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 retrieve or update required parts on request just as efficiently as for unencrypted data. 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 of the constructions are also the first for asymmetric deterministic encryption schemes in general.

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

1 Introduction

SETUP. Despite a continuous decrease in 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 [52]. 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 setup 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 [29, 46, 31, 30, 18, 19, 46, 4, 32, 3, 33, 39]. Following M. Kantracioglu and C. Clifton [33], we observe 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 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 the strong definitions.

Previous and Related Work. On the one hand, previous works 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 that support these tasks [46, 4, 29, 19, 31, 32, 30] and often refer to such primitives as order-preserving hash functions and encryption or deterministic encryption, without suggesting proper definitions of security or candidate constructions. Indeed, the drawbacks inherent in this approach cause [33] 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 [53] in the symmetric-key setting and one by Boneh et al. [12] in the asymmetric setting, focus on the better-defined subproblem of secure keyword search on encrypted data [12, 28, 25, 13, 2, 6]. The schemes basically allow a server, just when given some secret information, to locate the ciphertexts containing particular keywords, without revealing the keywords (at least in the symmetric setting) or any other information about the message. But using them to answer queries asking to find records containing a particular keyword basically requires testing each record in the database one-by-one, which, as explained above, is prohibitive. And while symmetric-setting indexing methods of [17] allow constant time retrievals with even stronger security guarantees for this context, all records and possible keywords must be known in advance and the corresponding index pre-computed by the client, and updates remain prohibitive.

[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 [51, 38] can be employed.

Our Contributions. Our results address exact-match queries, where queries ask to retrieve records containing given attribute values, as well as updating such records or inserting new records. 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 also focus on the asymmetric (aka. public-key) setting, which allows anyone or a group of people to submit data for another group of people or a single person. In a medical context, for example, by encrypting data under a public key and submitting it appropriately to the server, nurses, secretaries, etc., can update (encrypted) medical records stored in a remote database or retrieve them as needed for the intended physicians, who can decrypt them with their corresponding secret key.

In Section 3, we introduce a definition to capture (asymmetric) encryption that *efficiently* supports this additional functionality, a primitive that we call *efficiently-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 by (separately) encrypting searchable attributes of records with an ESE scheme, exact-match index and search mechanisms on encrypted databases will be essentially the same as on unencrypted ones (this is what we mean by "efficiently"), which is appealing to implementors and means that search time has not gone up over unencrypted databases. For example, the server can dynamically index records according to the tags of their ESE-encrypted attributes using a tree-based index such as a B-tree, allowing logarithmic search time (in the size of the database).

Disregarding other attacks for the time being, querying the database at least amounts to encrypting and decrypting messages of the adversary's choice. But it turns out that finding a suitable security definition for ESE is not straightforward. We would like to capture the intuition that no useful information about the underlying message can be computed given a ciphertext, but clearly the the tag is *always* such useful information. Moreover, the adversary, given a ciphertext, can always compute its tag (which is the same as that of the underlying message) and compare it with the tag of candidate plaintexts. Hence we make two important 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, to make the definition achievable we do not allow the challenge ciphertext and the useful information that the adversary needs to compute to depend on the public key, which we argue is fine in practice. We also consider message spaces that have "enough entropy;" security otherwise is shown to be impossible with the given functionality. Note that non-searchable parts of records, in particular those with small entropy, would still be encrypted normally. We refer the reader to Section 3 for a discussion of ESE security and its implications for DAS.

In Section 4, we propose and analyze several ESE schemes, the high-level ideas for which, namely using the hash of message as its tag or else some form of deterministic encryption (for which the tag and ciphertext coincide), derive from the database literature. 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 (Construction 4.2) that replaces the randomness used in encryption by a standard scheme with the hash of the message, giving greater bandwidth-efficiency over the network and reduced computation-cost at the client side. We show it also achieves priv-cca in the RO model assuming that the underlying encryption scheme is only ind-cpa and satisfies a slight additional property (Theorem 4.3). In fact, we show that 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 unnecessary since known practical ind-cpa schemes already posses it.

Our last construction is a deterministic, length-preserving (in terms of ciphertext vs. plaintext length) ESE scheme based on RSA-OAEP [8, 22], which we call RSA-DOAEP (Construction 4.4). Note that for the first two constructions the underlying scheme can be a hybrid encryption scheme (i.e., one that encrypts a message under a symmetric key that is then encrypted itself under the public key and sent along with the ciphertext), so can be used to efficiently encrypt messages of various lengths. RSA-DOAEP, however, allows one to efficiently encrypt messages of arbitrary length without making use of any hybrid scheme, meaning it also saves on bandwidth for long messages by not having to include an (encrypted) symmetric key. We prove that RSA-DOAEP is priv-cpa in the RO model assuming RSA is one-way (Theorem 4.5). Then, in Section 5 we show that in addition to providing data authenticity, 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, this implies that by using encrypt-then-sign with a secure digital signature scheme, RSA-DOAEP in fact achieves priv-cca in applications requiring authentication of data anyway.

Though we typically discuss them in the context of DAS, 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] Cryptographic schemes are proven secure assuming a source truly random bits whereas computers are in actuality deterministic, and implementing "randomness" generation in practice remains a tricky process that can end up compromising security. Using a deterministic scheme instead is therefore attractive whenever possible in terms of security guarantee.

Further Related Work. Obfuscated databases, studied in [44], 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 [44] 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 [54, 27], private information retrieval [15, 37, 21], searching on streaming data [45], oblivious RAM [26] and secure data mining and statistical databases [48, 16, 34, 14, 36, 20, 11] 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. The protocols are also usually not efficient.

