp. "PT algorithm"). If A is a randomized algorithm then $A(x, y, \ldots; R)$, or $A(x, y, \ldots)$ for short, denotes the result of running A on inputs $x, y, \ldots$ and with coins R, and $a \xleftarrow{\$} A(x, y, \ldots)$ means that we choose R at random and let $a = A(x, y, \ldots; R)$. Recall that a function $f\colon \mathbb{N} \to [0, 1]$ is called *negligible* if it approaches zero faster than the reciprocal of any polynomial, i.e., for any polynomial p, there exists $n_p \in \mathbb{N}$ such that for all $n \geq n_p$, $f(n) \leq 1/p(n)$.

We recall the standard syntax and security definitions for asymmetric (aka. public-key) encryption schemes in Appendix A.

3 Efficiently-Searchable Encryption (ESE) and its Security

We extend Definition A.1 in a way that captures encryption schemes that allow to efficiently search encrypted databases essentially in the same way as for unencrypted ones, a new primitive we call efficiently-searchable encryption (ESE) schemes. We refer the reader to the introduction for a discussion of how the schemes can be used and why they are useful.

Definition 3.1 [Efficiently-searchable encryption scheme] Let $\mathcal{SPE} = (\mathcal{K}, \mathcal{SE}, \mathcal{SD})$ be a public-key encryption scheme with associated security parameter $k \in \mathbb{N}$ and the *message space* $\mathrm{MsgSp}(k)$ that can also depend on some public parameters, e.g. a group description. We say that $\mathcal{SPE}$ is *efficiently-searchable encryption (ESE)* scheme if there exist PTAs F, G and a negligible function $\epsilon(\cdot)$ such that the following conditions hold:
(1) Completeness:

$$\Pr\left[\ (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ f_1 \leftarrow F(pk, m_1)\ ;\ g_1 \leftarrow G(pk, \mathcal{SE}(pk, m_1))\ :\ f_1 = g_1\ \right] = 1 \text{ and}$$

(2) Soundness:

$$\Pr\left[\ (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ f_2 \leftarrow F(pk, m_0)\ ;\ g_2 \leftarrow G(pk, \mathcal{SE}(pk, m_1))\ :\ f_2 = g_2\ \right] \leq \epsilon(k)\ ,$$

for all distinct messages $m_0, m_1 \in \mathrm{MsgSp}(k)$. We refer to the output of F, G as the *tag* of a message m or a corresponding ciphertext C.

Now we formally define the examples of ESE discussed in the introduction. The first one is perhaps the simplest approach one could take in which the tag of a message is its hash, while the second shows that deterministic encryption can also be used to achieve ESE.

Construction 3.2 [Hash-and-encrypt construction] Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a (standard) pubic-key encryption scheme and H be a hash function. We define a new public-key encryption scheme whose ciphertexts include some extra, "searchable" information. Namely, define $\mathcal{HPE} = (\mathcal{K}, \mathcal{HE}, \mathcal{HD})$, where $\mathcal{HE}(pk, m) = H(m) \parallel \mathcal{E}(pk, m)$ and $\mathcal{HD}(sk, h \parallel C) = \mathcal{D}(sk, C)$ if $H(\mathcal{D}(sk, C)) = h$ and $\perp$ otherwise. Then $\mathcal{HPE}$ is efficiently-searchable under Definition 3.4 if H is a randomly-chosen member of a collision-resistant family of hash functions. Here we let F and G be the PTAs that on inputs pk, m and $pk, H(m) \parallel \mathcal{E}(pk, m)$, respectively, return the tag $H(m)$.

Construction 3.3 [Deterministic encryption schemes] Let $\mathcal{DPE} = (\mathcal{K}, \mathcal{DE}, \mathcal{D})$ be a public-key encryption scheme such that $\mathcal{DE}$ is deterministic (meaning PT). Then letting F and G be

the PTAs that on inputs pk, m and $pk, \mathcal{DE}(pk, m)$, respectively, return $\mathcal{DE}(pk, m)$, we see that every deterministic encryption scheme is effciently-searchable under Definition 3.1. Note that the function $\epsilon(\cdot)$ in the definition is zero here due to the consistency requirement in Definition A.1.

It is easy to see that no ESE scheme can be ind-cpa. We introduce a definition of security for asymmetric ESE schemes, which captures the intuition that the tag should not reveal any information about the corresponding plaintext beyond what is needed for the ability to index and search for it efficiently. The high-level idea for our definition originates with the definition of semantic security in [38]; however, we make several important relaxations in our definition that are explained below. Note that by Example 3.3 our definition applies just as well to asymmetric deterministic encryption as to ESE in general.

Definition 3.4 [Privacy of efficiently-searchable encryption schemes] Let $\mathcal{SPE} = (\mathcal{K}, \mathcal{SE}, \mathcal{SD})$ be an asymmetric ESE scheme with associated security parameter $k \in \mathbb{N}$ and the *message space* $\mathrm{MsgSp}(k)$. Let $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ be a pair of algorithms, the latter with oracle access. (We clarify that $A_{\mathrm{m}}, A_{\mathrm{g}}$ are distinct algorithms that share neither coins nor state.) A_{m} takes input the security parameter, and returns a message $m \in \mathrm{MsgSp}(k)$ together with a string t that represents some information about m. Later, A_{g} gets a public key and an encryption of m under this key, and tries to compute t. (Note that t is not required to be efficiently computable given m. For example, A_{m} could output m, t such that $m = f(t)$ for a one-way function f). For $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$, define the following experiments:

Experiment $\mathbf{Exp}^{\text{priv-atk-1}}_{\mathcal{SPE},A}(k)$	**Experiment** $\mathbf{Exp}^{\text{priv-atk-0}}_{\mathcal{SPE},A}(k)$
$(t_1, m_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$	$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $A_{\mathrm{m}}(t_1, m_1) \xleftarrow{\$} (1^k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$	$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$
$g \xleftarrow{\$} A_{\mathrm{g}}^{\mathcal{O}(sk,\cdot)}(pk, \mathcal{SE}(pk, m_1))$	$g \xleftarrow{\$} A_{\mathrm{g}}^{\mathcal{O}(sk,\cdot)}(pk, \mathcal{SE}(pk, m_0))$
If $g = t_1$ then return 1	If $g = t_1$ then return 1
Else return 0	Else return 0

where $\mathcal{O}(sk, \cdot) = \mathcal{D}(sk, \cdot)$ if $\mathrm{atk} = \mathrm{cca}$ and is the empty oracle otherwise. We say that $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ is a *privacy adversary* if there is a function l such that for every k:

$$\Pr\left[\ (t, m) \xleftarrow{\$} A_{\mathrm{m}}(1^k)\ :\ |m| = l(k)\ \right]\ =\ 1\ ,$$

and A_{g} does not query its challenge ciphertext to its decryption oracle. The *privacy-advantage* of a privacy adversary A against $\mathcal{SPE}$ is the function defined for all k via:

$$\mathbf{Adv}^{\text{priv-atk}}_{\mathcal{SPE},A}(k)\ \ =\ \ \Pr\left[\ \mathbf{Exp}^{\text{priv-atk-1}}_{\mathcal{SPE},A}(k) = 1\ \right] - \Pr\left[\ \mathbf{Exp}^{\text{priv-atk-0}}_{\mathcal{SPE},A}(k) = 1\ \right]\ .$$

It would be natural to now define an asymmetric ESE scheme $\mathcal{SPE} = (\mathcal{K}, \mathcal{SE}, \mathcal{SD})$ to be *private against chosen-plaintext attack* or *priv-cpa* for $\mathrm{atk} = \mathrm{cpa}$ and *private against chosen-ciphertext attack* or *priv-cca* for $\mathrm{atk} = \mathrm{cca}$ if for every RPT privacy adversary A the function $\mathbf{Adv}^{\text{priv-atk}}_{\mathcal{SPE},A}(\cdot)$ is negligible. (When we say that A is RPT we mean that $A_{\mathrm{m}}, A_{\mathrm{g}}$ are both RPTAs.) However, under this definition, *no* ESE scheme is private. To see this, consider the RPTA A_{m} that on input 1^k picks t at random from $\{0, 1\}$ and returns $(0, 0^k)$ if $t = 0$ and $(1, 1^k)$ if $t = 1$. Let A_{g} be the RPTA that on input pk, C returns 0 if $G(pk, C) = F(pk, 0^k)$ and 1 otherwise. Then according to Definition 3.1, for any $\mathrm{atk} \in \{\mathrm{cpa}, \mathrm{cca}\}$ $\Pr\left[\ \mathbf{Exp}^{\text{priv-atk-1}}_{\mathcal{SPE},A}(k) = 1\ \right]\ =\ 1$. However

$\Pr\left[\mathbf{Exp}^{\text{priv-atk-0}}_{\mathcal{SPE},A}(k) = 1\right] \leq 1/2$ because A_{g} gets no information about the bit t_1 chosen by A_{m} in the experiments. So $A = (A_{\text{m}}, A_{\text{g}})$ is a RPT privacy adversary such that $\mathbf{Adv}^{\text{priv-atk}}_{\mathcal{SPE},A}(k) \geq 1/2$, meaning the scheme is not private. What this shows is that the best we can hope for is security against privacy adversaries $A = (A_{\text{m}}, A_{\text{g}})$ where the message space implicitly defined by A_{m} has large min-entropy. To capture this, we say that $\text{me}_A(\cdot)$ is a *message space min-entropy* function for A if for every $m^* \in \{0,1\}^*$ and every k:

$$\Pr\left[(t, m) \xleftarrow{\$} A_{\text{m}}(1^k) \ : \ m = m^*\right] \ \leq \ \frac{1}{2^{\text{me}_A(k)}} \ .$$

Then we will see that security can be achieved against adversaries for which this function is super-logarithmic, which we take as our definition.

