A preliminary version of this paper appears in *Advances in Cryptology - CRYPTO '07 Proceedings*. Lecture Notes in Computer Science, Vol. 4622, pp. 535–552, A. Menezes ed., Springer, 2007. This is the full version.

Deterministic and Efficiently Searchable Encryption

Mihir Bellare* Alexandra Boldyreva† Adam O'Neill‡

Abstract

We present as-strong-as-possible definitions of privacy, and constructions achieving them, for public-key encryption schemes where the encryption algorithm is *deterministic*. We obtain as a consequence database encryption methods that permit fast (i.e. sub-linear, and in fact logarithmic, time) search while provably providing privacy that is as strong as possible subject to this fast search constraint. One of our constructs, called RSA-DOAEP, has the added feature of being length preserving, so that it is the first example of a public-key cipher. We generalize this to obtain a notion of efficiently-searchable encryption schemes which permit more flexible privacy to search-time trade-offs via a technique called bucketization. Our results answer much-asked questions in the database community and provide foundations for work done there.

Keywords: Public-key encryption, deterministic encryption, searchable encryption, database security.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-Mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. Supported in part by NSF grants CNS-0524765, CNS-0627779, and a gift from Intel Corporation.

†School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA. E-mail: aboldyre@cc.gatech.edu. URL: http://www.cc.gatech.edu/∼ aboldyre. Supported in part by NSF CAREER award 0545659.

‡School of Computer Science, College of Computing, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, USA. E-mail: amoneill@cc.gatech.edu. URL: http://www.cc.gatech.edu/∼ amoneill. Supported in part by the grant of the second author.

Contents

1 Introduction

The classical notions of privacy for public-key encryption schemes, namely indistinguishability or semantic security under chosen-plaintext or chosen-ciphertext attack [35, 44, 47, 28, 10], can only be met when the encryption algorithm is randomized. This paper treats the case where the encryption algorithm is deterministic. We begin by discussing the motivating application.

FAST SEARCH. Remote data storage in outsourced databases is of increasing interest [51]. Data will be stored in encrypted form. (The database service provider is not trusted.) We are interested in a public key setting, where anyone can add to the database encrypted data which a distinguished "receiver" can retrieve and decrypt. The encryption scheme must permit search (by the receiver) for data retrieval. Public-key encryption with keyword search (PEKS) [16, 1, 18] is a solution that provably provides strong privacy but search takes time linear in the size of the database. Given that databases can be terabytes in size, this is prohibitive. The practical community indicates that they want search on encrypted data to be as efficient as on unencrypted data, where a record containing a given field value can be retrieved in time logarithmic in the size of the database. (For example, via appropriate tree-based data structures.) Deterministic encryption allows just this. The encrypted fields can be stored in the data structure, and one can find a target ciphertext in time logarithmic in the size of the database. The question is what security one can expect. To answer this, we need a definition of privacy for deterministic encryption.

A DEFINITION. One possibility is to just ask for one-wayness, but we would like to protect partial information about the plaintext to the maximum extent possible. To gauge what this could be, we note two inherent limitations of deterministic encryption. First, no privacy is possible if the plaintext is known to come from a small space. Indeed, knowing that c is the encryption under public key pk of a plaintext x from a set X, the adversary can compute the encryption c_x of x under pk for all $x \in X$, and return as the decryption of c the x satisfying $c_x = c$. We address this by only requiring privacy when the plaintext is drawn from a space of large min-entropy. Second, and more subtle, is that the ciphertext itself is partial information about the plaintext. We address this by only requiring non-leakage of partial information when the plaintext and partial information do not depend on the public key. This is reasonable because in real life public keys are hidden in our software and data does not depend on them. We provide a semantic-security style definition of privacy for deterministic encryption that takes these issues into account. While certainly weaker than the classical notions met by randomized schemes, our notion, which we call PRIV, is still quite strong. The next question is how to achieve this new notion.

CONSTRUCTIONS. Our first construction is generic and natural: Deterministically encrypt plaintext x by applying the encryption algorithm of a randomized scheme but using as coins a hash of (the public key and) x. We show that this "Encrypt-with-Hash" deterministic encryption scheme is PRIV secure in the random oracle (RO) model of [12] assuming the starting randomized scheme is IND-CPA secure. Our second construction is an extension of RSA-OAEP [13, 31]. The padding transform is deterministic but uses three Feistel rounds rather than the two of OAEP. RSA-DOAEP is proven PRIV secure in the RO model assuming RSA is one-way. This construction has the attractive feature of being length-preserving. (The length of the ciphertext equals the length of the plaintext.) This is important when bandwidth is expensive —senders in the database setting could be power-constrained devices— and for securing legacy code.

HISTORICAL CONTEXT. Diffie and Hellman [26] suggested that one encrypt plaintext x by applying to it an injective trapdoor function. A deterministic encryption scheme is just a family of injective

trapdoor functions, so our definition is an answer to the question of how much privacy Diffie-Hellman encryption can provide. (We clarify that not all trapdoor functions meet our definition. For example, plain RSA does not.)

In the symmetric setting, deterministic encryption is captured by ciphers including block ciphers. So far there has been no public key analog. Deterministic encryption meeting our definition provides one, and in particular RSA-DOAEP is the first length-preserving public-key cipher.

Efficiently searchable encryption. We introduce the notion of *efficiently searchable encryption* (ESE) schemes. These are schemes permitting fast (i.e. logarithmic time) search. Encryption may be randomized, but there is a deterministic function of the plaintext that can also be computed from the ciphertext and serves as a "tag," permitting the usual (fast) comparison-based search. Deterministic encryption schemes are a special case and the notion of security remains the same. (Our PRIV definition does not actually require encryption to be deterministic.) The benefit of the generalization is to permit schemes with more flexible privacy to search-time trade-offs. Specifically, we analyze a scheme from the database literature that we call "Hash-and-Encrypt." It encrypts the plaintext with a randomized scheme but also includes in the ciphertext a deterministic, collision-resistant hash of the plaintext. (This is an ESE scheme with the hash playing the role of the tag, and so permits fast search.) We prove that this scheme is PRIV secure in the RO model when the underlying encryption scheme is IND-CPA. With this scheme, loss of privacy due to lack of entropy in the plaintext space can be compensated for by increasing the probability of hash collisions. (This can be done, for example, by using truncated output of the hash function.) The trade-off is that the receiver then also gets "false positives" in response to a search query and must spend the time to sift through them to obtain the true answer. This technique is known as *bucketization* in the database literature, but its security was not previously rigorously analyzed.

Discussion. Our schemes only provide privacy for plaintexts that have high min-entropy. (This is inherent in being deterministic or efficiently searchable, not a weakness of our particular constructs.) We do not claim database fields being encrypted have high min-entropy. They might or they might not. The point is that practitioners have indicated that they will not sacrifice search time for privacy. Our claim is to provide the best possible privacy subject to allowing fast search. In some cases, this may very well mean no privacy. But we also comment that bucketization can increase privacy (at the cost of extra processing by the receiver) when the database fields being encrypted do not have high min-entropy.

Extensions. Our basic PRIV definition, and the above-mentioned results, are all for the CPA (chosen-plaintext attack) case. The definition easily extends to the CCA (chosen-ciphertext attack) case, and we call the resulting notion PRIV-CCA. Our Encrypt-with-Hash deterministic encryption scheme is not just PRIV, but in fact PRIV-CCA, in the RO model even if the underlying randomized encryption scheme is only IND-CPA, as long as the latter has the extra property that no ciphertext is too likely. In Section 7 we detail this and also discuss how RSA-DOAEP and Encrypt-and-Hash fare under CCA.

Open question. All our constructs are in the RO model. An important open question is to construct ESE or deterministic encryption schemes meeting our definition in the standard model. We note that in the past also we have seen new notions first emerge only with RO constructions achieving them, but later standard model constructs have been found. This happened for example for IBE [15, 54] and PEKS [18]. Note that the results of [33] rule out a standard model black-box reduction from deterministic public-key encryption to ordinary public-key encryption, but the former could still be built under other assumptions.

RELATED WORK. In the symmetric setting, deterministic encryption is both easier to define and to achieve than in the asymmetric setting. Consider the experiment that picks a random challenge bit b and key K and provides the adversary with a left-or-right oracle that, given plaintexts x_0, x_1 returns the encryption of x_b under K. Security asks that the adversary find it hard to guess b as long as its queries $(x_0^1, x_1^1), \ldots, (x_0^q, x_1^q)$ satisfy the condition that $x_0^1, \ldots, x_0^q$ are all distinct and also $x_1^1, \ldots, x_1^q$ are all distinct. To the best of our knowledge, this definition of privacy for deterministic symmetric encryption first appeared in [11]. However, it is met by a PRP and in this sense deterministic symmetric encryption goes back to [43].

Previous searchable encryption schemes provably meeting well-defined notions of privacy include [16, 36, 1, 6, 17, 18] in the public-key setting and [52, 34, 23] in the symmetric setting. However, all these require linear-time search, meaning the entire database must be scanned to answer each query. In the symmetric setting, further assumptions such as the data being known in advance, and then having the user (who is both the "sender" and "reciever" in this setting) pre-compute a specialized index for the server, has been shown to permit efficiency comparable to ours without sacrificing security [24]. Follow-on work to ours [4] treats ESE (as we mean it here) in the symmetric setting, providing the symmetric analog of what we do in our current paper.