2 Preliminaries

Notation. We refer to members of the set $\{0,1\}^*$ as strings. If X is a string then $|X|$ denotes its length in bits and if X, Y are strings then $X \| Y$ denotes the concatenation of X and Y. If S is a set then $X \stackrel{\$}{\leftarrow} S$ denotes that X is selected uniformly at random from S. For convinience,

[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.

$k \in \mathbb{N}$ 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 $a = A(x, y, \ldots; R)$. By convention, the running-time of an algorithm here is measured relative to bit-length of the input and refers to both the actual running-time and program size, including that of any overlying experiment, all relative to some fixed RAM model of computation.

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 also recall the standard syntax and security definitions for asymmetric encryption schemes in Appendix A. Note that the definition for security of encryption there (Definition A.2) allows the adversary multiple lr-encryption queries (allowing one such query is equivalent), which is to better interface with our new definitions below.

We will also need to consider vectors of messages and ciphertexts. Following [10], we denote them like $\mathbf{x}$, where $\mathbf{x}[i]$ denotes the ith component of the vector $\mathbf{x}$. We allow encryption and decryption to operate on such vectors component-wise, so, for example, if $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an encryption scheme then $\mathbf{y} \leftarrow \mathcal{E}(pk, \mathbf{x})$ means for all i do: $\mathbf{y}[i] \xleftarrow{\$} \mathcal{E}(pk, \mathbf{x}[i])$. We also extend set-membership notation to vectors, writing $x \in \mathbf{x}$ to mean that there exists i such that $x = \mathbf{x}[i]$.

3 Efficiently-Searchable Encryption (ESE) and its Security

We formulate an extension of Definition A.1 to capture encryption schemes that allow to efficiently search encrypted databases as discussed in the introduction, 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 *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 ϵ such that the following conditions hold:
(1) Completeness:

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

(2) Soundness:

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

for every message $m_1 \in \mathrm{MsgSp}(k)$ and every RPTA $\mathcal{M}$ that outputs distinct messages $m_0, m_1 \in \mathrm{MsgSp}(k)$ when given the public key. We refer to the output of F, G as the *tag* of a message m or a corresponding ciphertext C.

Note that the soundness condition, which prevents false-positive results in searches (possibly due to malicious data entry), is "computational," in the spirit of [2]. A stronger, chosen-ciphertext version of this condition would give $\mathcal{M}$ access to a decryption box as well. We omit to do this for simplicity and remark that all our proposed constructions also meet this stronger version (in the random oracle model) anyway.

Next let us formally define the examples of ESE discussed in the introduction. The first one is a simple approach one can take here 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 it is easy to see that $\mathcal{HPE}$ 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. The function ϵ in the definition is identically zero here due to the consistency requirement in Definition A.1.

It is easy to see that no ESE scheme can be ind-cpa. Thus we introduce a definition of security for asymmetric ESE schemes, which captures the intuition that a ciphertext should not reveal any information about the corresponding plaintext beyond what is needed for the server to index and search for it efficiently. The high-level idea originates from the definition of semantic security in [40]; however, we make several important modifications in our definition that are explained below.

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 vector of distinct messages $\mathbf{x}$ with $\mathbf{x}[i] \in \mathrm{MsgSp}(k)$ for all i, together with a string t that represents some information about $\mathbf{x}$. Later, A_{g} gets a public key and an encryption of $\mathbf{x}$ under this key, and tries to compute t. (Note that t is not required to be efficiently computable given $\mathbf{x}$. For example, A_{m} could output $\mathbf{x}, t$ such that $\mathbf{x}[1] = f(t)$ 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 $\mathbf{Exp}^{\text{priv-atk-0}}_{\mathcal{SPE},A}(k)$
$(t_1, \mathbf{x}_1) \stackrel{\$}{\leftarrow} A_{\mathrm{m}}(1^k)$	$(t_0, \mathbf{x}_0) \stackrel{\$}{\leftarrow} A_{\mathrm{m}}(1^k)$; $A_{\mathrm{m}}(t_1, \mathbf{x}_1) \stackrel{\$}{\leftarrow} (1^k)$
$(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k)$	$(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k)$
$g \stackrel{\$}{\leftarrow} A_{\mathrm{g}}^{\mathcal{O}(sk,\cdot)}(pk, \mathcal{SE}(pk, \mathbf{x}_1))$	$g \stackrel{\$}{\leftarrow} A_{\mathrm{g}}^{\mathcal{O}(sk,\cdot)}(pk, \mathcal{SE}(pk, \mathbf{x}_0))$
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. For the corresponding case, we say that $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ is a *priv*-atk *adversary* if there are functions v, l such that for every k:

$$\Pr\left[\, (t, \mathbf{x}) \stackrel{\$}{\leftarrow} A_{\mathrm{m}}(1^k) \; : \; |\mathbf{x}| = v(k) \, \wedge \, |\mathbf{x}[i]| = l(k) \,\right] \; = \; 1 \;\; \text{for all } i \; ,$$

the messages in the output vector of A_{m} are always distinct, and A_{g} does not query any component of its challenge-ciphertext vector to its decryption oracle. The *privacy-advantage* of a priv-atk 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 atk = cpa and *private against chosen-ciphertext attack* or *priv-cca* for atk = cca if for every RPT priv-atk 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 $(0, 0^k)$ (the second component being a vector of size one) 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 $\Pr\left[\, \mathbf{Exp}^{\text{priv-atk-1}}_{\mathcal{SPE},A}(k) = 1 \,\right] = 1$. 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 $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ is a RPT priv-atk 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 priv-atk adversaries $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ where the message space implicitly defined by A_{m} has large min-entropy. To capture this, we say that $\mathrm{me}_A(\cdot)$ is a *message space min-entropy* function for A if for every $m^* \in \{0,1\}^*$ and every k:

$$\Pr\left[\, (t, \mathbf{x}) \stackrel{\$}{\leftarrow} A_{\mathrm{m}}(1^k) \;:\; m^* = \mathbf{x}[i] \,\right] \;\leq\; \frac{1}{2^{\mathrm{me}_A(k)}} \quad \text{for all } i \;.$$