Note that lack of security when the message space min-entropy is small is an inescapable consequence of having an efficiently-searchable scheme, not a weakness in our definition. Thus in this respect what we show is the best possible. Further note that A_{m} is not given the public key in the experiments, meaning we only provide security for messages unrelated to the public key. In practice this is just fine, because no normal data set is related to any public key. In real life, adversaries do not pick the data and public keys are abstractions hidden in our software, not strings that we look at.

The proofs of security for all ESE schemes we propose will be done in the random oracle (RO) model [7]. According to the RO model, a random function with appropriate domain and range (the RO) is chosen during the key generation algorithm, and then all the algorithms (adversaries included) are given oracle access to this function. Thus in the above definition A_{g} will also have access to the RO. However A_{m} will not because, as explained above, it does not have the public key, which in practice contains a key for the hash function family used to instantiate the random oracle, meaning A_{m} cannot compute a hash without it.

4 Secure ESE Constructions

We propose and analyze several constructions of ESE schemes. To begin with, we analyze the "simple" construction given in Construction 3.2 in which the hash of the message is used as its tag.

4.1 Hash-and-Encrypt

We analyze security of the hash-and-encrypt construction given in Construction 3.2.

Theorem 4.1 Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme and let H be a hash function. Let $\mathcal{HPE} = (\mathcal{K}, \mathcal{HE}, \mathcal{HD})$ the ESE scheme defined according to Construction 3.2. Then $\mathcal{HPE} = (\mathcal{K}, \mathcal{HE}, \mathcal{HD})$ is priv-cpa (resp. priv-cca) secure in the RO model if $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is ind-cpa (resp. ind-cca). More precisely, for $\text{atk} \in \{\text{cpa}, \text{cca}\}$, let $A = (A_{\text{m}}, A_{\text{g}})$ be a RPT privacy adversary against $\mathcal{HPE}$ having message space min-entropy function $\text{me}_A(\cdot)$ and making at most $q_h(\cdot)$ queries to its hash oracle. Then there exists an RPT ind-cca-adversary B against $\mathcal{PE}$ such that for all k,

$$\mathbf{Adv}^{\text{priv-atk}}_{\mathcal{HPE},A}(k) \ \leq \ \mathbf{Adv}^{\text{ind-atk}}_{\mathcal{PE},B}(k) + \frac{2q_h(k)}{2^{\text{me}_A(k)}} \ . \tag{1}$$

The proof is in Appendix B.

Assuming the underlying scheme is ind-cca, the concrete security of the reduction in the theorem implies that the construction achieves security in typical applications when message space min-entropy is on the order of, say, 80 bits, since this means the adversary needs to make about 2^{80} hash computations to break the resulting scheme. This will be the case for our other constructions as well.

Based on the argument concluding Definition 3.4, one can intuit that ESE cannot achieve better security than asymmetric deterministic encryption, which leads to the remaining constructions.

4.2 Encrypt-with-Hash

We propose a general deterministic ESE construction replacing the coins used by a standard encryption scheme with the hash of the message, which is more bandwidth- and computation-efficient than the "hash-and-encrypt" construction. The new construction is also priv-cca in the RO model assuming only the underlying scheme is ind-cpa and satisfies a slight additional property met by ind-cpa schemes in practice. A similar result for priv-cpa security does not need any additional properties of the underlying scheme besides its ind-cpa security. Recall that we introduced deterministic encryption in general and showed it implies ESE in Example 3.3. The construction is as follows.

Construction 4.2 [Encrypt-with-hash construction] Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme and let H be a hash function. Then we define a public-key deterministic encryption scheme $\mathcal{DPE} = (\mathcal{K}, \mathcal{DE}, \mathcal{DD})$, where $\mathcal{DE}(pk, m) = \mathcal{E}(pk, m; H(m))$ and $\mathcal{DD}(C) = \mathcal{D}(sk, C)$ if $\mathcal{E}(pk, \mathcal{D}(sk, C); H(\mathcal{D}(sk, C)) = C$ and $\perp$ otherwise. Here we assume that when the security parameter is k then the output of H has the same length as the random tape of $\mathcal{E}$.

To define the extra property of the underlying encryption scheme that we will need, consider the probability that a message encrypts twice to the same ciphertext using independent random coins. Namely, we say that a public-key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has a *max-collision probability* $\mathrm{mc}_{\mathcal{PE}}(\cdot)$ if we have that:

$$\Pr\left[\ (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ C_1, C_2 \xleftarrow{\$} \mathcal{E}(pk, m)\ :\ C_1 = C_2\ \right] \leq \mathrm{mc}_{\mathcal{PE}}(k)$$

for every $m \in \mathrm{MsgSp}(k)$. (We clarify that m is allowed to depend on pk.) We will show that the construction achieves priv-cca if the underlying encryption scheme is ind-cpa and moreover has negligible max-collision probability.

Note that ind-cpa schemes that would typically be used in practice have zero or negligible max-collsion probability. For example, one can check that the ind-cpa version of RSA-OAEP (i.e., without needing to pad zeros onto the plaintext) [9] has max-collision probability equal to $1/2^{n(k)}$, where $n(\cdot)$ is the length of the messages to encrypt, and ElGamal [21] has zero max-collision probability. Thus in practice this does not amount to any extra assumption. Moreover, we show how in general any ind-cpa scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ can be modified to achieve this property. Let r be the number of coins $\mathcal{E}$ uses, and let k be the security parameter. Define $\mathcal{PE}^* = (\mathcal{K}, \mathcal{E}^*, \mathcal{D}^*)$ as follows. $\mathcal{E}^*(pk, m; R_1 \,\|\, R_2)$ returns $(\mathcal{E}(pk, m; R_1) \,\|\, R_2)$, where $|R_2| = k$, meaning $\mathcal{E}^*$ uses $r + k$ coins; and $\mathcal{D}^*(sk, C \,\|\, R)$ returns $\mathcal{D}(sk, C)$. It easy to see that $\mathcal{PE}^*$ is ind-cpa and $\mathrm{mc}_{\mathcal{PE}^*}(k) \leq 2^{-k}$.

We now state the security result.

Theorem 4.3 Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme with max-collision probability $\mathrm{mc}_{\mathcal{PE}}(k)$ and let H be a hash function. Let $\mathcal{DPE} = (\mathcal{K}, \mathcal{DE}, \mathcal{DD})$ be the deterministic encryption scheme defined according to Construction 4.2. Then the ESE scheme $\mathcal{DPE} = (\mathcal{K}, \mathcal{DE}, \mathcal{DD})$ is priv-cca in the RO model if $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is ind-cpa. More precisely, let $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ be a RPT privacy adversary against $\mathcal{DPE}$ having message space min-entropy function $\mathrm{me}_A(\cdot)$ and making at most $q_h(\cdot)$ queries to its hash oracle and $q_d(\cdot)$ queries to its decryption oracle. Then there exists an RPT ind-cpa-adversary B against $\mathcal{PE}$ such that for all k,

$$\mathbf{Adv}^{\text{priv-cca}}_{\mathcal{DPE},A}(k) \;\le\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PE},B}(k) + \frac{2q_h(k)}{2^{\mathrm{me}_A(k)}} + 2q_d(k)\mathrm{mc}_{\mathcal{PE}}(k)\ . \tag{2}$$

The proof is in Appendix C. The weaker result about priv-cpa security analogous to the above, but with the last additive term in Equation (2) omitted, can be stated and proved, but we omit this since the max-collision probability requirement is satisfied by the practical ind-cpa schemes anyway, meaning the stronger priv-cca can be achieved under basically the same assumptions.

As a secure public-key deterministic encryption scheme is simply a family of (injective) trapdoor functions with some extra security properties, we observe that by a result of Gertner et al. [22] showing that in the standard model there is no black-box reduction from trapdoor functions to trapdoor predicates (i.e., randomized trapdoor functions hard to invert on a random message from a small message space), it will be hard to build a secure public-key deterministic encryption scheme based solely on a secure standard one without random oracles. However, it does not preclude building a secure deterministic one using other primitives not implied by secure public-key encryption, such as a collision-resistant family of hash functions, though we do not imply this is actually possible.

4.3 RSA-DOAEP, A Length-Preserving ESE scheme

It is desirable to minimize the number of bits transmitted over the network, for example, when users have a low-bandwidth connection to the outsourced database. Our final scheme is optimal in this regard, in that it is length-preserving. (Note that it is a standard fact that this cannot be achieved using ind-cpa encryption, a further draw for using this scheme in any application requiring length-preserving encryption and where message space min-entropy is large.) The scheme, which we term RSA-DOAEP ("D" for deterministic), derives from RSA-OAEP [8, 20]. The high-level design is also reminiscent of the more-recent "3-round OAEP" scheme of Phan and Pointcheval [46, 45], which achieves ind-cca security in the RO model without redundancy (i.e., appending zeros to the plaintext).

We stress that for the previous two ESE constructions the underlying scheme can be a hybrid encryption scheme, so they can be used to efficiently encrypt messages of various lengths. For RSA-DOAEP, we also show how to efficiently encrypt messages of arbitrary length without making use of any hybrid scheme (see Case 2 of Theorem 4.5). A hybrid encryption can in some sense never be length-preserving because an (encrypted) symmetric key is included with the ciphertext; thus RSA-DOAEP further saves on bandwidth in this respect.