Sub-linear time searchable encryption has been much targeted by the database security community [45, 3, 37, 25, 39, 40, 42, 38, 20, 53]. However, they mostly employ weak, non-standard or non-existing primitives and lack definitions or proofs of security. As a notable exception, Kantarcioglu and Clifton [41] recently called for a new direction of research on secure database servers aiming instead for "efficient encrypted database and query processing with *provable* security properties." They also propose a new cryptographic definition that ensures schemes reveal only the number of records accessed on each query, though a scheme meeting the definition requires tamper-resistant trusted hardware on the server.

Definitions that, like ours, restrict security to high min-entropy plaintexts have appeared before, specifically in the contexts of perfectly one-way probabilistic hash functions (POWHFs) [21, 22] and information-theoretically secure one-time symmetric encryption [49, 27]. The first however cannot be met by deterministic schemes, and neither handle the public-key related subtleties we mentioned above. (Namely that we must limit security to plaintexts not depending on the public key.) Also our definition considers the encryption of multiple related messages while those of [21, 22] consider only independent messages.

USE FOR OTHER APPLICATIONS. We note that one can also use our definitions to analyze other systems-security applications. In particular, a notion of "convergent encryption" is proposed in [2, 29] for the problem of eliminating wasted space in an encrypted file system by combining duplicate files across multiple users. Despite providing some correct intuition for security of their scheme, they are only able to formally show (for lack of a suitable security definition) that it achieves the very weak security notion of one-wayness. One can use our definitions to show that their scheme achieves much stronger security.

VERSIONS OF THIS PAPER AND CORRECTIONS. This full version of the paper corrects several typos and mistakes from the preliminary version [9], as well as includes all omitted proofs from the latter. Section 6, in particular the part of Section 6.1 treating bucketization, is significantly revised in this full version. We had to withdraw Theorem 4 of the proceedings version, which we are not able to prove. The latter is replaced here by Theorem 6.2. Although it does not cover as general a case as was claimed in the preliminary version, it still makes significant headway in this direction. (See Section 6.1 for more details.)

2 Notation and Conventions

Unless otherwise indicated, an adversary or algorithm may be randomized. An adversary is either an algorithm or a tuple of algorithms. In the latter case, we say it is polynomial time if each constituent algorithm is polynomial time. Unless otherwise indicated, an adversary or algorithm is polynomial time. By convention, the running-time of an adversary includes both its actual running-time and the time to run its overlying experiment. We let "$A(\ldots) \Rightarrow z$" denote the event that algorithm A outputs z when run on the elided arguments.

If x is a string then $|x|$ denotes its length in bits. We let $x[i \ldots j]$ denote bits i through j of string x, for $1 \le i \le j \le |x|$. By $x_1 \| \cdots \| x_n$ we denote an encoding of $x_1, \ldots, x_n$ from which $x_1, \ldots, x_n$ are uniquely recoverable. Vectors are denoted in boldface, for example $\mathbf{x}$. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes the number of components of $\mathbf{x}$ and $\mathbf{x}[i]$ denotes its ith component ($1 \le i \le |\mathbf{x}|$).

Asymmetric encryption. An (asymmetric) encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. The key-generation algorithm $\mathcal{K}$ takes input the unary encoding 1^k of the security parameter k to return a public key pk and matching secret key sk. The encryption algorithm $\mathcal{E}$ takes $1^k, pk$ and a plaintext x to return a ciphertext. The deterministic decryption algorithm $\mathcal{D}$ takes $1^k, pk, sk$ and a ciphertext c to return a plaintext. We require that $\mathcal{D}(1^k, pk, sk, c) = x$ for all k and all $x \in \mathrm{PtSp}(k)$, where the probability is over the experiment

$$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ c \xleftarrow{\$} \mathcal{E}(1^k, pk, x)$$

and PtSp is a plaintext space associated to Π. Unless otherwise indicated, we assume $\mathrm{PtSp}(k) = \{0,1\}^*$ for all k. We extend $\mathcal{E}$ to vectors via

Algorithm $\mathcal{E}(pk, \mathbf{x})$
  For $i = 1, \ldots, |\mathbf{x}|$ do $\mathbf{y}[i] \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathbf{x}[i])$
  Return $\mathbf{y}$

We say that Π is deterministic if $\mathcal{E}$ is deterministic.

Standard security notions. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let LR be the oracle that on input m_0, m_1, b returns m_b. We associate to an adversary B and a bit b the following:

Experiment $\mathbf{Exp}^{\text{ind-cpa-b}}_{\Pi,B}(k)$	**Experiment** $\mathbf{Exp}^{\text{ind-cca-b}}_{\Pi,B}(k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$	$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$
$d \xleftarrow{\$} B^{\mathcal{E}(1^k, pk, \mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$	$d \xleftarrow{\$} B^{\mathcal{E}(1^k, pk, \mathrm{LR}(\cdot,\cdot,b)), \mathcal{D}(1^k, pk, sk, \cdot)}(1^k, pk)$
Return d	Return d

We call B an *IND-CPA adversary* if every query m_0, m_1 it makes to its LR oracle satisfies $|m_0| = |m_1|$ and an *IND-CCA adversary* if in addition it never queries to $\mathcal{D}(1^k, pk, sk, \cdot)$ a ciphertext c previously returned by the LR oracle. For atk $\in \{\mathrm{cpa}, \mathrm{cca}\}$, the *advantage* of an IND-atk adversary B is

$$\mathbf{Adv}^{\text{ind-atk}}_{\Pi,B}(k) \;=\; \Pr\left[\, \mathbf{Exp}^{\text{ind-atk-1}}_{\Pi,B}(k) \Rightarrow 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\text{ind-atk-0}}_{\Pi,B}(k) \Rightarrow 1 \,\right].$$

A standard conditioning argument shows that

$$\mathbf{Adv}^{\text{ind-atk}}_{\Pi,B}(k) \;=\; 2 \cdot \Pr\left[\, \mathbf{Exp}^{\text{ind-atk-b}}_{\Pi,B}(k) \Rightarrow b \,\right] - 1\ ,$$

where the probability is over a random choice of bit b and the coins of the experiment. Π is said to be *IND*-atk *secure* if $\mathbf{Adv}^{\text{ind-atk}}_{\Pi,B}(\cdot)$ is negligible for every B.

Note that the definition of IND-CPA (or -CCA) allows an adversary to make as many queries as it likes to its LR-oracle. This is known to be equivalent (with loss in security by a factor less than or equal to the total number of LR-queries made) to allowing only one such query [8].

GAME-PLAYING. Our security analyses often employ the code-based game-playing technique of [14]. We recall some game-related language and conventions from [14] that we will use.

A game consists of an Initialize procedure, procedures that respond to an adversary's oracle queries, and a Finalize procedure. In a proof, one typically considers a sequence of games $G_1, \ldots, G_n$. The adversary is executed with each of these games, its execution with G_i being determined as follows. First, the Initialize procedure executes, and its outputs, as given by the Return statement, are passed as inputs to the adversary. Now the latter executes, oracle queries being answered by the procedures for this purpose associated to G_i. The output of the adversary 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 "$G_i^A \Rightarrow s$" denote the event that the output of Game G_i when executed with an adversary A is s. Both for the games and for the adversary, we adopt the convention that boolean variables are automatically initialized to false and arrays begin everywhere undefined.

3 Deterministic Encryption and its Security

PRIVACY ADVERSARIES. A privacy adversary $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ is a pair of algorithms. We clarify that $A_{\mathrm{m}}, A_{\mathrm{g}}$ share neither coins nor state. A_{m} takes input 1^k but *not* the public key, and returns a plaintext vector $\mathbf{x}$ together with some side information t. A_{g} takes input $1^k, pk$ and an encryption of $\mathbf{x}$ under pk, and tries to compute t.

The adversary must obey the following rules. First, there must exist functions $v(\cdot), n(\cdot)$ such that $|\mathbf{x}| = v(k)$ and $|\mathbf{x}[i]| = n(k)$ for k, all $(\mathbf{x}, t)$ output by $A_{\mathrm{m}}(1^k)$, and all $1 \leq i \leq v(k)$. Second, all plaintext vectors must have the same *equality pattern*, meaning for all $1 \leq i, j \leq v(k)$ there is a symbol $\diamond \in \{=, \neq\}$ such that $\mathbf{x}[i] \diamond \mathbf{x}[j]$ for all $(\mathbf{x}, t)$ output by $A_{\mathrm{m}}(1^k)$. We say that A has *min-entropy* $\mu(\cdot)$ if

$$\Pr\left[\, \mathbf{x}[i] = x \;:\; (\mathbf{x}, t) \xleftarrow{\$} A_{\mathrm{m}}(1^k) \,\right] \;\leq\; 2^{-\mu(k)}$$

for all $1 \leq i \leq v(k)$, all k, and all $x \in \{0,1\}^*$. We say that A has *high* min-entropy if $\mu(k) \in \omega(\log(k))$. The definition below is for chosen-plaintext attacks (CPA). In Section 7 we extend the definition to take chosen-ciphertext attacks (CCA) into account.