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. Of course, this requirement is not meant to preclude a given attribute value drawn from a message space with large min-entropy from being encrypted and stored in many records, as is the nature of database encryption. Thus it is unavoidable that the adversary will detect ciphertexts corresponding to equal plaintexts, as well as the query-access patterns for the encrypted records. We also stress that non-searchable attributes, in particular those with poor min-entropy, should always be encrypted with a standard ind-cca encryption scheme as a hedge against "statistical" attacks arising from a priori semantic relationships in the data. (For example, if an attribute "profession" has the value "student," it may mean that other attributes like income are more likely to take certain values as compared to an arbitrary record.)

Moreover, to make the definition achievable 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 just abstractions hidden in our software, not strings that we look at. Because of this restriction, however, considering vectors of messages is needed to ensure achieving multi-message security from our definition, allowing the adversary to see encryptions of some related messages (which is captured in standard indistinguishability-based definitions via equivalence to multiple lr-encryption queries). For example, a scheme that appends the encryption of the bit-wise complement of the plaintext to the ciphertext can still be shown insecure under our definition.

As mentioned above, 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 RO, 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, where the hash of a message is 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 atk $\in \{\text{cpa}, \text{cca}\}$, let $A = (A_{\text{m}}, A_{\text{g}})$ be a RPT priv-atk adversary against $\mathcal{HPE}$ that outputs a vector of length $v(\cdot)$, having message space min-entropy function $\text{me}_A(\cdot)$ and making at most $q_h(\cdot)$ queries to its hash oracle and $q_d(\cdot)$ to its decryption oracle. Then there exists an RPT ind-atk-adversary B against $\mathcal{PE}$ such that for all k,

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

Furthermore, the running-time of B is at most that of A plus $O(q_h(k)v(k)l(k))$.

The proof is in Appendix B.

To get an idea of its security in practice, note that, assuming the underlying (standard) scheme is secure, the theorem implies that an adversary's maximal advantage against the construction (in the analogous sense) when its message space min-entropy is at least, say, 80 bits is small unless it makes about 2^{60} hash queries or related-encryption computations (the number of the latter given by $v(k)$), at which point it still has only around a $1/2^{20}$ chance of breaking the resulting scheme.

Also note that the underlying scheme can be a hybrid encryption scheme, so can be used to efficiently encrypt messages of various lengths.

4.2 Encrypt-with-Hash

We propose a general deterministic ESE construction (as per Construction 3.3) replacing the coins used by a standard encryption scheme with the hash of the message, which, as compared to the previous hash-and-encrypt construction, offers better bandwidth-efficiency over the network and computation-efficiency on the client side. The 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 (for priv-cpa security of the construction, ind-cpa security of the underlying scheme alone suffices). 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 $m \in \mathrm{MsgSp}(k)$. (Note that m here can 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 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 [23] has max-collision probability $1/|G|$, where G is the used group (which also depends on k). 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. $\mathcal{E}^*(pk, m; R_1 \parallel R_2)$ returns $(\mathcal{E}(pk, m; R_1) \parallel R_2)$, where $|R_2| = k$, meaning $\mathcal{E}^*$ uses $r + k$ coins; and $\mathcal{D}^*(sk, C \parallel R)$ 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 $\mathcal{DPE}$ that outputs a vector of length $v(\cdot)$, having message space min-entropy function $\mathrm{me}_A(\cdot)$ and making at most $q_h(\cdot)$ queries to its hash oracle and $q_d(\cdot)$ to its decryption oracle. Then there exists an RPT ind-cpa-adversary B against $\mathcal{PE}$ such that for all k,

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

Furthermore, the running-time of B is at most that of A plus $O(q_h(k)l(k)(v(k) + T_{\mathcal{E}}(k) + q_d(k)))$, where $T_{\mathcal{E}}(k)$ is the time for one computation of $\mathcal{E}$ when the security parameter is k.

The proof is in Appendix C. The weaker result about priv-cpa security analogous to the above, but with the last additive term in (2) removed, can be stated and proved, but we omit to do this in light of the above discussion.

As a secure public-key deterministic encryption scheme is simply a family of (injective) trapdoor functions with some extra security properties, we remark that by a result of Gertner et al. [24] showing that in the standard model there is no black-box reduction from trapdoor functions to trapdoor predicates, it will be hard to build a secure public-key deterministic encryption scheme based solely on a secure standard one without random oracles.[3] However, it does not preclude building a secure deterministic one without random oracles 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 database or a battery-constrained device ([43],

[3] More specifically, as discussed in [24], according to this result such a construction will either have to rely in an intrinsic way on the semantic security of the standard encryption scheme or be non-black-box.

Section 2). We devise an efficient scheme that is optimal in this regard, being length-preserving. (Note that it is a standard fact that this cannot be achieved using ind-cpa encryption, a further draw for this scheme in any application requiring length-preserving encryption where message space min-entropy is large.) We term the scheme RSA-DOAEP ("D" for deterministic), as it is based on RSA-OAEP [8, 22]. The design is also reminiscent of the more-recent "3-round OAEP" scheme of Phan and Pointcheval [47], which achieves ind-cca security in the RO model without redundancy.