Trapdoor permutations. The security of our scheme assumes the existence of one-way trapdoor permutations, the basics of which we recall here. A *trapdoor-permutation generator* with associated security parameter k is an RPTA $\mathcal{F}$ that on input 1^k returns a pair (f, f^{-1}), where $f\colon \{0,1\}^{n(k)} \to \{0,1\}^{n(k)}$ for some polynomial n is an efficient encoding of a permutation and f^{-1} (called the "trapdoor") is an efficient encoding of its inverse. An *inverter* I against $\mathcal{F}$ is an RPTA that takes

as input $f, f(x)$ and tries to compute x. We say that $\mathcal{F}$ is *one-way* if for every RPTA I the function $\mathbf{Adv}^{\mathrm{powf}}_{\mathcal{F},I}(\cdot)$ defined as:

$$\Pr\left[\ (f, f^{-1}) \xleftarrow{\$} \mathcal{F}(1^k)\ ;\ x \xleftarrow{\$} \{0,1\}^{n(k)}\ ;\ x' \xleftarrow{\$} I(f, f(x))\ :\ x = x'\ \right]$$

is negligible. We will use the well-studied RSA trapdoor permutation generator $\mathcal{F}_{\mathrm{RSA}}$ [49], which is widely assumed as one-way. ▌

For simplicity, we will assume all messages have length $n(k)$ for some polynomial n. In practice, to maintain the length-preserving property, one can bypass this assumption by using a "variable-length" instantiation for the ROs, meaning the length of its output depends on the input. Such an instantiation is easily obtained, for example, via the common instantiation heuristic first suggested in [7], in which one obtains, say, k random bits of output for a given string x by computing the string $H(K, x \,\|\, \langle 1\rangle) \,\|\, H(K, x \,\|\, \langle 2\rangle) \,\|\, \ldots$ to sufficient length, where H is a "good" hash function with key K derived from the public key and $\langle k\rangle$ denotes $k \in \mathbb{N}$ encoded as a binary string in the natural way, and truncating the result to any desired length. To apply our security analysis in this case, one could model the scheme by (mentally) fixing the message length to the shortest allowable length, the worst case from a security standpoint.

Construction 4.4 [RSA-DOAEP] Let $\mathcal{F}_{\mathrm{RSA}}$ be the RSA trapdoor-permutation generator. The scheme is parameterized by the length functions $k_0(\cdot), k_1(\cdot)$ satisfying and $n(k) > 2k_0(k)$ and $n(k) \geq k_1(k)$. The key-generation algorithm of the efficiently-searchable encryption scheme RSA-DOAEP on input 1^k runs $\mathcal{F}_{\mathrm{RSA}}$ on the same input and returns its output, meaning it returns f as the public key and f^{-1} as the secret key. The encryption and decryption algorithms have access to independent oracles $H_1, H_2 \colon \{0,1\}^{n(k)-k_0(k)} \to \{0,1\}^{k_0(k)}$ and $R \colon \{0,1\}^{k_0(k)} \to \{0,1\}^{n(k)-k_0(k)}$, and are defined as follows:

Algorithm $\mathcal{E}_f^{H_1,H_2,R}(m)$
Parse m as $m_l \,\|\, m_r$,
where $|m_r| = n(k) - k_0(k)$
$S_0 \leftarrow H_1(m_r) \oplus m_l$; $T_0 \leftarrow R(S_0) \oplus m_r$
$S_1 \leftarrow H_2(T_0) \oplus S_0$
Parse $S_1 \,\|\, T_0$ as $X_1 \,\|\, X_2$,
where $|X_2| = k_1(k)$
$Y \leftarrow X_1 \,\|\, f(X_2)$
Return Y

Algorithm $\mathcal{D}_{f^{-1}}^{H_1,H_2,R}(Y)$
Parse Y as $X_1 \,\|\, Y'$,
where $|Y'| = k_1(k)$
$X \leftarrow X_1 \,\|\, f^{-1}(Y')$
Parse X as $S_1 \,\|\, T_0$
where $|S_1| = k_0(k)$ and $|T_0| = n(k) - k_0(k)$
$S_0 \leftarrow H_2(T_0) \oplus S_1$; $m_r \leftarrow R(S_0) \oplus T_0$
$m_l \leftarrow H_1(m_r) \oplus S_0$
Return $m_l \,\|\, m_r$

Here now is the security result.

Theorem 4.5 Let $\mathcal{F}_{\mathrm{RSA}}$ be the RSA trapdoor permutation generator. Let $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ be an RPT privacy adversary against RSA-DOAEP having message space min-entropy function $\mathrm{me}_A(\cdot)$ and making at most $q_{H_i}(\cdot)$ queries to oracle H_i for $i \in \{1,2\}$ and at most $q_R(\cdot)$ queries to oracle R. We divide the result into two cases:

- Case 1: $n(k) - k_0(k) < k_1(k) \leq n(k)$. Then there exists an inverter I against $\mathcal{F}_{\mathrm{RSA}}$ such that

for all k,

$$\begin{aligned}\mathbf{Adv}^{\text{priv-cpa}}_{\mathsf{RSA\text{-}DOAEP},A}(k) &\leq q_{H_2}(k)\sqrt{\mathbf{Adv}^{\text{owf}}_{\mathcal{F}_{\text{RSA}},I}(k) + 2^{4k_0(k)-2k_1(k)+10}} - 2^{2k_0(k)-k_1(k)+5} + \frac{2q_R(k)}{2^{k_0(k)}} \\ &+ \frac{2q_{H_1}(k)q_R(k)}{2^{\text{me}_A(k)}} \ . \end{aligned} \tag{3}$$

- Case 2: $k_1(k) \leq n(k) - k_0(k) \leq n(k)$. Then there exists an inverter I against $\mathcal{F}_{\text{RSA}}$ such that for all k,

$$\mathbf{Adv}^{\text{priv-cpa}}_{\mathsf{RSA\text{-}DOAEP},A}(k) \ \leq \ \mathbf{Adv}^{\text{owf}}_{\mathcal{F}_{\text{RSA}},I}(k) + \frac{2q_R(k)}{2^{k_0(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\text{me}_A(k)}} \ . \tag{4}$$

The proof is in Appendix D. There we provide an intuition why we are able to achieve only priv-cpa security of the scheme. Nevertheless, when combined properly with a digital signature scheme, RSA-DOAEP in fact achieves priv-cca security, as we shall see in the following section.

To conclude this section, let us deduce how to set the length functions $k_0(k), k_1(k)$ in order to achieve good concrete security with respect to RSA for a particular application. Let us say that 1024-bit RSA takes on the order of 2^{80} computations to invert [36], which we assume infeasible. So we set $k_1(k)$ to 1024. If the message length $n(k)$ is at least 1104 bits, then evidently to achieve the same level of security for DOAEP we should set $k_0(k)$ as high as needed under the inequalities $k_1(k) \leq n(k) - k_0(k) \leq n(k)$ and $n(k) > 2k_0(k)$, so to 80 bits. Case 2 of the theorem then shows that A still needs make about 2^{80} computations to break RSA-DOAEP assuming, as for our previous constructions, that message space min-entropy is about 80 bits as well.

If instead $n(k)$ is between 1024 and 1103 bits, we are forced into Case 1 of the theorem. The expression under the radical in (3) increases super-linearly as a function of $2^{k_0(k)}$, so we see that to maximize concrete security $k_0(k)$ should be set as low as possible while prohibiting a brute-force attack, i.e., again to 80 bits. But here the theorem only guarantees that A must make on the order of 2^{40} computations to break RSA-DOAEP. However, this situation is completely analogous to the state-of-the-art for RSA-OAEP [20], and, following the discussion in Section 3.3 of [48], we stress any known algorithm satisfying the hypothesis in Lemma D.1 is simply an algorithm to invert RSA. Indeed, RSA-OAEP in the analogous case is still trusted widely in practice.

5 Efficiently-Searchable Signcryption

Signcryption [5] is an asymmetric-setting primitive where the sender has a signing key and the receiver's public key, designed to simultaneously protect the receiver's privacy and the sender's authenticity. In this section, we obtain analogues of some results of [5], which show that an "encrypt-then-sign" (ETS) construction of ESE and digital signatures can provably achieve this goal. The motivation is two-fold: for one, it also addresses the issue of data authenticity with respect to its origin and not having been modified over the network or server-side. Authentication mechanisms in DAS have been recently studied for this purpose in [41, 40, 39] with the focus on more flexible (e.g., aggregation) queries and experimental results. Furthermore, we show that an ETS construction can actually be used to boost security of ESE; in particular, RSA-DOAEP, shown in the last section to achieve priv-cpa security as a stand-alone scheme, in fact achieves priv-cca in applications that require authentication of data anyway.

We recall the standard syntax and security definitions for digital signature schemes in Appendix A. Next we formalize the "encrypt-then-sign" construction of an efficiently-searchable signcryption scheme.

Construction 5.1 [Encrypt-then-sign efficiently-searchable signcyption] Let $\mathcal{SPE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be an ESE scheme and let $\mathcal{DS} = (\mathcal{K}_S, \mathcal{S}, \mathcal{V})$ be a digital signature scheme scheme. We define the *"encrypt-then-sign" (ETS) efficiently-searchable signcryption scheme*, which we denote $\mathcal{ETS}^{\mathcal{SPE}}_{\mathcal{DS}} = (\mathcal{K}_{\mathcal{ES}}, \mathcal{ES}, \mathcal{VD})$, as follows:

Algorithm $\mathcal{K}_{ES}(1^k)$
$(pk_E, sk_E) \xleftarrow{\$} \mathcal{K}_E(1^k)$
$(pk_S, sk_S) \xleftarrow{\$} \mathcal{K}_S(1^k)$
Return $((pk_E, sk_S), (sk_E, pk_S))$

Algorithm $\mathcal{ES}((pk_E, sk_S), m)$
$c \xleftarrow{\$} \mathcal{E}(pk_E, m)$
$\sigma \xleftarrow{\$} \mathcal{S}(sk_S, c)$
Return (c, σ)

Algorithm $\mathcal{VD}((sk_E, pk_S), (c, \sigma))$
$b \leftarrow \mathcal{V}(pk_S, c, \sigma)$
If $b = 0$ then return $\perp$
$m \leftarrow \mathcal{D}(sk_E, c)$
Return m

The idea behind a notion of security for the above scheme is to evaluate the "induced" signature and efficiently-searchable encryption schemes under their respective appropriate definitions, uf-cma and priv-cca. Namely, the induced encryption scheme is simply $(\mathcal{K}_{ES}, \mathcal{ES}, \mathcal{VD})$ (it is easy to check that this meets Definition 3.1), and the induced signature scheme is $(\mathcal{K}_{ES}, \mathcal{ES}, \mathcal{VD}')$, where the verification algorithm $\mathcal{VD}'$ runs $\mathcal{VD}'$ on its input and returns 1 just when the output is not $\perp$. The formal security definitions in [5] are trivially extended to the ESE setting and we do not provide the details here.