THE DEFINITION. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Although this is an important case of interest, the definition that follows does not assume Π to be deterministic. The definition is in the semantic-security style of [35]. Let A be a privacy adversary as above. We associate to A, Π the following:

Experiment $\mathbf{Exp}^{\text{priv-1}}_{\Pi,A}(k)$	**Experiment** $\mathbf{Exp}^{\text{priv-0}}_{\Pi,A}(k)$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$	$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$
$(\mathbf{x}_1, t_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$	$(\mathbf{x}_0, t_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(\mathbf{x}_1, t_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$\mathbf{c} \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathbf{x}_1)$	$\mathbf{c} \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathbf{x}_0)$
$g \xleftarrow{\$} A_{\mathrm{g}}(1^k, pk, \mathbf{c})$	$g \xleftarrow{\$} A_{\mathrm{g}}(1^k, pk, \mathbf{c})$
If $g = t_1$ then return 1	If $g = t_1$ then return 1
Else return 0	Else return 0

The *advantage* of a privacy adversary A against Π is

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

Again, a standard conditioning argument shows that

$$\mathbf{Adv}^{\text{priv}}_{\Pi,A}(k) \;=\; 2 \cdot \Pr\left[\,\mathbf{Exp}^{\text{priv-b}}_{\Pi,A}(k) \Rightarrow b\,\right] - 1 \;,$$

where the probability is over a random choice of bit b and the coins of the experiment. We say that Π is PRIV secure if $\mathbf{Adv}^{\text{priv}}_{\Pi,A}(\cdot)$ is negligible for every PTA A with high min-entropy.

As usual, in the random oracle (RO) model [12], all algorithms and adversaries are given access to the RO(s). In particular, both A_{m} and A_{g} get this access. Let us now discuss some noteworthy aspects of the new definition.

Access to the public key. If A_{m} were given pk, the definition would be unachievable for deterministic Π. Indeed, $A_{\mathrm{m}}(1^k)$ could output $(\mathbf{x}, t)$ where $\mathbf{x}[1]$ is chosen at random from $\{0,1\}^k$, $|\mathbf{x}| = 1$, and $t = \mathcal{E}(pk, \mathbf{x})$. Then $A_{\mathrm{g}}(1^k, pk, c)$ could return c, and A would have min-entropy k but

$$\mathbf{Adv}^{\text{priv}}_{\Pi,A}(k) \;\geq\; 1 - 2^{-k} \;.$$

Intuitively, the ciphertext is non-trivial information about the plaintext, showing that any deterministic scheme leaks information about the plaintext that depends on the public key. Our definition asks that information unrelated to the public key not leak. Note that this also means that we provide security only for messages unrelated to the public key, which is acceptable in practice because normal data is unlikely to depend on any public key. In real life, public keys are abstractions hidden in our software, not strings we look at.

Vectors of messages. The classical definitions explicitly only model the encryption of a single plaintext, but a simple hybrid argument from [8] shows that security when multiple plaintexts are encrypted follows. This hybrid argument fails in our setting, and in fact the two versions are not equivalent, which is why we have explicitly considered the encryption of multiple messages. To be more precise, let us say that Π is PRIV1 secure if $\mathbf{Adv}^{\text{priv}}_{\Pi,A}(\cdot)$ is negligible for every privacy adversary $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ satisfying

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

for all $k \in \mathbb{N}$. We claim that PRIV1 security does not imply PRIV security. To prove this, we now give an example of an encryption scheme $\overline{\Pi}$ which is PRIV1 secure but not PRIV secure. We obtain $\overline{\Pi} = (\mathcal{K}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ by modifying a given deterministic PRIV secure encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. Specifically, we define:

Algorithm $\overline{\mathcal{E}}(1^k, pk, x)$	**Algorithm** $\overline{\mathcal{D}}(1^k, pk, sk, y\|z)$
$y \leftarrow \mathcal{E}(1^k, pk, x)$	$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$
$z \leftarrow \mathcal{E}(1^k, pk, \overline{x})$	$x' \leftarrow \mathcal{D}(1^k, pk, sk, z)$
Return $y\|z$	If $x' = \overline{x}$ then return x
	Else return $\perp$

Above and in what follows, $\overline{s}$ denotes the bitwise complement of a string s. The assumption that Π is PRIV secure implies that $\overline{\Pi}$ is PRIV1 secure. However, the following attack shows $\overline{\Pi}$ is not PRIV secure. Consider $A_{\rm m}(1^k)$ that picks m_1, m_2 from $\{0,1\}^k$ and a bit b at random, and if $b = 1$ outputs $((m_1, \overline{m}_1), 1)$ and otherwise $((m_1, m_2), 0)$. $A_{\rm g}(1^k, pk, (y_1\|z_1, y_2\|z_2))$ outputs 1 if $z_1 = y_2$ and 0 otherwise. Then $A = (A_{\rm m}, A_{\rm g})$ has min-entropy k but advantage $1/2$.

The high min-entropy requirement. In the absence of the high-entropy restriction on A, it is clear that the definition would be unachievable for deterministic Π. To see this, consider $A_{\rm m}(1^k)$ that outputs $(0,0)$ with probability $1/2$ and $(1,1)$ with probability $1/2$. Then $A_{\rm g}(1^k, pk, c)$ could return 0 if $\mathcal{E}(pk, 0) = c$ and 1 otherwise, giving A an advantage of $1/2$. This reflects the fact that trial encryption of candidate messages is always a possible attack when encryption is deterministic.

Security for multiple users. The classical notions of privacy, as well as ours, only model a single user (SU) setting, where there is just one receiver and thus just one public key. An extension of the classical notions to cover multiple users, each with their own public key, is made in [8, 7], and these works go on to show that SU security implies multi-user (MU) security in this case. We leave it open to appropriately extend our definition to the MU setting and then answer the following questions: does SU security imply MU security, and do our schemes achieve MU security? But we conjecture that the answer to the first question is "no," while the answer to the second is "yes."

4 A Useful Fact

Define the *max public-key probability* $\mathsf{mpk}(\cdot)_{\mathsf{AE}}$ of AE as follows: for every k, we let $\mathsf{mpk}_{\mathsf{AE}}$ be the maximum, taken over all $w \in \{0,1\}^*$, of the quantity

$$\Pr\left[\, pk = w \;:\; (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k) \,\right] .$$

The following shows that $\mathsf{mpk}_{\mathsf{AE}}(\cdot)$ is negligible for any IND-CPA scheme.

Proposition 4.1 Let $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Then there is an IND-CPA adversary B such that

$$\mathsf{mpk}_{\mathsf{AE}} \;\leq\; \sqrt{\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B}} \,. \tag{1}$$

Furthermore, the running-time of B is at that for $O(1)$ computations of $\mathcal{K}, \mathcal{D}$, and B makes one LR-query. ∎

We clarify that (1) is a relationship between functions of k, so we are saying it holds for all $k \in \mathbb{N}$. For simplicity of notation we omit k here and further in the paper in similar instances.

Proof: Adversary B works as follows:

Algorithm $B^{\mathcal{E}(1^k,pk,\mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$
$(pk', sk') \xleftarrow{\$} \mathcal{K}(1^k)$
If $pk \neq pk'$ then return 0
Else $c \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathrm{LR}(0,1,b))$
$b' \leftarrow \mathcal{D}(1^k, pk', sk', c)$
Return b'

Then

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{AE},B} \Rightarrow 1\,\right] \;=\; \Pr\left[\,pk = pk'\,\right] \;\geq\; (\mathsf{mpk}_{\mathsf{AE}})^2\,.$$

$$\Pr\left[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{AE},B} \Rightarrow 1\,\right] \;=\; 0\,.$$

Subtracting, we get

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \;\geq\; (\mathsf{mpk}_{\mathsf{AE}})^2\,,$$

as desired. ∎

5 Secure Deterministic Encryption Schemes

We propose two constructions of deterministic schemes that we prove secure under our definition.

5.1 Encrypt-with-Hash

We first propose a generic deterministic encryption scheme that replaces the coins used by a standard encryption scheme with the hash of the message. More formally, let $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be any public-key encryption scheme. Say that $\mathcal{E}(1^k, pk, x)$ draws its coins from a set $\mathsf{Coins}_{pk}(|x|)$. We write $\mathcal{E}(1^k, pk, x; R)$ for the output of $\mathcal{E}$ on inputs pk, x and coins R. Let $H\colon \{0,1\}^* \to \{0,1\}^*$ be a hash function with the property that $H(pk\|x) \in \mathsf{Coins}_{pk}(|x|)$ for all $pk, x \in \{0,1\}^*$. The RO-model "*Encrypt-with-Hash*" deterministic encryption scheme $\mathsf{EwH} = (\mathcal{K}, \mathcal{DE}, \mathcal{DD})$ is defined via

Algorithm $\mathcal{DE}^H(1^k, pk, x)$	**Algorithm** $\mathcal{DD}^H(1^k, pk, sk, y)$
$R \leftarrow H(pk\|x)$	$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$
$y \leftarrow \mathcal{E}(1^k, pk, x; R)$	$R \leftarrow H(pk\|x)$
Return y	If $\mathcal{E}(1^k, pk, x; R) = y$ then return x
	Else Return $\perp$

The following implies that the Encrypt-with-Hash construction achieves PRIV-security if the starting encryption scheme is IND-CPA and has negligible max public-key probability.