While for the previous two ESE constructions the underlying scheme can be a hybrid encryption scheme so can be used to efficiently encrypt messages of various lengths, as can be seen from the construction below with RSA-DOAEP one can efficiently encrypt messages of arbitrary length without making use of any hybrid scheme. Hybrid encryption can in some sense never be length-preserving because an (encrypted) symmetric key is included with a ciphertext, thus RSA-DOAEP also saves on bandwidth for long messages in this respect.

The security of our scheme assumes the existence of one-way trapdoor permutations, the basics for which we recall in Appendix A. In particular, we will use the well-studied RSA trapdoor permutation generator $\mathcal{F}_{\mathrm{RSA}}$ [50], which is widely assumed as one-way. For simplicity, we will further assume that all messages have length $n(k)$ for some polynomial n. In practice, to bypass this assumption and maintain the length-preserving property, one can use a variable-length instantiation for the ROs. For example, one can use a common instantiation heuristic first suggested in [7] where one obtains the RO output for a given string x by computing $H(K, x \,\|\, \langle 1\rangle) \,\|\, H(K, x \,\|\, \langle 2\rangle) \,\|\, \ldots$ to sufficient length, where H is a cryptographic 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 then truncating the result as needed. To apply our security analysis in this case, one can 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 length functions k_0, k_1 satisfying and $n(k) > 2k_0(k)$ and $n(k) \geq k_1(k)$ for all k. The key-generation algorithm of the ESE 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 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 $\mathcal{E}_f^{H_1,H_2,R}(m)$	**Algorithm** $\mathcal{D}_{f^{-1}}^{H_1,H_2,R}(Y)$
Parse m as $m_l \,\|\, m_r$,	Parse Y as $X_1 \,\|\, Y'$,
where $\lvert m_r\rvert = n(k) - k_0(k)$	where $\lvert Y'\rvert = k_1(k)$
$S_0 \leftarrow H_1(m_r) \oplus m_l$; $T_0 \leftarrow R(S_0) \oplus m_r$	$X \leftarrow X_1 \,\|\, f^{-1}(Y')$
$S_1 \leftarrow H_2(T_0) \oplus S_0$	Parse X as $S_1 \,\|\, T_0$
Parse $S_1 \,\|\, T_0$ as $X_1 \,\|\, X_2$,	where $\lvert S_1\rvert = k_0(k)$ and $\lvert T_0\rvert = n(k) - k_0(k)$
where $\lvert X_2\rvert = k_1(k)$	$S_0 \leftarrow H_2(T_0) \oplus S_1$; $m_r \leftarrow R(S_0) \oplus T_0$
$Y \leftarrow X_1 \,\|\, f(X_2)$	$m_l \leftarrow H_1(m_r) \oplus S_0$
Return Y	Return $m_l \,\|\, m_r$

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 that outputs a vector of length $v(\cdot)$, 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 $q_R(\cdot)$ to oracle R. Then RSA-DOAEP is priv-cca in the RO model if $\mathcal{F}_{\mathrm{RSA}}$ is one-way. More precisely, we divide the result into two cases:

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

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

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

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

Furthermore, in the first case the running-time of I is at most twice that of A plus $O(v(k)l(k) + q_{H_2}(k)\log q_{H_2}(k) + k^3)$, while in the second it is at most that of A plus $O(v(k)l(k) + q_{H_2}(k)\log q_{H_2}(k))$.

The proof is in Appendix D. In fact, when combined properly with a digital signature scheme RSA-DOAEP achieves priv-cca security, as we shall see in the following section.

To conclude this section, let us derive parameter settings to maximize "exact" security with respect to RSA, when we fix, say, $k = 1024$ [49]. Thus n must be at least 1024 bits, and we set k_1 to 1024 bits. In fact, we can also set k_0 to, say, 80 bits, regardless of the exact value of n. To see this, first suppose n is at least 1104 bits. Then evidently we should set k_0 as high as possible, or high enough to thwart brute-force guessing on oracle R, under the inequality $k_1 \leq n - k_0 \leq n$, so to, say, 80 bits, and Case 2 of the theorem then applies to give the better security guarantee of the two cases. (Note that the multiplicative factor $v(k)$ here is not abnormal, a factor like this being implicit if using an indistinguishability-based definition allowing an adversary only one lr-encryption query, as in [22].) If instead $n(k)$ is between 1024 and 1103 bits, we are resigned to Case 1. The expression under the radical in (3) increases super-linearly as a function of $2^{k_0(k)}$, so we see that in this case $k_0(k)$ should be set as *low* as possible, which is again to (say) 80 bits. The weak security guarantee here is analogous to what is known for RSA-OAEP [22, 49].

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 at the server side. Note that it will guarantee this at the field level, and not on the record level or for the database as a whole; the adversary can still, for example, switch (encrypted) attribute values stored in different records. If the data is updated and returned as whole records, then one can simply authenticate at the record level instead. In many applications, however, the server can be trusted to return the correct ciphertexts to its paying customers, even though it may try to learn and sell their data. (Otherwise, ensuring that the server returns all the current, requested data on each query, which is outside our scope, can be dealt with using the methods of [41, 42, 43, 38].)

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 this way for 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. (We omit more general definitions, which trivially extend from [5].)

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 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 $\perp$. But, as discussed in [5], we do not allow the adversary 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. This means that to also prevent legitimate signers from forging or modifying and possibly decrypting messages from other legitimate signers, they all need distinct key-pairs for signatures. (It is called "outsider" security in [5].)

Here then is the 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 (induced uf-cma security) is identical to part of the proof of Theorem 1 in [5], hence omitted. The proof of the second part (induced priv-cca security) is nearly 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 details, note that 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 encryption (ESE) as a tool 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. Also, we 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.

7 Acknowledgements

We thank Jun Li and Ed Omiecinski for introducing the problem to us, Brian Cooper for useful comments and references, and Alex Dent and Ulrich Kühn for their feedback on an earlier draft of this paper.