One noteworthy point is that, as discussed in [5], we do not allow the adversaries in the experiments with the induced schemes access to the secret signature key despite the fact that it is a component of the "public" signcryption key. On the one hand, this prevents the adversary in the uf-cma experiment on the induced signature scheme from succeeding trivially by signing a message itself, and the adversary in a priv-cca experiment on the induced ESE scheme from likewise resigning the first component of the challenge ciphertext (c, σ) and then decrypting it. On the other, it means that if one wants to prevent legitimate signers (e.g., the nurses) from forging messages that appear to come from other legitimate signers, then all legitimate signers must have their own key-pair for signatures.

Here then is the precise security result for the above construction.

Theorem 5.2 Let $\mathcal{DS}$ be a digital signature scheme that is uf-cma, and $\mathcal{SPE}$ be an ESE scheme that is priv-cpa. Then the induced signature scheme of the efficiently-searchable signcryption scheme $\mathcal{ETS}^{\mathcal{SPE}}_{\mathcal{DS}}$ is uf-cma and the induced efficiently-searchable encryption scheme of $\mathcal{ETS}^{\mathcal{SPE}}_{\mathcal{DS}}$ is priv-cca secure.

The proof of the first part is identical to part (induced uf-cma security) of the proof of Theorem 1 in [5], hence omitted. The proof of the second part (induced priv-cca security) is almost the same as the proof of Theorem 2 in [5], but we use a different security definition for encryption. Thus

there is an obvious difference in the functionality of the adversaries, but oracle simulation remains identical. While we omit the formal details, the intuition from [5] still applies: if the adversary can make a valid query (c, σ) to its decryption oracle without having first received it from its encryption oracle (in which case the simulator knows the underlying message), then the query constitutes a valid forgery against $\mathcal{DS}$.

As promised, we observe that Theorem 5.2 together with Theorem 4.5 in the previous section implies that DOAEP achieves priv-cca security when used with a uf-cma secure signature scheme in the ETS construction of an efficiently-searchable signcryption scheme.

6 Conclusions

In this paper, we formally developed efficiently-searchable asymmetric encryption (ESE) schemes as tools to support practical exact-match query processing on encrypted databases. We defined asymmetric ESE and its security and provided several constructions with provable security (in the RO model) and various efficiency properties. In particular, we saw how the essential "weakness" of deterministic encryption for general use, namely its injectivity, actually makes it a useful primitive in this setting. We also discussed how to simultaneously achieve privacy and authenticity in ESE schemes. We believe that our work will both help researchers and developers in the area of outsourced databases and be of independent interest for other applications where our deterministic asymmetric encryption schemes can be used securely. Open problems include studying symmetric-key searchable encryption, answering more flexible queries, finding asymmetric ESE schemes secure in the standard model and identity-based ESE schemes.

7 Acknowledgements

We thank Jun Li and Ed Omiecinski for introducing the problem to us and Brian Cooper for useful comments.

References

[1] The final HIPAA security rule. *Federal Register. Available at* `http://www.hipaadvisory.com/regs/finalsecurity/index.htm`, 2003.

[2] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In V. Shoup, editor, *Crypto 2005*, Lecture Notes in Computer Science. Springer.

[3] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu. Two can keep a secret: A distributed architecture for secure database services. In *CIDR*, pages 186–199, 2005.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2004. ACM Press.

[5] J.-H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 83–107, London, UK, 2002. Springer-Verlag.

[6] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. *Cryptology ePrint Archive, Report 2005/151.*

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

[8] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, pages 92–111, 1994.

[9] M. Bellare and P. Rogaway. The game-playing technique and its application to triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004.

[10] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, New York, NY, USA, 2005. ACM Press.

[11] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[12] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 442–455, 2005.

[13] S. Chawla, C. Dwork, F.McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In Kilian [33], pages 363–385.

[14] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS: IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 41–50, 1995.

[15] C. Clifton, M. Kantarcioglu, and J. Vaidya. Defining privacy for data mining. National Science Foundation Workshop on Next Generation Data Mining, H. Kargupta, A. Joshi, and K. Sivakumar, Eds., Baltimore, MD, Nov. 1-3 2002, pp. 126–133. 21, 2002.

[16] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Computing range queries on obfuscated data. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 2004.

[17] E. Damiani, S. De Capitani Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 93–102, New York, NY, USA, 2003. ACM Press.

[18] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, New York, NY, USA, 2003. ACM Press.

[19] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In Kilian [33], pages 303–324.

[20] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.

[21] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

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

[23] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[24] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *STOC*, New York, May 1987. ACM Press.

[25] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[26] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Proc. of the 2004 Applied Cryptography and Network Security Conference*, pages 31–45. Lecture Notes in Computer Science 3089, 2004.

[27] H. Hacigümüs, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227, New York, NY, USA, 2002. ACM Press.

[28] H. Hacigümüs, B. R. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In Y. Lee, J. Li, K.-Y. Whang, and D. Lee, editors, *DASFAA*, volume 2973 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2004.

[29] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In M. A. Nascimento, M. Tamer Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 720–731. Morgan Kaufmann, 2004.

[30] B. R. Iyer, S. Mehrotra, E. Mykletun, G. Tsudik, and Y. Wu. A framework for efficient storage security in rdbms. In E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, and E. Ferrari, editors, *EDBT*, volume 2992 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2004.

[31] M. Kantracioglu and C. Clifton. Security issues in querying encrypted data. In *DBSec*, 2005.

[32] A. Kiayias and A. Mitrofanova. Testing disjointness of private datasets. In *Financial Cryptography '05*, 2005.

[33] J. Kilian, editor. *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*. Springer, 2005.

[34] L. Kissner and D. Song. Private and threshold set-intersection. In V. Shoup, editor, *CRYPTO 2005*, Lecture Notes in Computer Science, 2005.

[35] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[36] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. In *PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, pages 446–465, London, UK, 2000. Springer-Verlag.

[37] J. Li and E. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In *DBSec*, pages 69–83, 2005.

[38] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM J. Comput.*, 17(2):412–426, 1988.

[39] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *NDSS*, 2004.

[40] M. Narasimha and G. Tsudik. DSAC: integrity for outsourced databases with signature aggregation and chaining. In *CIKM*, pages 235–236, 2005.

[41] M. Narasimha and G. Tsudik. Authentication of outsourced databases using signature aggregation and chaining. In *DASFAA*, pages 420–436, 2006.

[42] A. Narayanan and V. Shmatikov. Obfuscated databases and group privacy. In *CCS '05*. ACM Press, 2005.

[43] R. Ostrovsky and W. E. Skeith III. Private searching on streaming data. In V. Shoup, editor, *CRYPTO 2005*, volume 3621, 2000.

[44] G. Özsoyoglu, D. A. Singer, and S. S. Chung. Anti-tamper databases: Querying encrypted databases. In S. De Capitani di Vimercati, I. Ray, and I. Ray, editors, *DBSec*, pages 133–146. Kluwer, 2003.

[45] D. H. Phan and D. Pointcheval. Chosen-ciphertext security without redundancy. In *ASIACRYPT*, pages 1–18, 2003.

[46] D. H. Phan and D. Pointcheval. OAEP 3-round: A generic and secure asymmetric encryption padding. In *ASIACRYPT*, pages 63–77, 2004.

[47] B. Pinkas. Cryptographic techniques for privacypreserving data mining. SIGKDD Explorations, 4(2), Dec. 2002., 2002.

[48] D. Pointcheval. How to encrypt properly with rsa. *RSA Laboratories' CryptoBytes*, 5(1):9–19, Winter/Spring 2002.

[49] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining public-key cryptosystems and digital signatures. Technical Report MIT/LCS/TM-82, 1977.

[50] R. Sion. Query execution assurance for outsourced databases. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 601–612. VLDB Endowment, 2005.

[51] Arsenal Digital Solutions. Top 10 reasons to outsource remote data protection. *Available at* `http://www.arsenaldigital.com/services/remote_data_protection.htm`.

[52] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[53] A. Yao. Protocols for secure computations. In *Twenty-third annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE, 1982.

A Basics of Asymmetric Encryption and Digital Signatures

We recall the standard syntax and security definitions for asymmetric (aka. public-key) encryption and digital signature schemes.

ASYMMETRIC ENCRYPTION.