Theorem 5.1 Suppose there is a privacy adversary $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ against EwH with min-entropy μ, which outputs vectors of size v with components of length n and makes at most q_{h} queries to its hash oracle. Then there exists an IND-CPA adversary B against AE such that

$$\mathbf{Adv}^{\text{priv}}_{\mathsf{EwH},A} \;\leq\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \;+\; \frac{2q_{\mathrm{h}}v}{2^{\mu}} \;+\; 8q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}}\,, \tag{2}$$

where $\mathsf{mpk}_{\mathsf{AE}}$ is the max public-key probability of AE. Furthermore, B makes v queries to its LR-oracle and its running-time is at most that of A plus $O(vn)$. ∎

The proof is readily obtained from the proof of Theorem 7.1, which is in Appendix D. (We comment in the latter in more detail about the differences.)

Although Proposition 4.1 shows that any IND-CPA encryption scheme has negligible max public-key probability, the reduction is not tight. For most specific schemes, one can in fact easily and unconditionally (meaning, without assumption) show that the max public-key probability is asymptotically small. For example, in ElGamal [30], the public key contains a value g^x, where x is a random exponent in the secret key. In this case, the max public-key probability is $1/|G|$, where $|G|$ is the order of the corresponding group.

5.2 RSA-DOAEP, A length-preserving deterministic scheme

It is sometimes important to minimize the number of bits transmitted over the network. We devise an efficient deterministic encryption scheme that is optimal in this regard, namely is length-preserving. (That is, the length of the ciphertext equals the length of the plaintext.) Length-preserving schemes can also be needed for securing legacy code. Ours is the first such construction shown secure under a definition of security substantially stronger than one-wayness, and in particular is the first construction of an asymmetric cipher.

RSA AND ITS SECURITY. We recall some background on the RSA function [48]. An *RSA trapdoor-permutation generator* security parameter k is an algorithm $\mathcal{F}$ that on input 1^k returns $(N,e),(N,d)$ where N is the product of two distinct $k/2$-bit primes and $ed \equiv 1 \bmod \phi(N)$. (Here $\phi(\cdot)$ is Euler's phi function.) An *inverter* I against $\mathcal{F}$ is algorithm that takes input $1^k,(N,e),x^e \bmod \phi(N)$ and tries to compute x. RSA trapdoor permutation generator $\mathcal{F}$ is *one-way* if for every I the function:

$$\mathbf{Adv}^{\mathrm{owf}}_{\mathcal{F},I}(k) \;=\; \Pr\Big[\, x = x' \;:\; ((N,e),(N,d)) \xleftarrow{\$} \mathcal{F}(1^k)\,;\; x \xleftarrow{\$} \{0,1\}^k\,;\; x' \xleftarrow{\$} I(1^k,(N,e),x^e) \,\Big]$$

is negligible.

THE SCHEME. Our construction is based on RSA-OAEP [13, 31]. But in place of the randomness in this scheme we use part of the message, and we add an extra round to the underlying transform.

Formally, the scheme is parameterized by functions $k_0(\cdot), k_1(\cdot) > 0$. The plaintext space $\mathrm{PtSp}(k)$ consists of all strings of length at least $\max(k_1(k), 2k_0(k)+1)$. We assume here for simplicity that all messages to encrypt have a fixed length $n = n(k)$ satisfying $n > 2k_0$ and $n \geq k_1$. Let $\mathcal{F}$ be an RSA trapdoor-permutation generator with modulus length to $k_1 = k$. The key-generation algorithm of the associated RO-model deterministic encryption scheme RSA-DOAEP ("D" for "deterministic") runs $\mathcal{F}$ and returns (N,e) as the public key and (N,d) as the secret key. Its encryption and decryption algorithms have oracle access to functions $H_1, H_2\colon \{0,1\}^* \to \{0,1\}^{k_0}$ and $R\colon \{0,1\}^* \to \{0,1\}^{n-k_0}$, and are defined as follows (see Figure 1 for an illustration of the encryption algorithm):

Algorithm $\mathcal{E}^{H_1,H_2,R}(1^k,(N,e),x)$
- $x_l \leftarrow x[1\ldots k_0]$
- $x_r \leftarrow x[k_0+1\ldots n]$
- $s_0 \leftarrow H_1((N,e)\|x_r) \oplus x_l$
- $t_0 \leftarrow R((N,e)\|s_0) \oplus x_r$
- $s_1 \leftarrow H_2((N,e)\|t_0) \oplus s_0$
- $p_l \leftarrow (s_1\|t_0)[1\ldots n-k_1]$
- $p_r \leftarrow (s_1\|t_0)[n-k_1+1\ldots n]$
- $c \leftarrow p_l\|(p_r^e \bmod N)$
- Return c

Algorithm $\mathcal{D}^{H_1,H_2,R}(1^k,(N,e),(N,d),c)$
- $p_1 \leftarrow c[1\ldots n-k_1]$
- $y \leftarrow c[n-k_1+1\ldots n]$
- $p \leftarrow p_1\|(y^d \bmod N)$
- $s_1 \leftarrow p[1\ldots k_0]$
- $t_0 \leftarrow p[k_0+1\ldots n]$
- $s_0 \leftarrow H_2((N,e)\|t_0) \oplus s_1$
- $x_r \leftarrow R((N,e)\|s_0) \oplus t_0$
- $x_l \leftarrow H_1((N,e)\|x_r) \oplus s_0$
- Return $x_l\|x_r$

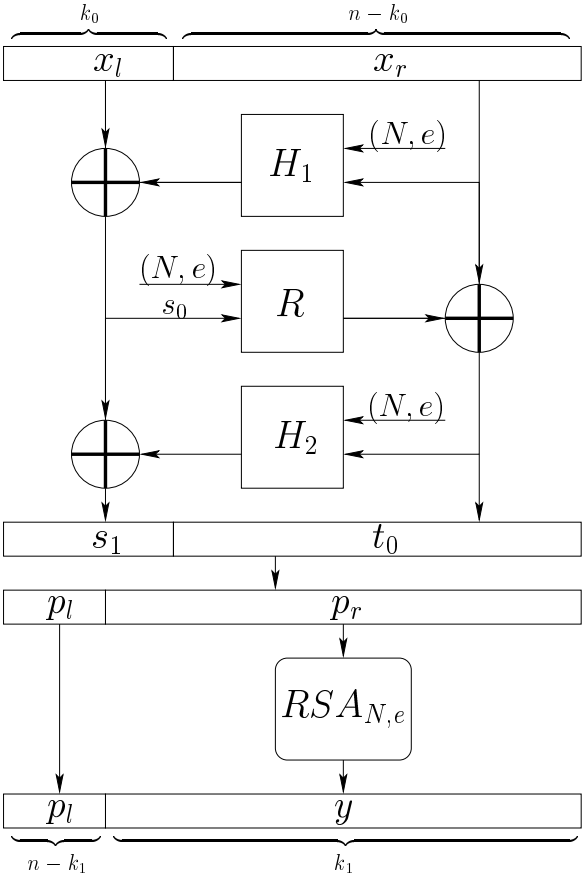

Figure 1: An illustration of the encryption algorithm of RSA-DOAEP.

SECURITY. The following implies that RSA-DOAEP achieves PRIV-security if RSA is one-way.

Theorem 5.2 Suppose there exists a privacy adversary $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ against RSA-DOAEP with min-entropy μ that makes at most $q_{\mathrm{h_i}}$ queries to oracle H_i for $i \in \{1, 2\}$ and q_{r} to oracle R, and outputs vectors of size v with components of length n. Let $q_{\mathrm{tot}} = q_{\mathrm{h}_1} + q_{\mathrm{r}} + q_{\mathrm{h}_2}$. We consider two cases:
1

- Case 1: $n < k_0 + k_1$. Then there is an inverter I against RSA trapdoor-permutation generator $\mathcal{F}$ such that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{RSA\text{-}DOAEP},A} \quad \leq \quad & q_{\mathrm{h}_2} v \cdot \sqrt{\mathbf{Adv}^{\mathrm{owf}}_{\mathcal{F},I} + 2^{2k_1 - 4(n-k_0)+5}} \\ & + 2^{k_1 - 2(n-k_0)+5} \; + \; \frac{2q_{\mathrm{r}} v}{2^{k_0}} \; + \; \frac{2q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} \; + \; 4q_{\mathrm{tot}} \cdot \frac{\ln(2^{k_1/2} - 1)}{2^{k_1/2} - 1} \; . \quad (3)\end{aligned}$$

  Furthermore the running-time of I is at most twice that of A plus $O(\log v + q_{\mathrm{h}_2} \log q_{\mathrm{h}_2} + k_1^3)$.

- Case 2: $n \geq k_0 + k_1$. Then there is an inverter I against RSA trapdoor-permutation generator $\mathcal{F}$ such that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{RSA\text{-}DOAEP},A} \quad \leq \quad & v \cdot \mathbf{Adv}^{\mathrm{owf}}_{\mathcal{F},I} \\ & + \frac{2q_{\mathrm{r}} v}{2^{k_0}} \; + \; \frac{2q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} \; + \; 4q_{\mathrm{tot}} \cdot \frac{\ln(2^{k_1/2} - 1)}{2^{k_1/2} - 1} \; .\end{aligned}$$

  Furthermore, the running-time of I is at most that of A plus $O(\log v + q_{\mathrm{h}_2} \log q_{\mathrm{h}_2})$. ∎

The proof is in Appendix A.