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] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, pages 26–45, 1998.

[11] 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.

[12] 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.

[13] 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.

[14] S. Chawla, C. Dwork, F.McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In Kilian [35], pages 363–385.

[15] 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.

[16] 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.

[17] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. Cryptology ePrint Archive, Report 2006/210, 2006. `http://eprint.iacr.org/`.

[18] 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.

[19] 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.

[20] 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.

[21] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In Kilian [35], pages 303–324.

[22] 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.

[23] 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.

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

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

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

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

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

[38] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin. Dynamic authenticated index structures for outsourced databases. In *SIGMOD*. ACM Press, 2006.

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

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

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

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

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

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

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

[46] 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.

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

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

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

[50] 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.

[51] 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.

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

[53] 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.

[54] 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 and, following this, trapdoor permutations.

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 *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 $(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$.
- 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 $C \xleftarrow{\$} \mathcal{E}(pk, m)$ 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 $\in \{\mathrm{cpa}, \mathrm{cca}\}$, adversary B_{atk} and $b \in \{0, 1\}$ define the experiment:

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

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 every query (m_0, m_1) it makes to its left-or-right encryption oracle satisfy $|m_0| = |m_1|$, 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:

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

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 $(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k)$.
- 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 $\sigma \stackrel{\$}{\leftarrow} \mathcal{E}(pk, m)$.
- 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:

Experiment $\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{DS},B}(k)$
$(pk, sk) \stackrel{\$}{\leftarrow} \mathcal{K}(1^k)$
$(m, \sigma) \stackrel{\$}{\leftarrow} B^{\mathcal{S}(sk,\cdot)}(pk)$
Return $\mathcal{V}(pk, m, \sigma)$

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. ▮

Trapdoor permutations. 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 RPT encoding of a permutation and f^{-1} (called the "trapdoor") is an RPT encoding of its inverse. An *inverter* I against $\mathcal{F}$ is an algorithm that takes as input $f, f(x)$ and tries to compute x. We say that $\mathcal{F}$ is *one-way* if for every RPT inverter I the function $\mathbf{Adv}^{\text{owf}}_{\mathcal{F},I}(\cdot)$ defined as:

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

is negligible. ▮

Adversary $B^{\mathcal{E}(pk,\mathrm{LR}(\cdot,\cdot,b)),\mathcal{D}(sk,\cdot)}(pk)$
$(t_0,\mathbf{x}_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1,\mathbf{x}_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
For all i do:
  $H \xleftarrow{\$} \{0,1\}^{l(k)}$
  $\mathbf{y}[i] \xleftarrow{\$} H \parallel \mathcal{E}(pk, \mathrm{LR}(\mathbf{x}_0[i], \mathbf{x}_1[i], b))$
Run A_{g} on input $pk, \mathbf{y}$, replying to its oracle queries as follows:
  On hash query x:
  If $x \in \mathbf{x}_0$ then
    If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
  If $x \in \mathbf{x}_1$ 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 $H_y \parallel C_y$
  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.

B Proof of Theorem 4.1

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

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

procedure Initialize All games

$b \xleftarrow{\$} \{0,1\}$
$(t_0, \mathbf{x}_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, \mathbf{x}_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$
For $i \leftarrow 1, 2$ do:
  $H_{i,0}, \ldots, H_{i,v(k)} \xleftarrow{\$} \{0,1\}^{l(k)}$
For all j do:
  $\mathbf{y}[j] \xleftarrow{\$} H_{b,j} \parallel \mathcal{E}(pk, \mathbf{x}_b[j])$
Return $pk, \mathbf{y}$

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 $\exists i$ such that $x = \mathbf{x}_0[i]$ then
  If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
  $\boxed{H[x] \leftarrow H_{0,i}}$
If $\exists j$ such that $x = \mathbf{x}_1[j]$ then
  If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
  $\boxed{H[x] \leftarrow H_{1,j}}$
Return $H[x]$

On decryption query y: All games

Parse y as $H_y \parallel C_y$
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.

---

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_{g} with each of these games. The execution of A_{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_{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_{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 "$G_i^{A_\mathrm{g}} \Rightarrow b$" denote the event that the output of Game G_i, when executed with A_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:

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

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 $H(\mathbf{x}_i[1]), \ldots, H(\mathbf{x}_i[v(k)])$ for $i \in \{0,1\}$ upfront, these being justified by the fact that the messages in the output vector of A_m are distinct, 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_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_g makes a hash query $x \in \mathbf{x}_{1-b}$ when executed with G_3 is at most $q_h(k)v(k)/2^{\mathrm{me}_A(k)}$ because A_g gets no information about $\mathbf{x}_{1-b}$. 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_g with either G_4 or G_5. Consider the execution of A_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

Adversary $B^{\mathcal{E}(pk,\mathrm{LR}(\cdot,\cdot,b))}(pk)$
$(t_0, \mathbf{x}_0) \stackrel{\$}{\leftarrow} A_{\mathrm{m}}(1^k)$; $(t_1, \mathbf{x}_1) \stackrel{\$}{\leftarrow} A_{\mathrm{m}}(1^k)$
$\mathbf{y} \stackrel{\$}{\leftarrow} \mathcal{E}(pk, \mathrm{LR}(\mathbf{x}_0, \mathbf{x}_1, b))$
Run A_{g} on input $pk, \mathbf{y}$, replying to its oracle queries as follows:
  On hash query x:
  If $x \in \mathbf{x}_0$ then
    If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
  If $x \in \mathbf{x}_1$ 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.

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.