Definition A.1 [Public-key encryption scheme] An *asymmetric (aka public-key) encryption scheme* $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with associated security parameter $k \in \mathbb{N}$ and the *message space* $\mathrm{MsgSp}(k)$ that can also depend on some public parameters, e.g. a group description, consists of three algorithms:

- The *key generation* RPTA $\mathcal{K}$ takes as input the security parameter and returns a pair (pk, sk) consisting of a public key and a corresponding secret key; we write $(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k)$.
- The *encryption* RPTA $\mathcal{E}$ takes input the public key pk and a plaintext $m \in \mathrm{MsgSp}(k)$ and returns a ciphertext; we write $C \stackrel{\$}{\leftarrow} \mathcal{E}(pk, m)$ or $C \leftarrow \mathcal{E}(pk, m; R)$. If $C = \mathcal{E}(pk, m, R)$ for some coins R then we say C is a *valid* ciphertext for m under pk.
- The *decryption* PTA $\mathcal{D}$ takes the secret key sk and a ciphertext C to return the corresponding plaintext or a special symbol $\perp$ indicating that the ciphertext was invalid; we write $m \leftarrow \mathcal{D}(sk, C)$ (or $\perp \leftarrow \mathcal{D}(sk, C)$.)

Consistency: we require that $\mathcal{D}(sk, (\mathcal{E}(pk, m)) = m$ for all messages m.

Definition A.2 [Security of encryption schemes] Let $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let LR be the oracle that on input m_0, m_1, b returns m_b. For atk $\in \{\text{cpa}, \text{cca}\}$, adversary B_{atk} and $b \in \{0, 1\}$ define the experiment:

$$\begin{array}{l}\textbf{Experiment } \mathbf{Exp}^{\text{ind-atk-b}}_{\mathcal{PE},B_{\text{atk}}}(k)\\ (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\\ d \xleftarrow{\$} B_{\text{atk}}^{\mathcal{E}(pk,\text{LR}(\cdot,\cdot,b)),\mathcal{O}(\cdot)}(pk)\\ \text{Return } d\end{array}$$

where $\mathcal{O}(sk, \cdot) = \mathcal{D}(sk, \cdot)$ if atk = cca and is the empty oracle otherwise. We call B_{atk} an *ind-atk adversary* if it makes at most one query (m_0, m_1) to its left-or-right encryption oracle with $|m_0| = |m_1|$, and does not query the challenge ciphertext to its oracle. The *advantage* of a ind-atk adversary B_{atk} is defined for every k as follows:

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathcal{PE},B_{\text{atk}}}(k) \quad = \quad \Pr\left[\,\mathbf{Exp}^{\text{ind-atk-0}}_{\mathcal{PE},B_{\text{atk}}}(k) = 0\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{ind-atk-1}}_{\mathcal{PE},B_{\text{atk}}}(k) = 0\,\right].$$

The scheme $\mathcal{PE}$ is said to be *secure against chosen-plaintext attack* or *ind-cpa* (resp. *chosen-ciphertext attack* or *ind-cca*) if for every RPTA B_{atk} the function $\mathbf{Adv}^{\text{ind-atk}}_{\mathcal{PE},B_{\text{atk}}}(\cdot)$ is negligible in k, where atk = cpa in the former case and atk = cca in the latter. ▌

Digital Signatures.

Definition A.3 [Digital signature scheme] A *digital signature scheme* $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ with associated security parameter $k \in \mathbb{N}$ a *message space* $\text{MsgSp}(k)$ that can also depend on some public parameters, e.g. a group description, consists of three algorithms:

- The *key generation* RPTA $\mathcal{K}$ takes as input the security parameter and returns a pair (pk, sk) consisting of a public key and a corresponding secret key; we write $(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$.
- The *signature* RPTA $\mathcal{S}$ takes input the public key pk and a plaintext $m \in \text{MsgSp}(k)$ and returns a signature for m; we write $\sigma \xleftarrow{\$} \mathcal{E}(pk, m)$.
- The *verification* PTA $\mathcal{V}$ takes the secret key sk, a message m, and a signature σ to return a bit $b \in \{0, 1\}$. We write $b \leftarrow \mathcal{V}(sk, m, \sigma)$. In the case that the above b is 1 we say that σ is a valid signature for m under pk.

Consistency: we require that $\mathcal{V}(sk, m, (\mathcal{S}(pk, m)) = 1$ for all $m \in \text{MsgSp}(k)$

Definition A.4 [Security of signature schemes] Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a digital signature scheme scheme. For an adversary B define the experiment:

$$\begin{array}{l}\textbf{Experiment } \mathbf{Exp}^{\text{uf-cma}}_{\mathcal{DS},B}(k)\\ (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\\ (m, \sigma) \xleftarrow{\$} B^{\mathcal{S}(sk,\cdot)}(pk)\\ \text{Return } \mathcal{V}(pk, m, \sigma)\end{array}$$

Adversary $B^{\mathcal{E}(pk,\mathrm{LR}(\cdot,\cdot,b)),\mathcal{D}(sk,\cdot)}(pk)$
$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, m_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$C \leftarrow \mathcal{E}(pk, \mathrm{LR}(m_0, m_1, b))$
$h \xleftarrow{\$} \{0,1\}^{l(k)}$
Run A_{g} on input $pk, C \,\|\, h$, replying to its oracle queries as follows:
On hash query x:
If $x = m_0$ then
If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
If $x = m_1$ then
If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
If $H[x]$ is undefined
then $H[x] \xleftarrow{\$} \{0,1\}^{l(k)}$
Return $H[x]$
On decryption query y:
Parse y as $H_y \,\|\, C_y$
If $C_y = C$ then return $\perp$
$m \leftarrow \mathcal{D}(sk, C_y)$
If $m = \perp$ then Return $\perp$
If $H[m]$ is undefined then
$H[m] \xleftarrow{\$} \{0,1\}^{l(k)}$
If $H[m] = H_y$ then Return m
Else Return $\perp$
Let g be the output of A_{g}
If $\mathsf{zer} = \mathsf{true}$ then $d \leftarrow 0$
Else If $\mathsf{one} = \mathsf{true}$ then $d \leftarrow 1$
Else If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
Return d

Figure 1: Ind-cca adversary B for proof of Theorem 4.1.

We call B an *uf-cma adversary* if it does not query m to its signing oracle. The *advantage* of a uf-cma adversary B is defined for every k as follows:

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},B}(k) \quad = \quad \Pr\left[\, \mathbf{Exp}^{\text{uf-cma}}_{\mathcal{DS},B}(k) = 1 \,\right] .$$

The scheme $\mathcal{DS}$ is said to be *unforgeable against chosen-message attack* or *uf-cma* if for every RPTA B the function $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS},B}(\cdot)$ is negligible in k. ▌

B Proof of Theorem 4.1

We prove the theorem here for atk equal to cca. For atk equal to cpa, the decryption oracle in the games and adversary should be removed.

procedure Initialize All games
$b \xleftarrow{\$} \{0,1\}$
$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, m_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $R \xleftarrow{\$} \{0,1\}^{l(k)}$
$H_0, H_1 \xleftarrow{\$} \{0,1\}^{l(k)}$
$C \leftarrow \mathcal{E}(pk, m_b)$
Return $pk, H_b \,\|\, C$

On hash query x: Games G_1–G_4/G_5
If $H[x]$ is undefined then
  $H[x] \xleftarrow{\$} \{0,1\}^{l(k)}$
If $x = m_0$ then
  If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
  $\boxed{H[x] \leftarrow H_0}$
If $x = m_1$ then
  If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
  $\boxed{H[x] \leftarrow H_1}$
Return $H[x]$

On decryption query y: All games
Parse y as $H_y \,\|\, C_y$
If $C_y = C$ then Return $\perp$
$m \leftarrow \mathcal{D}(sk, C_y)$
If $m = \perp$ then Return $\perp$
If $H[m]$ is undefined then
  $H[m] \xleftarrow{\$} \{0,1\}^{l(k)}$
If $H[m] = H_y$ then Return m
Else Return $\perp$

---

procedure Finalize(g) Game G_1
If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
Return d

---

procedure Finalize(g) Games G_2/G_3
If $\mathsf{zer} = \mathsf{true}$ then $d \leftarrow 0$
Else If $\mathsf{one} = \mathsf{true}$ then $d \leftarrow 1$
  Else If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
If $(b = 1 \wedge \mathsf{zer} = \mathsf{true}) \vee (b = 0 \wedge \mathsf{one} = \mathsf{true})$
  then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{d \leftarrow b}$
Return d

---

procedure Finalize(g) Games G_4, G_5
If $\mathsf{zer} = \mathsf{true}$ then $d \leftarrow 0$
Else If $\mathsf{one} = \mathsf{true}$ then $d \leftarrow 1$
  Else If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
Return d

Figure 2: Games for the proof of Theorem 4.1. All 5 games have the same Initialize procedure and procedure to respond to decryption queries. The procedure to respond to hash queries includes the boxed statements for games G_1, G_2, G_3, G_4 and excludes them for G_5. The Finalize procedure of Game G_2 includes the boxed statement while that of G_3 does not. The Finalize procedures of G_4, G_5 are the same.

Ind-cca adversary B is depicted in Figure 1. The analysis used to establish (1) uses the game-playing technique in the style of [9]. In particular, we consider the games depicted in Figure 2. Let us begin by recalling some game-related language and conventions from [9] that we use here.

A game consists of an Initialize procedure, procedures that respond to adversary oracle queries (in this case, two: one to respond to hash oracle queries and one to respond to decryption oracle queries) and a Finalize procedure. Figure 2 presents a total of five games. All have the same

Initialize procedure and procedure to respond to decryption queries. The first four have the same procedure to respond to hash oracle queries, this being the one shown when the boxed statements are included, while for the last game, namely G_5, the procedure omits the boxed statements. The Finalize procedures are as shown, with those of Games G_2, G_3 the same except that the former includes the boxed code statement while the latter does not, and those of G_4, G_5 being the same. We will be executing $A_{\rm g}$ with each of these games. The execution of $A_{\rm g}$ with G_i is determined as follows. First, the Initialize procedure executes, and its outputs pk, C, as given by the Return statement, are passed as inputs to $A_{\rm g}$. Now the latter executes, its hash and decryption oracle queries being answered by the procedures for this purpose associated to G_i. The output g of $A_{\rm g}$ becomes the input to the Finalize procedure of G_i. The output of the game is whatever is returned by the Finalize procedure. We let "$G_i^{A_{\rm g}} \Rightarrow b$" denote the event that the output of Game G_i, when executed with $A_{\rm g}$, is the bit b chosen at random in the Initialize procxedure.