In practice, we will have e.g. $k_1 = 1024$, and then we can set parameter k_0 to, say, 160 bits to effectively maximize security in either case of the theorem. Then, for a given application, the relation between $n - 160$ and 1024 then determines which case of the theorem applies. We note that the weaker security guarantee in Case 1 is analogous to the state-of-the-art for RSA-OAEP itself [31, 46].

ENCRYPTING LONG MESSAGES. Typically, to encrypt long messages efficiently using an asymmetric scheme in practice, one employs hybrid encryption. This methodology in particular applies to the Encrypt-with-Hash construction, in which the starting scheme can be a hybrid one. However, if using hybrid encryption, RSA-DOAEP would no longer be length-preserving (since an encrypted symmetric key would need to be included with the ciphertext). We therefore stress that one can efficiently encrypt long messages using RSA-DOAEP *without* making use of hybrid encryption. Intuitively, this is possible because the underlying Feistel network in the scheme acts like an "all-or-nothing transform" [19], such that unless an adversary with high min-entropy inverts the RSA image in a ciphertext then it cannot recover any information about the (long) message.

6 Efficiently Searchable Encryption (ESE)

We now turn to the application to outsourced databases, where data is sent to a remote server. The database server is untrusted. The data in each field in the database is to be encrypted separately under the public key of a receiver, who needs to be able to query the server and retrieve encrypted records containing particular data efficiently. Encrypting the data deterministically provides a possible solution to this problem. The database can be then organized into a standard (e.g. tree-based) data structure, and a query, i.e. a ciphertext, specifies the exact piece of data the server needs to locate, so the sever can find it (in say logarithmic time) just like for unencrypted data. In general, though, encryption permitting efficient search does not to be deterministic *per se*. Accordingly, we first define a more general primitive we call *efficiently searchable encryption (ESE)*.

THE NEW PRIMITIVE. The idea is to associate a (deterministic) "tag" to each plaintext, which can be computed both from the plaintext and from a ciphertext that encrypts it. The server can compute the tag of a ciphertext to be stored in the database and use it to store the ciphertext appropriately in a data structure. To query a plaintext, the receiver can compute and send its tag, which the server can look up in the data structure, returning any matches and associated data. (The receiver can then decrypt these matches and filter out any false-positives, i.e. ciphertexts with the same tag as the query but which do not decrypt to the intended plaintext.) These two functionalities are captured by the functions F, G below.

Let $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme with associated plaintext space $\mathrm{PtSp}(\cdot)$. We say AE is a *δ-efficiently searchable encryption (-ESE)* scheme for some function $\delta(\cdot) < 1$ if there exist deterministic algorithms F, G such that for every k we have

1. Perfect Consistency: For every k and $x_1 \in \mathrm{PtSp}(k)$, the probability that the following experiment returns 1 is 1:

$$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ c \xleftarrow{\$} \mathcal{E}(1^k, pk, x_1)$$
$$\text{If } F(pk, x_1) = G(pk, c) \text{ then return } 1\ \ ;\ \text{Else return } 0$$

2. Computational Soundness: For every k and every algorithm $\mathcal{M}$ that on input 1^k outputs a pair of distinct messages in $\mathrm{PtSp}(k)$, the probability that the following experiment returns 1 is at

most $\delta(k)$:

$$(x_0, x_1) \xleftarrow{\$} \mathcal{M}(1^k)\ ;\ (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ c \xleftarrow{\$} \mathcal{E}(1^k, pk, x_1)$$
$$\text{If } F(pk, x_0) = G(pk, c) \text{ then return 1 } ;\ \text{Else return 0}$$

We refer to the output of F, G as the *tag* of the message or a corresponding ciphertext.

Above, consistency ensures that the server locates *at least* the desired ciphertext(s) on a query, because their tags and those of the plaintext used to form the query are the same. Soundness limits the number of false-positives located as well; if using a δ-ESE scheme, for a size n database we expect to have at most $\delta \cdot n$ false-positives on a query, over the choice of data in the database and of the query (both not depending on the remaining), and key-generation and encryption computations. The reasoning for this is as follows. Given query $F(pk, x)$, define for each ciphertext c in the database such that $\mathcal{D}(1^k, pk, sk, c) \neq x$ the random variable X_c taking value 1 if $F(pk, x) = G(pk, c)$ and 0 otherwise. Define $X = \sum_c X_c$. The expected number of false positives on the query is then

$$\mathbf{E}[\,X\,] \;=\; \sum_c \mathbf{E}[\,X_c\,] \;\leq\; \sum_c \delta \;\leq\; \delta \cdot n\ ,$$

where the equality is by linearity of expectation and the first inequality follows from soundness.

Indeed, any deterministic encryption scheme is a 0-ESE scheme under our definition. Hence the two deterministic encryption schemes we studied, namely Encrypt-with-Hash and RSA-DOAEP, represent two particular 0-ESE schemes. To see this, let $\mathsf{DPE} = (\mathcal{K}, \mathcal{DE}, \mathcal{D})$ be any deterministic public-key encryption scheme. Then letting F and G be the algorithms that on inputs pk, m and $pk, \mathcal{DE}(pk, m)$, respectively, return $\mathcal{DE}(pk, m)$, we see that DPE is 0-efficiently-searchable. That is, the function δ is identically 0 here, due consistency of encryption.

However, for other ESE schemes δ may be large. This is why $\mathcal{M}$ is not given input pk in the soundness condition; if it were, the condition would not make sense for large δ, because then $\mathcal{M}$ could compute tags of messages itself and just output two it finds to agree on their tags. As in our definition of privacy, this restriction is reasonable because real data and queries are unlikely to depend on any public key.

Security of ESE. As our PRIV does not actually require the encryption scheme in question to be deterministic, we can also use it for ESE. Here the rule that a privacy adversary output vectors with the same equality pattern means, intuitively, that we allow the server to learn only which records in the database contain the same field values and how many times each such value occurs (called the *occurrence profile* of the data), so that it is able to efficiently search. In contrast to deterministic encryption, we will see that using an ESE scheme with large δ can increase privacy (at the cost of more processing by the receiver on a query) when database fields being encrypted do not have high min-entropy.

6.1 Encrypt-and-Hash ESE

We formalize and analyze a probabilistic ESE scheme based on an approach from the database literature in which one appends a (deterministic) hash of the plaintext to a ciphertext to enable efficient search. Let $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be any pubic-key encryption scheme and $H\colon \{0,1\}^* \to \{0,1\}^l$ for some $l > 0$ be a hash function. The RO-model "*Encrypt-and-Hash*" encryption scheme $\mathsf{EaH} = (\mathcal{K}, \mathcal{HE}, \mathcal{HD})$ is defined via

Algorithm $\mathcal{HE}^H(1^k, pk, x)$	**Algorithm** $\mathcal{HD}^H(1^k, pk, sk, y\|h)$
$h \leftarrow H(pk\|x)$	$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$
$y \leftarrow \mathcal{E}(1^k, pk, x)$	$h' \leftarrow H(pk\|x)$
Return $y\|h$	If $h' = h$ then return x
	Else Return $\perp$

Then EaH is a 2^{-l}-ESE scheme. For this, we let F, G be the algorithms that on inputs pk, x and $pk, \mathcal{E}(pk, x)\|H(pk\|x)$, respectively, return the tag $H(pk\|x)$; the fact that δ is identically 2^{-l} here follows from the fact that H is a RO with output length l.

The following implies that this construct is PRIV secure if the underlying encryption scheme is IND-CPA, independent of l.

Theorem 6.1 Suppose there is a privacy adversary $A = (A_{\rm m}, A_{\rm g})$ against EaH that outputs vectors of size v and makes at most $q_{\rm h}$ queries to its hash oracle. Then there exists an IND-CPA adversary B against AE such that

$$\mathbf{Adv}^{\rm priv}_{\mathsf{EaH},A} \quad \leq \quad 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \; + \; \frac{q_{\rm h} v}{2^{\mu}} \; + \; q_{\rm h} \cdot \mathsf{mpk}_{\mathsf{AE}} \; ,$$

where $\mathsf{mpk}_{\mathsf{AE}}$ is the max public-key probability of AE. Furthermore, B makes v queries to its LR-oracle and its running-time is at most that of A. ▮

The proof is in Appendix B.

Bucketization. Theorem 6.1 tells us that EaH achieves PRIV if plaintexts being encrypted have high min-entropy. But in the case that they do not, one can try to compensate for this by decreasing the length of the hash l (at the cost of more processing by the receiver on a query), resulting in more plaintexts with the same tag. This technique is known as *bucketization* in the database literature. Intuitively, the scheme may then provide some privacy, even for small message spaces, if an efficient adversary cannot distinguish with too high a probability between ciphertexts whose tags are equal.

In the proceedings version of this paper [9], we incorrectly claimed a general bound on the advantage of a privacy adversary A against EaH with hash length $l > 0$. In fact, our analysis applies only to case that the adversary outputs vectors of size 1 and is interested in recovering the entire message from the ciphertext. The corrected theorem is given below; we leave it open to extend our techniques to the more general case.