It remains to justify the running-time analysis of B. Recall our convention to include in the running-time of A that of its overlying experiment, so the only extra overhead for B is initially selecting $v(k)$-many $l(k)$-bit random numbers, and then on each hash query made by A_{g} searching through two vectors of size $v(k)$ vectors with each component an $l(k)$-bit message and setting some flags accordingly, as well as checking those flags at the end of its execution. This gives us the $O(q_h(k)v(k)l(k))$ overhead stated in the theorem. ▌

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:

procedure Initialize All games

$b \xleftarrow{\$} \{0,1\}$

$(t_0, \mathbf{x}_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, \mathbf{x}_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$

For $i \leftarrow 1, 2$ do:

  $R_{i,0}, \ldots, R_{i,v(k)} \xleftarrow{\$} \{0,1\}^{l(k)}$

For all j do:

  $\mathbf{y}[j] \xleftarrow{\$} \mathcal{E}(pk, \mathbf{x}_b[j]; R_{b,j})$

Return $pk, \mathbf{y}$

On hash query x:
  Games G_1–G_4/G_5–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 $\exists i$ such that $x = \mathbf{x}_0[i]$ then

  If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$

  $\boxed{H[x] \leftarrow R_{0,i}}$

If $\exists j$ such that $x = \mathbf{x}_1[j]$ then

  If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$

  $\boxed{H[x] \leftarrow R_{1,j}}$

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.

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

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 $\in \{\text{cpa}, \text{cca}\}$ we would basically be done at this point.

We bound the probability that $A_{\rm g}$ when executed with G_5 makes a query y to its decryption oracle that is a valid ciphertext for some message m but $A_{\rm g}$ has not queried m to its hash oracle H, i.e., $H[m]$ is not defined as follows. Since $A_{\rm g}$ 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 $A_{\rm g}$ at random (here $A_{\rm g}$ 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 $A_{\rm g}$ 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 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.

Finally, to justify the claim about the running-time of B, recall the convention to include in the running-time of A that of its overlying experiment. So extra overhead for B here includes encrypting each hash query made by $A_{\rm g}$ and storing the result in an array (indexed by the

$l(k)$-bit query), and searching the array for a corresponding message to each decryption query. The rest of its overhead analogous to in the proof of Theorem 4.1 in the previous appendix. ▌

We remark that in general ind-cpa security alone is not enough to guarantee priv-cca security of the construction. To see this, let us 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 the corresponding encrypt-with-hash scheme is not priv-cca secure. The constituent algorithms of $\mathcal{PE}'$ work as follows:

- On input 1^k, $\mathcal{K}'$ runs $\mathcal{K}$ and on the same input and returns its output (pk, sk).
- On input (pk, m), $\mathcal{E}'$ outputs 0 if $m = sk$ and $\mathcal{E}(pk, m)$ otherwise.
- On input (sk, C), $\mathcal{D}'$ outputs sk if $C = 0$ and $\mathcal{D}(sk, C)$ otherwise.

(For $\mathcal{PE}'$ to meet Definition A.1, we 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 is not priv-cca secure. Note that any max-collision probability $\mathrm{mc}_{\mathcal{PE}'}(\cdot)$ for $\mathcal{PE}'$ is identically one.

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 [22], stated in a form convenient to us.

Lemma D.1 [22] Let $\mathcal{F}_{\mathrm{RSA}}$ be the trapdoor-permutation generator. Let A be an RPT 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 $f(z \,\|\, x) = y$. Then there exists an RPT inverter I against $\mathcal{F}_{\mathrm{RSA}}$ such that

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

where the running-time of I is at most twice that of A plus $O(k^3)$.

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 [22].

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

To simplify the proof, in a game with multiple boolean variables bad_i defined, we will assume in the analysis when considering the probability that bad_i is set that no other $\mathsf{bad}_{i'}$ for $i \neq i'$ has been set previously. This is justified because the contrary would be subsumed by another case anyway, i.e., some bad_i has to be set first.

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

Algorithm GetQuery(f, Y)
$ctr \leftarrow 0$
$j \xleftarrow{\$} \{1, \ldots, q_{H_2}\}$; $w \xleftarrow{\$} \{1, \ldots, v(k)\}$
$\mathbf{y} \xleftarrow{\$} \{0,1\}^{n(k)} \times \cdots \times \{0,1\}^{n(k)}$ /* pick random $v(k)$-size vector */
$Y' \xleftarrow{\$} \{0,1\}^{n(k)-k_1(k)}$
$\mathbf{y}[w] \leftarrow Y' \parallel Y$
Run A_{g} on input $f, \mathbf{y}$, 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 GetQuery for proof of Theorem 4.5. The algorithm translates to an RPT inverter I against $\mathcal{F}_{\mathrm{RSA}}$ as per Lemma D.1. See [22] for details of the construction.