Both for the games and for the adversary in Figure 3, we adopt the convention that boolean variables like $\mathsf{bad}, \mathsf{zer}, \mathsf{one}$ are automatically initialized to false and arrays like $H[\cdot]$ begin everywhere undefined.

Equation (1) follows from the following sequence of inequalities which we will justify below:

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{priv-cca}}_{\mathcal{HPE},A}(k) &= \Pr\left[\, G_1^{A_{\rm g}} \Rightarrow b \,\right] && (5)\\
&\le \Pr\left[\, G_2^{A_{\rm g}} \Rightarrow b \,\right] && (6)\\
&\le \Pr\left[\, G_3^{A_{\rm g}} \Rightarrow b \,\right] + \Pr[G_3^{A_{\rm g}} \text{ sets } \mathsf{bad}\,] && (7)\\
&\le \Pr\left[\, G_3^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{\mathrm{me}_A(k)}} && (8)\\
&= \Pr\left[\, G_4^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{\mathrm{me}_A(k)}} && (9)\\
&= \Pr\left[\, G_5^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{\mathrm{me}_A(k)}} && (10)\\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PE},B}(k) + \frac{q_h(k)}{2^{\mathrm{me}_A(k)}}\ . && (11)
\end{aligned}$$

An advantage that, defined in Definition 3.1 as the difference in the probabilities that two experiments return 1, is, as usual, also equal to $2p - 1$ where p is the probability that the adversary correctly guesses the challenge bit b in a game where we pick b at random and run the adversary with the first experiment if $b = 1$ and the second if $b = 0$. Game G_1 is exactly this game written in a convenient way. It makes the choices of $H(m_0), H(m_1)$ upfront, and also sets a few flags, but the flags do not influence the game output. We have justified (5).

The Finalize procedure of Game G_2 begins by defining its output bit d in certain ways depending on the flags $\mathsf{zer}, \mathsf{one}$ if either of these are true, and otherwise defining it as in G_1. However, in case the value of d set by the first two "If" statements is wrong, meaning not equal to b, the third "If" statement corrects, setting d to b. The net result is that in the cases that G_2 assigns d differently from G_1, the assignment made by G_2 is correct, meaning equal to b. Additionally G_2 sets a flag bad but this does not influence its choice of d. So the probability that the output of $A_{\rm g}$ equals b can only go up. We have justified (6).

Games G_2, G_3 differ only in statements that follow the setting of bad, meaning are, in the terminology of [9], identical-until-bad games. The Fundamental Lemma of Game Playing [9] thus

applies to justify (14). The probability that $A_{\rm g}$ makes hash query m_{1-b} when executed with G_3 is at most $q_h(k)/2^{{\rm me}_A(k)}$ because $A_{\rm g}$ gets no information about m_{1-b}. This justifies (15). Since the third "If" statement in G_3 only sets a flag that does not influence the game output, dropping this entire statement results in an equivalent game that we have called G_4. This justifies (9).

As in the proof of the Fundamental Lemma in [9], we can consider a common finite space of coins associated to the executions of $A_{\rm g}$ with either G_4 or G_5. Consider the execution of $A_{\rm g}$ with G_4 when a particular coin sequence is chosen at random from this set. One of the boxed statements in the procedure to respond to a hash query can be executed only if either $\mathsf{one} = \mathsf{true}$ or $\mathsf{zer} = \mathsf{true}$, due to the "If" statements that precede the boxed statements. However, once one of these flags is set to true, the output of the Finalize procedure is determined. (Nothing further that happens in the execution can change it. Note we use here that at most one of $\mathsf{zer}, \mathsf{one}$ can be true, never both, and once one of them is true, it never becomes false.) This means that the boxed statements have no effect on the output of the game, and eliminating them results in the equivalent game G_5. We have justified (11).

Now (11) is easy to see by comparing the code of B to that of Game G_5 and taking into account the definition of the advantage of B. ∎

C Proof of Theorem 4.3

The proof utilizes the game-playing technique in the style of [9]. We refer the reader to the beginning of the proof of Theorem 4.1 in the previous appendix for a summary of the fundamentals of the technique.

Ind-cpa adversary B is depicted in Figure 3. Equation (2) follows from the following sequence of inequalities which we will justify below:

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{priv-cca}}_{\mathcal{DPE},A}(k) &= \Pr\left[\, G_1^{A_{\rm g}} \Rightarrow b \,\right] && (12)\\
&\le \Pr\left[\, G_2^{A_{\rm g}} \Rightarrow b \,\right] && (13)\\
\le \Pr\left[\, G_3^{A_{\rm g}} \Rightarrow b \,\right] + \Pr[G_3^{A_{\rm g}} \text{ sets } \mathsf{bad}_0\,] && (14)\\
\le \Pr\left[\, G_3^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{{\rm me}_A(k)}} && (15)\\
= \Pr\left[\, G_4^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{{\rm me}_A(k)}} && (16)\\
= \Pr\left[\, G_5^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{{\rm me}_A(k)}} && (17)\\
\le \Pr\left[\, G_6^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{{\rm me}_A(k)}} + \Pr[G_6^{A_{\rm g}} \text{ sets } \mathsf{bad}_1\,] && (18)\\
\le \Pr\left[\, G_6^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{{\rm me}_A(k)}} + q_d(k){\rm mc}_{\mathcal{PE}}(k) && (19)\\
= \Pr\left[\, G_7^{A_{\rm g}} \Rightarrow b \,\right] + \frac{q_h(k)}{2^{{\rm me}_A(k)}} + q_d(k){\rm mc}_{\mathcal{PE}}(k) && (20)\\
= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PE},B}(k) + \frac{q_h(k)}{2^{{\rm me}_A(k)}} + q_d(k){\rm mc}_{\mathcal{PE}}(k)\,. && (21)
\end{aligned}$$

Adversary $B^{\mathcal{E}(pk,\mathrm{LR}(\cdot,\cdot,b))}(pk)$
$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, m_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$C \xleftarrow{\$} \mathcal{E}(pk, \mathrm{LR}(m_0, m_1, b))$
Run A_{g} on input pk, C, replying to its oracle queries as follows:
  On hash query x:
  If $H[x]$ is undefined then
    $H[x] \xleftarrow{\$} \{0,1\}^{l(k)}$
  $E[x] \leftarrow \mathcal{E}(pk, x; H[x])$
  If $x = m_0$ then
    If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
  If $x = m_1$ then
    If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
  Return $H[x]$
  On decryption query y:
  If $\exists x_y$ such that $E[x_y] = y$ then
    Return x_y
  Else Return $\perp$
Let g be the output of A_{g}
If $\mathsf{zer} = \mathsf{true}$ then $d \leftarrow 0$
Else If $\mathsf{one} = \mathsf{true}$ then $d \leftarrow 1$
    Else If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
Return d

Figure 3: Ind-cpa adversary B for proof of Theorem 4.3.

The justification for games $G_1 - G_5$ is essentially identical to that given in the proof Theorem 4.1, thus we justify only the remaining games here. Note that to prove instead that the construction achieves priv-atk assuming the underlying scheme is priv-atk for atk $\in \{\mathrm{cpa}, \mathrm{cca}\}$ we would basically be done at this point.

We bound the probability that A_{g} when executed with G_5 makes a query y to its decryption oracle that is a valid ciphertext for some message m but A_{g} has not queried m to its hash oracle H, i.e., $H[m]$ is not defined as follows. Since A_{g} gets no information about the random string $H[m]$ in this case, it can do no better in order to make such a decryption query for a particular message m (in terms of the probability it makes such a query, taken as usual over a common finite set of coins as in the proof of the Fundamental Lemma in [9]) than to make the query a ciphertext $\mathcal{E}(pk, m, R)$ for coins R chosen by A_{g} at random (here A_{g} may or may not actually know m; the point is that it can do no better than this regardless), because in this case the distributions on the values of the ciphertexts $\mathcal{E}(pk, m; H[m])$ and $\mathcal{E}(pk, m, R)$ induced by the choices of $H[m]$ and R, respectively, are the same. This means that the probability that a query made by A_{g} to its decryption oracle satisfies the above condition for *any* plaintext cannot be more than $\mathrm{mc}_{\mathcal{PE}}(k)$, which in turn justifies (19).