Say $A = (A_{\rm m}, A_{\rm g})$ is a *message-recovery* (MR) privacy adversary if the output of $A_{\rm m}$ is always of the form (x, x) for a string x. Suppose $A_{\rm m}$ of a MR privacy adversary A puts a distribution $\mathfrak{D}$ on a finite set of plaintexts S. Let $S = \{m_1, \ldots m_M\}$ and p_i be the probability assigned to m_i by $\mathfrak{D}$, meaning $A_{\rm m}$ outputs (m_i, m_i) with probability p_i. Intuitively, one cannot expect bucketization to increase privacy in all cases because $\mathfrak{D}$ may be too concentrated at a particular m_i. Then, if the adversary sees a ciphertext whose tag is the same as the hash of m_i, it can deduce the latter decrypts to m_i with very high probability. But we show that as long as $\mathfrak{D}$ is sufficiently "spread out" over S then bucketization can indeed increase privacy, even if $|S|$ is not too large. The metric we use to measure this is the *collision probability* $\mathsf{Coll}[\,\mathfrak{D}\,]$ of $\mathfrak{D}$, which is defined as $\sum_{i=1}^{M} p_i^2$.

Theorem 6.2 Suppose there is a MR privacy adversary $A = (A_{\rm m}, A_{\rm g})$ such that $A_{\rm m}$ puts distribution $\mathfrak{D}$ on a finite set of plaintexts S. Then there exists an IND-CPA adversary B against AE such that

$$\mathbf{Adv}^{\rm priv}_{\mathsf{EaH}',A} \quad \leq \quad 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \; + \; 12 \cdot \mathsf{Coll}[\,\mathfrak{D}\,] \cdot |S| \cdot \frac{2^{-\mu} \cdot 2^{l}}{(1 - 2^{-\mu})^2} \; . \tag{4}$$

Furthermore, B makes one query to its LR-oracle and its running-time is at most that of A. ▮

The proof is in Appendix C.

Note that it follows from the Cauchy-Schwarz inequality that $1/|S| \leq \mathsf{Coll}[\mathfrak{D}]$ above, with equality just when $A_{\rm m}$ puts a uniform distribution on S. So, the closer the collision probability of $\mathfrak{D}$ is to that of the uniform distribution on S, the better our bound. Indeed, it can be shown that the closer the former is to the latter, the "closer" $\mathfrak{D}$ is to the uniform distribution on S itself.

We remark that one cannot use a POWHF [21, 22] to compute the tags in place of the RO in the Encrypt-and-Hash construction, because POWHFs are randomized and this will violate the consistency requirement of ESE.

7 CCA and Other Extensions

Our definition, and so far our security proofs, are for the CPA case. Here we discuss extensions to the CCA case and then other extensions such as to hash functions rather than encryption schemes.

PRIV-CCA. Extend $\mathbf{Exp}^{\text{priv-b}}_{\Pi,A}(k)$ to give $A_{\rm g}$ on input $1^k, pk, \mathbf{c}$ oracle access to $\mathcal{D}(1^k, pk, sk, \cdot)$, for $b \in \{0,1\}$, which it can query on any string not appearing as a component of $\mathbf{c}$. Note that $A_{\rm m}$ does *not* get this decryption oracle. Then let

$$\mathbf{Adv}^{\text{priv-cca}}_{\Pi,A}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\text{priv-cca-1}}_{\Pi,A}(k) \Rightarrow 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{priv-cca-0}}_{\Pi,A}(k) \Rightarrow 1\,\right] .$$

Again, a standard conditioning argument shows that

$$\mathbf{Adv}^{\text{priv-cca}}_{\Pi,A}(k) \;=\; 2 \cdot \Pr\left[\,\mathbf{Exp}^{\text{priv-cca-b}}_{\Pi,A}(k) \Rightarrow b\,\right] - 1 \;,$$

We say that Π is *PRIV-CCA secure* if $\mathbf{Adv}^{\text{priv-cca}}_{\Pi,A}(\cdot)$ is negligible for every A with high min-entropy.

Encrypt-with-Hash. Deterministic encryption scheme EwH is PRIV-CCA secure even if the starting encryption scheme is only IND-*CPA* but meets an extra condition, namely that no ciphertext occurs with too high a probability. More precisely, the *max-ciphertext probability* $\mathsf{mc}(\cdot)$ of $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined as follows: we let $\mathsf{mc}_{\mathsf{AE}}(k)$ be the maximum taken over all $y \in \{0,1\}^*$ and all $x \in \mathrm{PtSp}(k)$ of the quantity

$$\Pr\left[\,(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\;;\; c \xleftarrow{\$} \mathcal{E}(1^k, pk, x) \;:\; c = y\,\right] .$$

Then Theorem 5.1 extends as follows.

Theorem 7.1 Suppose there is a PRIV-CCA adversary $A = (A_{\rm m}, A_{\rm g})$ against EwH with min-entropy μ, which outputs vectors of size v with components of length n and makes at most $q_{\rm h}$ queries to its hash oracle and at most $q_{\rm d}$ queries to its decryption oracle. Let $\mathsf{mpk}_{\mathsf{AE}}$ and $\mathsf{mc}_{\mathsf{AE}}$ be max public-key and max-ciphertext probabilities of AE, respectively. Then there exists an IND-CPA adversary B against AE such that

$$\mathbf{Adv}^{\text{priv-cca}}_{\mathsf{EwH},A} \;\leq\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \;+\; \frac{2q_{\rm h}v}{2^{\mu}} \;+\; 8q_{\rm h} \cdot \mathsf{mpk}_{\mathsf{AE}} \;+\; 2q_{\rm d} \cdot \mathsf{mc}_{\mathsf{AE}} \;. \tag{5}$$

Furthermore, B makes v queries to its LR-oracle and its running-time is at most that of A plus $O(vn + q_{\rm h}T_{\mathcal{E}})$, where $T_{\mathcal{E}}$ is the time for one computation of $\mathcal{E}$ on a message of length n. ▮

The proof is given in Appendix D.

The requirement that $\mathsf{mc}_{\mathsf{AE}}(\cdot)$ be small is quite mild. Most practical encryption schemes have negligible max-ciphertext probability. For example, the "no-authenticity," IND-CPA version of RSA-OAEP [14] has max-ciphertext probability equal to $1/2^{n(k)}$, where $n(\cdot)$ is the length of the messages to encrypt, and ElGamal [32] has max-ciphertext probability $1/|G|$, where G is the used group. Thus in practice this does not amount to any extra assumption. Moreover, in general any IND-CPA scheme $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ can be easily modified to achieve this property. Specifically, define:

Algorithm $\mathcal{E}^*(1^k, pk, x)$	**Algorithm** $\mathcal{D}^*(1^k, pk, sk, c\|r)$
$r \xleftarrow{\$} \{0,1\}^k$	$x \leftarrow \mathcal{D}(1^k, pk, sk, c)$
$c \xleftarrow{\$} \mathcal{E}(1^k, pk, x)$	Return x
Return $c\|r$	

IND-CPA security of AE^* follows from IND-CPA security of AE, but $\mathsf{mc}_{\mathsf{AE}^*} = 2^{-k}$.

On the other hand, IND-CPA security does not imply low max-ciphertext probability in general. To prove this, we start with an IND-CPA scheme $\mathsf{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and modify it to obtain a scheme AE' that is still IND-CPA but has high max-ciphertext probability (namely $\mathsf{mc}_{\mathsf{AE}'} = 1$). Scheme $\mathsf{AE}' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ works as follows:

Algorithm $\mathcal{E}'(1^k, pk, x)$	**Algorithm** $\mathcal{D}'(1^k, pk, sk, c)$
$y \xleftarrow{\$} \mathcal{E}(1^k, pk, 0)$	If $c = 0$ then return sk
$z \leftarrow \mathcal{D}(1^k, x, y)$	Else return $\mathcal{D}(1^k, pk, sk, c)$
If $z = 0$ then return 0	
Else return $\mathcal{E}(1^k, pk, x)$	

Above, we assume that "0" is a special ciphertext not output by $\mathcal{E}(1^k, pk, \cdot)$ for all $k \in \mathbb{N}$ and pk output by $\mathcal{K}(1^k)$. It is easy to see that AE' is IND-CPA but $\mathsf{mc}_{\mathsf{AE}'} = 1$.

RSA-DOAEP. Scheme RSA-DOAEP is not PRIV-CCA, because there is a chosen-ciphertext attack against it. Denote by DOAEP the three-round Feistel network underlying the scheme and DOAEP^{-1} its inverse. Consider PRIV-CCA adversary $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ against RSA-DOAEP that works as follows:

Adversary $A_{\mathrm{m}}(1^k)$	**Adversary** $A_{\mathrm{g}}^{\mathcal{D}(1^k,(N,e),(N,d),\cdot)}(1^k, (N,e), c)$
$x \xleftarrow{\$} \{0,1\}^n$	$y \xleftarrow{\$} \{0,1\}^{k_1}$
Return (x, x)	$c' \leftarrow c[1 \ldots n - k_1] \| (c[n - k_1 + 1 \ldots n] \cdot y^e \bmod N)$
	$x' \leftarrow \mathcal{D}(1^k, (N,e), (N,d), c')$
	$z' \leftarrow \mathsf{DOAEP}(x')$
	$s \leftarrow z'[n - k_1 + 1 \ldots n] \cdot y^{-1} \bmod N$
	Return $\mathsf{DOAEP}^{-1}(z)$

Above, n is as defined in Subsection 5.2, and, for simplicity, we do not explicitly denote RO access of the algorithms. Adversary A is efficient and has min-entropy n but PRIV-CCA advantage $1 - 1/2^n$.