$$
\begin{aligned}
\frac{1}{2}+\frac{1}{2}\mathbf{Adv}^{\text{priv-cpa}}_{\mathsf{DOAEP},A}(k) &= \Pr\left[\, G_1^{A_g} \Rightarrow b \,\right] && (22)\\
&\le \Pr\left[\, G_2^{A_g} \Rightarrow b \,\right] + \Pr[G_1^{A_g} \text{ sets } \mathsf{bad}_0\,] && (23)\\
&\le \Pr\left[\, G_2^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} && (24)\\
&\le \Pr\left[\, G_3^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \Pr[G_2^{A_g} \text{ sets } \mathsf{bad}_1\,] && (25)\\
&\le \Pr\left[\, G_3^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}} && (26)\\
&\le \Pr\left[\, G_4^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}}\\
&+ \Pr[G_3^{A_g} \text{ sets } \mathsf{bad}_2\,] && (27)\\
&= \Pr\left[\, G_5^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}}\\
&+ \Pr[G_3^{A_g} \text{ sets } \mathsf{bad}_2\,] && (28)\\
&\le \Pr\left[\, G_6^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_g} \text{ sets } \mathsf{bad}_2\,]\\
&+ \Pr[G_5^{A_g} \text{ sets } \mathsf{bad}_3\,] && (29)\\
&= \Pr\left[\, G_7^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_g} \text{ sets } \mathsf{bad}_2\,]\\
&+ \Pr[G_5^{A_g} \text{ sets } \mathsf{bad}_3\,] && (30)\\
&\le \Pr\left[\, G_8^{A_g} \Rightarrow b \,\right] + \frac{q_R(k)v(k)}{2^{n(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}} + \Pr[G_3^{A_g} \text{ sets } \mathsf{bad}_2\,]\\
&+ \Pr[G_5^{A_g} \text{ sets } \mathsf{bad}_3\,] + \Pr[G_7^{A_g} \text{ sets } \mathsf{bad}_4\,] && (31)\\
&\le q_{H_2}v(k)\sqrt{\mathbf{Adv}^{\mathrm{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 v(k)}{2^{k_0(k)}} + \frac{q_{H_1}(k)q_R(k)v(k)}{2^{\mathrm{me}_A(k)}}. && (32)
\end{aligned}
$$

As in the previous proofs (cf. the justification of (5) in Appendix B), (22) follows easily from the code for game G_1 and 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 $S_{i,0}$ to R for some i, while previously having queried neither $T_{i,0}$ to H_2 nor $m_{i,0}$ to H_1, is at most $q_R v(k)/2^{n(k)}$ since A_g cannot have any information about any such $S_{i,0}$ in this case, as it is also easy to see from the code that the responses to all its previous oracle queries and hence also its input (using the fact that if a string r is random and independent from $S_{i,0}$ then so is $r \oplus x$ for any string x) and are random and independent from $S_{i,0}$, up to the point in the execution that this query is made. 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 $m_{i,r}$ for some i to H_1 and later queries $S_{i,0}$ to R, without querying $T_{i,0}$ to H_2 at any point prior, as follows. First observe that the string $H^*_{i,1}$ given in response to query $m_{i,r}$ to H_1 is random and independent from the view of A_g up to the point that query $S_{i,0}$ to R is made. This is because without querying $T_{i,0}$ to H_2, A_g gets no information about $H^*_{i,2}$ so can obtain information about $H^*_{i,1}$ only by computing $S_{i,0} \oplus m_{i,l}$, but can obtain information about S_0 only by querying it to R. Similarly, without querying $S_{i,0}$ to R, A_g gets no information about the random string $T_{i,0} = R^*_i \oplus m_{i,r}$ (more accurately, it can only obtain an alleged value for R_i^* that it cannot verify), and without querying $T_{i,0}$ to $H_{i,2}$ it gets no information about $S_{i,1} = H_{i,2}^* \oplus S_{i,0}$. So until these queries are made, in the input $S_{i,1} \parallel f(T_{i,0})$ the strings $S_{i,1}, T_{i,0}$ are also independent and random from the view of $A_{\rm g}$. This means that if bad_1 is set here then $A_{\rm g}$ has "guessed" $m_{i,l} \parallel m_{i,r}$ without getting any information about it, in the technical sense that, since $m_{i,l}$ is equal to $S_{i,0} \oplus H[m_{i,r}]$, another RPTA K (a "knowledge extractor") can, with the same probability that this happens, output a list of strings containing $m_{i,l} \parallel m_{i,r}$ on input the transcript of queries that $A_{\rm g}$ made to oracles H_1, R and their responses when executed in a game where all input, including the oracle query responses, are given independently at random. (Here we use the assumption that no other bad_j for $j \neq 1$ has been set previously. Also note that as usual and as in the proof of the Fundamental Lemma in [9], this probability is taken over a common finite set of coins with which these algorithms are executed, which will not be explicitly mentioned again.) Namely, K outputs a list comprised of $q \oplus H[s] \,||\, s$, for every query s made to oracle R and every query r made to oracle H. We thus regard each pair of queries that that $A_{\rm g}$ makes to H_1, R as a single "guess" for $m_{i,l} \parallel m_{i,r}$, and then we see from the definition of message space min-entropy in Definition 3.4 that the probability K outputs such a list containing $m_{i,l} \parallel m_{i,r}$ is at most $q_{H_1} q_R v(k)/2^{\mathrm{me}_A(k)}$. 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 $S_{i,0}$ for some i to R and later queries $m_{i,r}$ to H_1 without having queried $T_{i,0}$ 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 $S_{i,0}$ to R, and, after it queries $m_{i,r}$ to H_1, if it later queries $T_{i,0}$ to H_2 the response is likewise random and independent. Thus the string $H_{i,1}^*$ given to $A_{\rm g}$ in response to query $m_{i,r}$ 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 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 and hence the game output; doing so therefore results in an equivalent game that we have called G_5, which justifies (28).

Now consider when $A_{\rm g}$, executed with game G_6, queries $m_{i,r}$ for some i to H_1 but prior to this has queried neither $S_{i,0}$ to R nor $T_{i,0}$ to H_2. After it queries $m_{i,r}$ to H_1 (to receive response $H_{i,1}^*$), from inspection of the code one sees that the responses given to $A_{\rm g}$ for the queries to any of its oracle are random and independent of $H_{i,1}^*$, meaning 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, which gives us an equivalent game G_7 and justifies (30).

We then change the Initialize procedure to pick the random strings $S_{i,1}, T_{i,0}$ for all i at random independently from everything else, which cannot affect the output distribution of the game now since any information given to $A_{\rm g}$ is likewise random and independent from everything else, justifying (31).