Unlike game G_7, game G_6 may make a choice of $H[m]$ prematurely during the procedure to respond to a decryption oracle query. But this choice does not effect the response to the decryption

procedure Initialize All games

$b \xleftarrow{\$} \{0,1\}$
$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, m_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $R_0, R_1 \xleftarrow{\$} \{0,1\}^{l(k)}$
$C \leftarrow \mathcal{E}(pk, m_b; R_b)$
Return pk, C

On hash query x: Games G_1–G_4/G_5–G_6

If $H[x]$ is undefined then
  $H[x] \xleftarrow{\$} \{0,1\}^{l(k)}$
  $E[x] \leftarrow \mathcal{E}(pk, x; H[x])$
If $x = m_0$ then
  If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
  $\boxed{H[x] \leftarrow R_0}$
If $x = m_1$ then
  If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
  $\boxed{H[x] \leftarrow R_1}$
Return $H[x]$

On hash query x: Game G_7

If $H[x]$ is undefined then
  $H[x] \xleftarrow{\$} \{0,1\}^{l(k)}$
$E[x] \leftarrow \mathcal{E}(pk, x; H[x])$
If $x = m_0$ then
  If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
If $x = m_1$ then
  If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
Return $H[x]$

On decryption query y:
  Games G_1–G_5/G_6

If $\exists x_y$ such that $E[x_y] = y$ then
  Return x_y
$m \leftarrow \mathcal{D}(sk, y)$
If $m = \perp$ then return $\perp$
If $H[m]$ is undefined then
  $H[m] \xleftarrow{\$} \{0,1\}^{l(k)}$
  $E[m] \leftarrow \mathcal{E}(pk, m; H[m])$
  If $E[m] = y$ then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boxed{\text{Return } m}$
Else Return $\perp$

On decryption query y: Game G_7

If $\exists x_y$ such that $E[x_y] = y$
  then Return x_y
Else Return $\perp$

---

procedure Finalize(g) Game G_1

If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
Return d

---

procedure Finalize(g) Games G_2/G_3

If $\mathsf{zer} = \mathsf{true}$ then $d \leftarrow 0$
Else If $\mathsf{one} = \mathsf{true}$ then $d \leftarrow 1$
    Else If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
If $(b = 1 \wedge \mathsf{zer} = \mathsf{true}) \vee (b = 0 \wedge \mathsf{one} = \mathsf{true})$
  then $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{d \leftarrow b}$
Return d

---

procedure Finalize(g) Games G_4–G_7

If $\mathsf{zer} = \mathsf{true}$ then $d \leftarrow 0$
Else If $\mathsf{one} = \mathsf{true}$ then $d \leftarrow 1$
    Else If $g = t_1$ then $d \leftarrow 1$ else $d \leftarrow 0$
Return d

Figure 4: Games for the proof of Theorem 4.3.

---

oracle query, so it has no influence on the output of the game as compared to making this choice only when the particular hash oracle query m is made. Dropping the relevant code in the G_6 in order to do the latter therefore results in the equivalent game G_7. This justifies (20).

Now (21) is easy to see by comparing the code of B to that of Game G_7 and taking into account the definition of the advantage of B. ▮

We observe that in general ind-cpa security does not seem to be enough to imply that the max-collision probability of a public-key encryption scheme is negligible. To see this, we can take an ind-cpa scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and modify it such that the resulting scheme $\mathcal{PE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is also ind-cpa but any max-collision probability $\mathrm{mc}_{\mathcal{PE}'}(\cdot)$ equals one. We will also assume the existence of a one-way function generator $\mathcal{F}$ on $\{0,1\}^k$ (defined analogously to trapdoor permutation generators in Subsection 4.3). Note that, as is a standard fact, this is not an extra assumption given that we are assuming $\mathcal{PE}$ to be ind-cpa in the first place. The constituent algorithms of $\mathcal{PE}'$ work as follows:

- On input 1^k, $\mathcal{K}'$ runs $\mathcal{K}$ and $\mathcal{F}$ on the same input to receive outputs (pk, sk) and f, respectively. It then chooses $x^* \in \{0,1\}^k$ at random, sets $y \leftarrow f(x^*)$ outputs $(pk \parallel f \parallel y, sk \parallel x^*)$.
- On input $(pk \parallel f \parallel y, m)$, $\mathcal{E}'$ outputs 0 if $f(m) = y$ and $\mathcal{E}(pk, m)$ otherwise.
- On input $(sk \parallel x^*, C)$, $\mathcal{D}'$ outputs x^* if $C = 0$ and $\mathcal{D}(sk, C)$ otherwise.

(For $\mathcal{PE}'$ to meet Definition A.1, we add x^* above to $\mathrm{MsgSp}(k)$ if necessary and assume 0 is a special ciphertext not used by $\mathcal{PE}$.) It is straightforward to show that $\mathcal{PE}'$ is also an ind-cpa public-key encryption scheme but has max-collision probability always equal to one. Moreover, an ind-cpa adversary against $\mathcal{PE}'$ simulating the interaction of a priv-cca adverssary against the corresponding "encrypt-with-hash" ESE scheme with its experiment has no hope of correctly answering a query 0 made by the latter to its decryption oracle, since this requires the ind-cpa adversary to invert the one-way function f. What this shows is that one is unlikely to get away without an extra assumption in order to prove priv-cca security of the construction based on the ind-cpa security of the underlying scheme. Actually, with a little more work than in current proof, one can show that an apparently weaker assumption (also violated by $\mathcal{PE}'$) suffices here, namely that an algorithm given only the public key pk can, with at most negligible probability over the coin tosses of both the key generation and encryption algorithms, output a ciphertext for a message such that it yields the same ciphertext when re-encrypted under the same pk using fresh random coins. But we use the present condition for simplicity since it is met in practice anyway.

D Proof of Theorem 4.5

We prove Case 1, meaning we assume that $n(k) - k_0(k) < k_1(k) \leq n(k)$ (and of course $n(k) > 2k_0(k)$, which is true in either case). The proof for Case 2 is nearly identical. Let us first recall Lemma 6 from [20], stated in a form convenient to us.

Lemma D.1 [20] Let $\mathcal{F}_{\mathrm{RSA}}$ be the trapdoor-permutation generator. Let A be an algorithm that on input f, where $(f, f^{-1}) \stackrel{\$}{\leftarrow} \mathcal{F}_{\mathrm{RSA}}(1^k)$, and $y \in k_1(k)$, with probability $\delta(k)$ outputs $x \in \{0,1\}^{n(k)-k_0(k)}$, such that there exists $z \in \{0,1\}^{k_1(k)-n(k)+k_0(k)}$ such that $f(z \parallel x) = y$. Then there exists an inverter I against $\mathcal{F}_{\mathrm{RSA}}$ such that

$$\delta(k) \;\leq\; \sqrt{\mathbf{Adv}^{\mathrm{owf}}_{\mathcal{F}_{\mathrm{RSA}},I}(k) + 2^{4(k_0(k)-n(k))-6k_1(k)+10}} - 2^{2(k_0(k)-n(k))-3k_1(k)+5}$$

Note that in the above A is not required to actually find z. We remark that the algorithm in the above lemma is a *partially one-way* adversary against $\mathcal{F}_{\mathrm{RSA}}$ as defined in [20].

The strategy for our proof is to construct an algorithm satisfying the hypothesis of the above lemma and conclude the existence of an inverter against $\mathcal{F}_{\mathrm{RSA}}$ by the lemma. The algorithm, which

Algorithm $\mathsf{GetQuery}(f, Y)$
$ctr \leftarrow 0$
$j \xleftarrow{\$} \{1, \ldots, q_{H_2}\}$
$Y' \xleftarrow{\$} \{0,1\}^{n(k)-k_1(k)}$
Run A_{g} on input $f, Y' \,\|\, Y$, replying to its oracle queries as follows:
  On query x **to oracle** H_1:
  If $H_1[x]$ is undefined then
    $H_1[x] \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$
  Return $H_1[x]$
  On query x **to oracle** R:
  If $R[x]$ is undefined then
    $R[x] \xleftarrow{\$} \{0,1\}^{k_0(k)}$
  Return $R[x]$
  On query x **to oracle** H_2:
  $ctr \leftarrow ctr + 1$
  If $H_2[x]$ is undefined then
    $H_2[x] \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$
  If $ctr = j$ then
    $T \leftarrow x$
Until A_{g} halts
Return T

Figure 5: Algorithm $\mathsf{GetQuery}$ for proof of Theorem 4.5. The algorithm translates to an inverter I against $\mathcal{F}_{\mathrm{RSA}}$ as per Lemma D.1. See [20] for details of the construction.

---

we call $\mathsf{GetQuery}$, is depicted in Figure 5, and the games for the proof are depicted in Figure 6. (Here, as in the previous proofs, we use the game-playing technique of [9]. See the beginning of Appendix B for a brief summary of this technique.)

Equation (3) of Theorem 4.5 follows from the following sequence of inequalities which we will justify below:

$$\begin{aligned}
\frac{1}{2}+\frac{1}{2}\mathbf{Adv}^{\text{priv-cpa}}_{\mathsf{DOAEP},A}(k) &= \Pr\left[\, G_1^{A_{\mathrm{g}}} \Rightarrow b \,\right] && (22)\\
&\le \Pr\left[\, G_2^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \Pr[G_1^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_0\,] && (23)\\
&\le \Pr\left[\, G_2^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} && (24)\\
&\le \Pr\left[\, G_3^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \Pr[G_2^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_1\,] && (25)\\
&\le \Pr\left[\, G_3^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}} && (26)\\
&\le \Pr\left[\, G_4^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_2\,] && (27)\\
&= \Pr\left[\, G_5^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_2\,] && (28)
\end{aligned}$$

$$\begin{aligned}
&\le \Pr\left[\, G_6^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_2\,]\\
&+ \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] && (29)\\
&= \Pr\left[\, G_7^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_2\,]\\
&+ \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] && (30)\\
&\le \Pr\left[\, G_8^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_R(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_2\,]\\
&+ \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] + \Pr[G_7^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_4\,] && (31)\\
&\le q_{H_2}\sqrt{\mathbf{Adv}^{\text{owf}}_{\mathcal{F}_{\mathrm{RSA}},I}(k) + 2^{4k_0(k)-2k_1(k)+10}} - 2^{2k_0(k)-k_1(k)+5}\\
&+ \frac{q_R}{2^{k_0(k)}} + \frac{q_{H_1}(k)q_R(k)}{2^{\mathrm{me}_A(k)}}. && (32)
\end{aligned}$$

Equation (22) follows easily upon inspection of the code for game G_1 and the definition of the experiments in Definition 3.4.