However, it is straightforward to show that when properly combined in the "encrypt-then-sign" fashion with a secure digital signature, RSA-DOAEP *does* achieve PRIV-CCA in a natural "outsider security" model analogous to that in [5]. This may come at no additional cost, for example in the outsourced database application, which also requires authenticity anyway.

EXTENSIONS TO OTHER PRIMITIVES. One can apply (as we do in the proof of Theorem 6.2) our PRIV definition to a more general primitive that we call a (public-key) *hiding scheme*, which we define as a pair $\mathsf{HIDE} = (\mathsf{Kg}, \mathsf{F})$ of algorithms, where Kg on input 1^k outputs a key K and F takes $1^k, K$ and an input x to return an output we call the ciphertext. Note that every public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ has an associated hiding scheme where Kg runs $\mathcal{K}$ to receive output (pk, sk) and then returns pk, and $\mathsf{F}(1^k, K, \cdot)$ is the same as $\mathcal{E}(1^k, pk, \cdot)$. In general, though, a hiding scheme is not required to be invertible, covering for example the case of hash functions.

ENCRYPT-AND-HASH. PRIV-CCA security of ESE scheme EaH requires IND-CCA security of the starting encryption scheme AE in general, in which case the analogous statements to the theorems in Section 6.1 hold when considering CCA attacks. (In other words, if the starting encryption scheme is IND-CCA then EaH is PRIV-CCA, and more specifically the same bounds as in the theorems hold for PRV-CCA adversaries in this case.) We comment in the proofs of the former how on the proofs change in the CCA case.

In fact, the basic construct generalizes to using any deterministic hiding scheme $\mathsf{HIDE} = (\mathsf{Kg}, \mathsf{F})$ as defined above in place of the RO in EaH, where we replace a query $H(pk, x)$ in the scheme by $\mathsf{F}(K, (pk, x))$. Let us call the resulting scheme $\mathsf{EaH}_{\mathsf{HIDE}}$. Theorem 6.1 then generalizes as follows.

Theorem 7.2 Suppose there is a privacy adversary $A = (A_{\mathrm{m}}, A_{\mathrm{g}})$ against $\mathsf{EaH}_{\mathsf{HIDE}}$ that outputs vectors of size v. Then there exists an IND-CPA adversary B against AE and a privacy adversary A' against HIDE such that

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH}_{\mathsf{HIDE}},A} \quad \leq \quad 2 \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \; + \; \mathbf{Adv}^{\mathrm{priv}}_{\mathsf{HIDE},A'} \; .$$

Furthermore, B makes v queries to its LR-oracle, A' outputs vectors of length v with components of length n, and the running-times of A', B are at most that of A. ▌

Proof: The proof is similar to the proof of Theorem 6.1, in Appendix B. We define three adversaries, namely $B, A' = (A'_{\mathrm{m}}, A'_{\mathrm{g}}), B'$ as follows:

Adv $B^{\mathcal{E}(1^k, pk, \mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$	**Adv** $A'_{\mathrm{m}}(1^k)$	**Adv** $B'^{\mathcal{E}(1^k, pk, \mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$
$K \xleftarrow{\$} \mathsf{Kg}(1^k)$	$(\mathbf{x}, t) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$	$K \xleftarrow{\$} \mathsf{Kg}(1^k)$
$(\mathbf{x}_1, t_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$	Return $(\mathbf{x}, t)$	$(\mathbf{x}_0, t_0) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$; $(\mathbf{x}_1, t_1) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$\mathbf{c} \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathrm{LR}(\mathbf{0}, \mathbf{x}_1, b))$		$\mathbf{c} \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathrm{LR}(\mathbf{0}, \mathbf{x}_0, b))$
$\mathbf{y} \leftarrow \mathsf{F}(1^k, K, \mathbf{x}_1)$	**Adv** $A'_{\mathrm{g}}(1^k, pk, \mathbf{c})$	$\mathbf{y} \leftarrow \mathsf{F}(1^k, K, \mathbf{x}_0)$
For $i = 1$ to v do:	$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$	For $i = 1$ to v do:
$\quad \mathbf{c}'[i] \leftarrow \mathbf{c}[i] \| \mathbf{y}[i]$	$\mathbf{y} \xleftarrow{\$} \mathcal{E}(pk, \mathbf{0})$	$\quad \mathbf{c}'[i] \leftarrow \mathbf{c}[i] \| \mathbf{y}[i]$
$g \xleftarrow{\$} A_{\mathrm{g}}(1^k, (pk, K), \mathbf{c}')$	For $i = 1$ to v do:	$g \xleftarrow{\$} A_{\mathrm{g}}(1^k, (pk, K), \mathbf{c}')$
If $g = t_1$ then return 1	$\quad \mathbf{c}'[i] \xleftarrow{\$} \mathbf{y}[i] \| \mathbf{c}[i]$	If $g = t_1$ then return 1
Else return 0	$g \xleftarrow{\$} A_{\mathrm{g}}(1^k, (pk, K), \mathbf{c}')$	Else return 0
	Return g	

The proof is a hybrid argument. Consider modified schemes $\mathsf{EaH}_i = (\mathcal{K}', \mathcal{E}_i)$ for $i \in \{1, 2, 3, 4\}$ defined as follows. Each EaH_i has the same key-generation algorithm, namely $\mathcal{K}'$, which on input 1^k outputs (pk, K) where $(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$ and $K \xleftarrow{\$} \mathsf{Kg}(1^k)$. The encryption algorithms operate

on vectors of size v, and are defined as follows:

$$\begin{aligned}
\mathcal{E}_1(1^k,(pk,K),\mathbf{x}) &= \mathcal{E}(1^k,pk,\mathbf{x})\|\mathsf{F}(1^k,K,\mathbf{x}) \\
\mathcal{E}_2(1^k,(pk,K),\mathbf{x}) &= \mathcal{E}(1^k,pk,\mathbf{0})\|\mathsf{F}(1^k,K,\mathbf{x}) \\
\mathcal{E}_3(1^k,(pk,K),\mathbf{x}) &= \mathcal{E}(1^k,pk,\mathbf{0})\|\mathsf{F}(1^k,K,\mathbf{x}'),\ \text{where } (\mathbf{x}',t') \xleftarrow{\$} A_{\mathrm{m}}(1^k) \\
\mathcal{E}_4(1^k,(pk,K),\mathbf{x}) &= \mathcal{E}(1^k,pk,\mathbf{x}')\|\mathsf{F}(1^k,K,\mathbf{x}'),\ \text{where } (\mathbf{x}',t') \xleftarrow{\$} A_{\mathrm{m}}(1^k)\ .
\end{aligned}$$

Above, $\mathbf{0}$ denotes the size-v vector with each component 0^n, and F is extended to vectors componentwise analogously to encryption, as is "$\cdot\|\cdot$" where both arguments are vectors. Consider a game with A and scheme EaH_i for $i \in \{1,2,3,4\}$ that goes as follows:

Game 1
$(pk,K) \xleftarrow{\$} \mathcal{K}'(1^k)$
$(\mathbf{x},t) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$g \xleftarrow{\$} A_{\mathrm{g}}(1^k,K,\mathcal{E}_i(1^k,(pk,K),\mathbf{x}))$
If $g = t$ then return 1
Else return 0

Denote by p_i the probability that this game outputs 1 when run with EaH_i. We have by construction

$$\Pr\left[\,\mathbf{Exp}^{\text{priv-1}}_{\mathsf{EaH}_{\mathsf{HIDE}},A} \Rightarrow 1\,\right] \;=\; p_1\ ; \quad \Pr\left[\,\mathbf{Exp}^{\text{priv-0}}_{\mathsf{EaH}_{\mathsf{HIDE}},A} \Rightarrow 1\,\right] \;=\; p_4\ .$$

So, by definition

$$\mathbf{Adv}^{\text{priv}}_{\mathsf{EaH}_{\mathsf{HIDE}},A} \;=\; p_1 - p_4\ .$$

Now we write

$$p_1 - p_4 \;=\; (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4)\ . \tag{6}$$

To bound the right-hand side above, taking into account the definition of the advantages of B, A', B' we have

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} = p_1 - p_2\ ; \quad \mathbf{Adv}^{\text{priv}}_{\mathsf{HIDE},A'} = p_3 - p_2\ ; \quad \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B'} = p_4 - p_3\ .$$

We substitute these into (6) to get

$$\mathbf{Adv}^{\text{priv}}_{\mathsf{EaH}_{\mathsf{HIDE}},A} \;=\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} + \mathbf{Adv}^{\text{priv}}_{\mathsf{HIDE},A'} + \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B'}\ .$$

Now assuming wlog that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B'}(k) \;\le\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B}(k)$$

and combining-like terms completes the proof. ∎

In the RO model, it is easy to construct a PRIV secure deterministic hiding scheme $\mathsf{HIDE}' = (\mathsf{Kg}', \mathsf{F}')$ by setting Kg' on input 1^k to output a random $K \in \{0,1\}^k$ and F' on input $1^k, K, x$ to return $H(K\|x)$, where H is a RO. But Theorem 6.1 says we can do better than scheme $\mathsf{EaH}_{\mathsf{HIDE}'}$ in this case, in the sense that K can be dropped and $H(K\|x)$ replaced with $H(pk\|x)$. However, the analysis in this case (given in Appendix B) is slightly more involved.