Next we want to show that in each of the following cases, up to the point in its execution that the relevant bad_j 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 (again using the assumption that bad'_j for $j \neq j'$ has been set previously, which will *not* be mentioned below): 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 then we are done, because, first of all, these cases exhaust the possible execution sequences in which $A_{\rm g}$ queries $T_{i,0}$ for some i 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 $T_{w,0}$ (with w defined as in $\mathsf{GetQuery}$) just when such a query is made. Namely, in the games, the procedure to respond to queries to oracle H_2 explicitly checks whether a query to H_2

is equal to $T_{i,0}$ for some i, while algorithm GetQuery simply guesses at random that this is case for some particular such query. Since w is random and independent its guess has a $1/(q_{H_2}v(k))$ chance of being equal to $T_{w,0}$; Equation (32) then follows by Lemma D.1.

So let us examine each of these cases in turn. First consider when A_{g}, executed with G_3, queries $T_{i,0}$ for some i to H_2 after querying $S_{i,0}$ to R but without querying $m_{i,r}$ to H_1 at any point prior. We see directly from the code that the responses given to A_{g} to its oracle queries to up to the point in its execution that it queries $T_{i,0}$ to H_2 are random and independent from everything else. So the first case is settled. For the second case, consider when A_{g}, executed with game G_5, queries $T_{i,0}$ for some i to H_2 after having queried $m_{i,r}$ to H_1. Here we see that the only previous query was not answered independently of everything else in the game is query $m_{i,r}$ to H_1. But until A_{g} queries $T_{i,0}$ to H_2 it gets no information about $H^*_{i,1}$ because, as noted in the justification of (26), such information can only be obtained by computing $S_{i,0} \oplus m_{i,l}$, and information about $S_{i,0}$ cannot be obtained in this game without querying $T_{i,0}$ to H_2 since until this query is made all queries to oracle R are answered independently at random. Thus query $m_{i,r}$ 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 $T_{i,0}$ for some i to H_2, without previously having queried $S_{i,0}$ 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.

To finish the proof, note that it is straightforward to see the running-time analysis of I by taking into account Lemma D.1 and the convention that the running-time of A includes that of its overlying experiment: the extra overhead for GetQuery mostly consists of picking $v(k)$-many $l(k)$-bit random numbers and maintaining a counter up to at most q_{H_2}, incremented each time A_{g} makes a query to oracle H_2. ▌

procedure Initialize Game $G_1 - G_6$

$b \xleftarrow{\$} \{0,1\}$
$(t_0, \mathbf{x}_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(t_1, \mathbf{x}_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(f, f^{-1}) \xleftarrow{\$} \mathcal{F}_{\mathrm{RSA}}(1^k)$
For $i \leftarrow 1, \ldots, v(k)$
  Parse $\mathbf{x}_b[i]$ as $m_{i,l} \parallel m_{i,r}$,
    where $|m_{i,l}| = |m_{i,r}| = n(k)$
  $H^*_{i,1}, H^*_{i,2} \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$
  $R^*_i \xleftarrow{\$} \{0,1\}^{k_0(k)}$
  $S_{i,0} \leftarrow H^*_{i,1} \oplus m_{i,l}$; $T_{i,0} \leftarrow R^*_i \oplus m_{i,r}$
  $S_{i,1} \leftarrow H^*_{i,2} \oplus S_{i,0}$
  $\mathbf{y}[i] \leftarrow S_{i,1} \parallel f(T_{i,0})$
Return $(f, \mathbf{y})$

---

procedure Initialize Game G_8

$b \xleftarrow{\$} \{0,1\}$
$(t_0, \mathbf{x}_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$(f, f^{-1}) \xleftarrow{\$} \mathcal{F}_{\mathrm{RSA}}(1^k)$
For $i \leftarrow 1, \ldots, v(k)$
  $S_{i,1} \xleftarrow{\$} \{0,1\}^{n(k)-k_0(k)}$; $T_{i,0} \xleftarrow{\$} \{0,1\}^{k_0(k)}$
  $\mathbf{y}[i] \leftarrow S_{i,1} \parallel f(T_{i,0})$
Return $(f, \mathbf{y})$

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 $\exists i$ such that $x = m_{i,r}$ then
  If $H_{i,2}[T_0]$ is defined then
    $H_1[x] \leftarrow H^*_{i,1}$
  Else If $R[S_{i,0}]$ is defined then
    $\boxed{H_1[x] \leftarrow H^*_{i,1}}$
  Else $\boxed{\boxed{H_1[x] \leftarrow H^*_{i,1}}}$
Return $H_1[x]$

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 $\exists i$ such that $x = S_{i,0}$ then
  If $H_2[T_{i,0}]$ is defined then
    $R[x] \leftarrow R^*_i$
  Else If $H_1[m_{i,r}]$ is undefined then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{R[x] \leftarrow R^*_i}$
  Else $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boxed{\boxed{R[x] \leftarrow R^*_i}}$
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 $\exists i$ such that $x = T_{i,0}$ then
  If $R[S_{i,0}]$ is defined
    $\wedge\ H_1[m_{i,r}]$ is undefined then
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boxed{H_2[x] \leftarrow H^*_{i,2}}$
  If $H_1[m_{i,r}]$ is defined then
    $\mathsf{bad}_3 \leftarrow \mathsf{true}$; $\boxed{\boxed{H_2[x] \leftarrow H^*_{i,2}}}$
  Else $H_2[x] \leftarrow H^*_{i,2}$
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 $\exists i$ such that $x = T_{i,0}\ \wedge\ R[S_{i,0}]$ is undefined
  $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boxed{H_2[x] \leftarrow H^*_{i,2}}$
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.)