Then we obtain (23) via application of the Fundamental Lemma in [9]. The probability that A_{g} when executed with game G_1 queries S_0 to R, while previously having queried neither T_0 to H_2 nor m_r to H_1, is at most $q_R/2^{n(k)}$ since A_{g} cannot possibly have information about S_0 in this case (all previous oracle queries are answered at random independently from everything else in the game). This justifies (24).

Now (25) is again obtained via application of the Fundamental Lemma. We bound the probability that A_{g} when executed with game G_2 queries m_r to H_1 and later queries S_0 to R, without querying T_0 to H_2 at any point prior, as follows. Observe that the string H_1^* given in response to query m_r to H_1 is random and independent from the view of A_{g} up to the point that query S_0 to R is made. This is because without querying T_0 to H_2, A_{g} can obtain information about H_1^* only by computing $S_0 \oplus m_r$ and can obtain information about S_0 only by querying S_0 to R. It means that if bad_1 is set here then A_{g} has "guessed" $m_l \,\|\, m_r$ without having any information about it,

in the technical sense that, since m_l is equal to $S_0 \oplus H[m_r]$, another machine can, with the same probability that this happens (taken, as in the proof of the Fundamental Lemma in [9], over a common finite set of coins with which these algorithms are executed), output $m_l \,\|\, m_r$ on input the input for $A_{\rm g}$ and the transcript of queries that $A_{\rm g}$ made to oracle H_1, R and their responses, which, unlike in G_2, are made at random independently from everything else (i.e., without performing any "If" checks). Regarding each pair of queries that that $A_{\rm g}$ makes to H_1, R as a single "guess" for $m_l \,\|\, m_r$, we see that this probability is at most $q_{H_1} q_R / 2^{\mathrm{me}_A(k)}$. We have justified (26).

As usual, the Fundamental Lemma justifies both (27) and (29). If $A_{\rm g}$ when executed with game G_4 queries S_0 to R and later queries m_r to H_1 without having queried T_0 to H_2 at any point prior, then it is given a random string independent from everything else in the game as the response to query S_0 to R, and, after it queries m_r to H_1, if it later queries T_0 to H_2 the response is likewise random and independent. Thus the string H_1^* given to $A_{\rm g}$ in response to query m_r to H_1 in this case is random and independent from its view during its entire execution with G_4. What this means is that we can safely drop the single-boxed "Else If" statement in the procedure to respond to oracle queries to H_1 in game G_4 without influencing the distribution of oracle query responses given to $A_{\rm g}$ in this game (taken again over a finite set of coins used in the executions) and hence the game output; doing so therefore results in an equivalent game that we have called G_5. This justifies (28).

Now consider when $A_{\rm g}$, executed with game G_6, queries m_r to H_1 but prior to this has queried neither S_0 to R nor T_0 to H_2. After it queries m_r to H_1, the responses given to $A_{\rm g}$ for its queries to any of its oracle are random and independent of everything else in the game. This means that the string H_1^* given to $A_{\rm g}$ as the response to its query m_r to H_1 here is also random and independent from the point of view of $A_{\rm g}$ during the rest of its execution. Thus, by the same reasoning as for (28) above, we drop the double-boxed "Else" statement to result in the equivalent game G_7, which justifies (30).

Applying the Fundamental Lemma again gives (31). Now we want to show that in each of the following cases, up to the point in its execution that the relevant bad is set, all queries made by $A_{\rm g}$ to its oracles can (if they are not already) be answered independently and at random from everything else without influencing the distribution of oracle query responses from the view of $A_{\rm g}$ up to that point (taken over a finite set of coins as in the proof of the Fundamental Lemma in [9]): game G_3 sets bad_2, game G_5 set bad_3, and game G_7 sets bad_4. The claim is that if this is true we are done, because, first of all, these cases exhaust the possible execution sequences in which $A_{\rm g}$ queries T_0 to H_2, and the algorithm $\mathsf{GetQuery}$, which answers all oracle queries at random independently from everything else, evidently succeeds with some positive probability in outputting T_0 just when such a query is made. Namely, in the games, the procedure to respond to queries to oracle H_2 explicitly checks for the case that a query is equal to T_0, while algorithm $\mathsf{GetQuery}$ simply guesses at random that this is the case for a particular query to H_2. It has a $1/q_{H_2}$ chance of being correct; Equation (32) then follows by Lemma D.1.

So let us examine each of these cases in turn. First consider when $A_{\rm g}$, executed with G_3, queries T_0 to H_2 after querying S_0 to R but without querying m_r to H_1 at any point prior. We see that the responses given to $A_{\rm g}$ to its oracle queries to up to the point in its execution that it queries T_0 to H_2 are random and independent from everything else. So the first case is settled. For the second case, consider when $A_{\rm g}$, executed with game G_5, queries T_0 to H_2 after having queried m_r to H_1. Here we see that the only previous query was not answered independently of everything else in the game is query m_r to H_1. But until $A_{\rm g}$ queries T_0 to H_2 it gets no information about H_1^* because, as noted in the justification of (26), such information can only be obtained by computing $S_0 \oplus m_l$,

and information about S_0 cannot be obtained in this game without querying T_0 to H_2 since until this query is made all queries to oracle R are answered independently at random. Thus query m_r to H_1 made by A_g can also be answered at random independent of everything else without influencing the distribution of oracle query responses from the view of A_g in the game up to the point that bad_3 is set, as desired. Finally, consider when A_g, executed with G_7, queries T_0 to H_2, previously having queried neither m_r to H_1 nor S_0 to R. Again it is clear that the responses to its oracle queries that A_g receives up to this point in its execution are random and independent of everything else in the game, which concludes the last case. ▮

We remark that the intuition for why we are able to achieve only priv-cpa security here is that, RSA-DOAEP being a length-preserving permutation, a random string of appropriate length is always a valid ciphertext. Correctly answering a decryption oracle query containing such strings requires a simulating adversary to invert RSA, and an incorrect response can be detected by a priv-cca adversary that encrypts the response and checks that it is correct, meaning equal to the original query. This is in contrast to the scheme of [46], where, due to randomization in encryption, it is unlikely the adversary would re-obtain the original ciphertext here.

procedure Initialize Game $G_1 - G_6$

$b \xleftarrow{\$} \{0,1\}$
$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, m_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(f, f^{-1}) \xleftarrow{\$} \mathcal{F}_{\mathrm{RSA}}(1^k)$
Parse m_b as $m_l \,\|\, m_r$,
  where $|m_l| = |m_r| = n(k)$
$H_1^*, H_2^* \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$; $R^* \xleftarrow{\$} \{0,1\}^{k_0(k)}$
$S_0 \leftarrow H_1^* \oplus m_l$; $T_0 \leftarrow R^* \oplus m_r$
$S_1 \leftarrow H_2^* \oplus S_0$
$Y \leftarrow S_1 \,\|\, f(T_0)$
Return (f, Y)

---

procedure Initialize Game G_7

$b \xleftarrow{\$} \{0,1\}$
$(t_0, m_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(f, f^{-1}) \xleftarrow{\$} \mathcal{F}_{\mathrm{RSA}}(1^k)$
$S_1 \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$; $T_0 \xleftarrow{\$} \{0,1\}^{k_0(k)}$
$Y \leftarrow S_1 \,\|\, f(T_0)$
Return (f, Y)

On query x **to** H_1:
Games $G_1 - G_4/G_5 - G_6/G_7$

If $H_1[x]$ is undefined then
  $H_1[x] \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$
If $x = m_r$ then
  If $H_2[T_0]$ is defined then
    $H_1[x] \leftarrow H_1^*$
  Else If $R[S_0]$ is defined then
    $\boxed{H_1[x] \leftarrow H_1^*}$
  Else $\boxed{\boxed{H_1[x] \leftarrow H_1^*}}$
Return $R[x_0]$

On query x **to** R: Games $G_1/G_2/G_3$

If $R[x]$ is undefined then
  $R[x] \xleftarrow{\$} \{0,1\}^{k_0(k)}$
If $x = S_0$ then
  If $H_2[T_0]$ is defined then
    $R[x] \leftarrow R^*$
  Else If $H_1[m_r]$ is undefined then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{R[x] \leftarrow R^*}$
  Else $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boxed{\boxed{R[x] \leftarrow R^*}}$
Return $R[x]$

On query x **to** H_2:
Games $G_1 - G_3/G_4, G_5/G_6$

If $H_2[x]$ is undefined then
  $H_2[x] \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$
If $x = T_0$ then
  If $R[S_0]$ is defined
    $\wedge\, H_1[m_r]$ is undefined then
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boxed{H_2[x] \leftarrow H_2^*}$
  If $H_1[m_r]$ is defined then
    $\mathsf{bad}_3 \leftarrow \mathsf{true}$; $\boxed{\boxed{H_2[x] \leftarrow H_2^*}}$
  Else $H_2[x] \leftarrow H_2^*$
Return $R[x]$

On query x **to** H_2: Games G_7/G_8

If $H_2[x]$ is undefined then
  $H_2[x] \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$
If $x = T_0$ then
  $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boxed{H_2[x] \leftarrow H_2^*}$
Return $R[x]$

---

procedure Finalize(g) All games

If $g = t_0$ then Return 1
Else Return 0

Figure 6: Games for the proof of Theorem 4.5. Here the single versus double-boxed statements indicate which statements are removed first and second, respectively, in the transitions indicated by the labels. (For example, the label "Game $G_1/G_2, G_3/G_4$" means that the single-boxed statement contained therein is absent for the games following (meaning in particular not including) G_1, while the double-boxed statement is absent just for the games following G_3. Thus both statements are absent for G_4.)