8 Acknowledgments

We would like to thank Brian Cooper and Cynthia Dwork for helpful discussions, and Alex Dent and Ulrich Kühn for feedback on an early draft of this paper. Thanks also to Diana Smetters and Dan Wallach for pointing us to the work of [2, 29]. Finally, we thank the anonymous reviewers of Crypto 2007 for their comments and suggestions.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In V. Shoup, editor, *Crypto '05*, volume 3621 of *LNCS*. Springer, 2005. (Cited on page 3, 5.)

[2] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Symposium on Operating System Design and Implementation (OSDI '02)*. Springer, 2002. (Cited on page 5, 20.)

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD '04*. ACM, 2004. (Cited on page 5.)

[4] G. Amanatidis, A. Boldyreva, and A. O'Neill. New security models and provably-secure schemes for basic query support in outsourced databases. In *Working Conference on Data and Applications Security (DBSec '07)*, LNCS. Springer, 2007. (Cited on page 5.)

[5] J.-H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. Knudsen, editor, *EUROCRYPT '02*, volume 2332 of *LNCS*. Springer, 2002. (Cited on page 17.)

[6] J. Baek, R. Safavi-Naini, and W. Susilo. Public key encryption with keyword search revisited. *Cryptology ePrint Archive, Report 2005/151*, 2005. (Cited on page 5.)

[7] O. Baudron, D. Pointcheval, and J. Stern. Extended notions of security for multicast public key cryptosystems. In *ICALP '00*, volume 1853 of *LNCS*. Springer, 2000. (Cited on page 9.)

[8] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In B. Preneel, editor, *EUROCRYPT '00*, volume 1807 of *LNCS*. Springer, 2000. (Cited on page 7, 8, 9.)

[9] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO '07*, volume 4622 of *LNCS*. Springer, 2007. (Cited on page 5, 15.)

[10] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *FOCS '97*, pages 394–403, 1997. (Cited on page 3.)

[11] M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In *Conference on Computer and Communications Security (CCS '02)*. ACM, 2002. (Cited on page 5.)

[12] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Conference on Computer and Communications Security (CCS '93)*. ACM, 1993. (Cited on page 3, 8.)

[13] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In A. De Santis, editor, *Eurocrypt '94*, volume 950. Springer, 1995. (Cited on page 3, 11.)

[14] M. Bellare and P. Rogaway. The game-playing technique and its application to triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. (Cited on page 7, 17, 24, 25, 35, 36.)

[15] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *Crypto '04*, volume 3027 of *LNCS*. Springer, 2004. (Cited on page 4.)

[16] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT '04*, volume 3027 of *LNCS*. Springer, 2004. (Cited on page 3, 5.)

[17] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data, 2007. (Cited on page 5.)

[18] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO '06*, volume 4117 of *LNCS*. Springer, 2006. (Cited on page 3, 4, 5.)

[19] V. Boyko. On the security properties of OAEP as an all-or-nothing transform. In M. J. Wiener, editor, *CRYPTO '99*, volume 1666 of *LNCS*. Springer, 1999. (Cited on page 13.)

[20] R. Brinkman, L. Feng, J. M. Doumen, P. H. Hartel, and W. Jonker. Efficient tree search in encrypted data. Technical Report TR-CTIT-04-15, Enschede, March 2004. (Cited on page 5.)

[21] R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In B. Kaliski, editor, *CRYPTO '97*, volume 1294 of *LNCS*. Springer, 1997. (Cited on page 5, 16.)

[22] R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. In *STOC '98*. ACM, 1998. (Cited on page 5, 16.)

[23] 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 '05*, volume 3531 of *LNCS*, 2005. (Cited on page 5.)

[24] R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *Conference on Computer and Communications Security (CCS '06)*. ACM, 2006. (Cited on page 5.)

[25] E. Damiani, S. De Capitani Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In S.Jajodia, V. Atluri, and T. Jaeger, editors, *Conference on Computer and Communications Security (CCS '03)*. ACM, 2003. (Cited on page 5.)

[26] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):1130–1140, 1976. (Cited on page 3.)

[27] Y. Dodis and A. Smith. Entropic security and the encryption of high entropy messages. In J. Kilian, editor, *TCC '05*, volume 3378 of *LNCS*. Springer, 2005. (Cited on page 5.)

[28] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2), 2000. (Cited on page 3.)

[29] J.R. Douceur, A. Adya, W.J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Conference on Distributed Computing Systems (ICDCS'02)*, 2002. (Cited on page 5, 20.)

[30] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, Vol. 31, 1985. (Cited on page 11.)

[31] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In J. Kilian, editor, *CRYPTO '01*, volume 2139 of *LNCS*. Springer, 2001. (Cited on page 3, 11, 13, 23.)

[32] 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. (Cited on page 17.)

[33] Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS '01*. IEEE, 2001. (Cited on page 4.)

[34] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`. (Cited on page 5.)

[35] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2), 1984. (Cited on page 3, 7.)

[36] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security Conference (ACNS '04)*, volume 3089 of *LNCS*. Springer, 2004. (Cited on page 5.)

[37] H. Hacigümüs, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Conference on Management of data (SIGMOD '02)*. ACM, 2002. (Cited on page 5.)

[38] 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 '04*, volume 2973 of *LNCS*. Springer, 2004. (Cited on page 5.)

[39] 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 '04*. Morgan Kaufmann, 2004. (Cited on page 5.)

[40] 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 '04*, volume 2992 of *LNCS*. Springer, 2004. (Cited on page 5.)

[41] M. Kantracioglu and C. Clifton. Security issues in querying encrypted data. In *Working Conference on Data and Applications Security (DBSec '05)*, LNCS. Springer, 2005. (Cited on page 5.)

[42] J. Li and E. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In *Working Conference on Data and Applications Security (DBSec '05)*, LNCS. Springer, 2005. (Cited on page 5.)

[43] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2), 1988. (Cited on page 5.)

[44] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, 1988. (Cited on page 3.)

[45] G. Özsoyoglu, D. A. Singer, and S. S. Chung. Anti-tamper databases: Querying encrypted databases. In *Working Conference on Data and Applications Security (DBSec '03)*, LNCS. Springer, 2003. (Cited on page 5.)

[46] D. Pointcheval. How to encrypt properly with RSA. *RSA Laboratories' CryptoBytes*, 5(1), Winter/Spring 2002. (Cited on page 13.)

[47] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO '91*, volume 576 of *LNCS*. Springer, 1992. (Cited on page 3.)

[48] 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. (Cited on page 11.)

[49] A. Russell and H. Wang. How to fool an unbounded adversary with a short key. *IEEE Transactions on Information Theory*, 52(3):1130–1140, 2006. (Cited on page 5.)

[50] V. Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, New York, NY, USA, 2005. (Cited on page 26.)

[51] Arsenal Digital Solutions. Top 10 reasons to outsource remote data protection. `http://www.arsenaldigital.com/services/remote_data_protection.htm`. (Cited on page 3.)

[52] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Symposium on Security and Privacy*. IEEE, 2000. (Cited on page 5.)

[53] H. Wang and L.V.S. Lakshmanan. Efficient secure query evaluation over encrypted XML databases. In *VLDB '06*. VLDB Endowment, 2006. (Cited on page 5.)

[54] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*. Springer, 2005. (Cited on page 4.)

A Proof of Theorem 5.2

Since RSA-DOAEP is deterministic, we assume wlog that the plaintext components of a vector $\mathbf{x}$ are distinct for all $(\mathbf{x}, t)$ output by $A_{\mathrm{m}}(1^k)$. We prove Case 1, meaning we suppose that $n - k_0 < k_1$. (This is used in the lemma below.) Case 1 is the "harder" case in the sense that we need to use Lemma 6 from [31]. To state this, we call a *partial one-way* adversary I' an algorithm that takes input $(N, e), x^e \bmod \phi(N)$ and tries to compute just the last $k_1 - (n - k_0)$ bits of x. (The fact that it computes the *last* $n - k_0$ bits of x is just for convenience; [31] actually proves something general.) Let $k_2 = k_1 - (n - k_0)$. For a partial one-way adversary I', define

$$\mathbf{Adv}^{\mathrm{powf}}_{\mathcal{F},I'} = \Pr\left[\, ((N,e),(N,d)) \xleftarrow{\$} \mathcal{F}(1^k)\ ;\ x \xleftarrow{\$} \{0,1\}^{k_1}\ ;\ z \xleftarrow{\$} I'((N,e), x^e)\ :\ z = x[k_2+1 \ldots k_1] \,\right] .$$

Intuitively, the lemma says that RSA is one-way if it is partial one-way (and $n - k_0$ is large enough relative to k_1).

Lemma A.1 [31, Lemma 6] Let $\mathcal{F}$ be an RSA trapdoor permutation generator with modulus length k_1 and let I' be a partial one-way adversary against $\mathcal{F}$. Then there exists an inverter I against $\mathcal{F}$ such that

$$\mathbf{Adv}^{\mathrm{powf}}_{\mathcal{F},I'} \ \leq\ \sqrt{\mathbf{Adv}^{\mathrm{owf}}_{\mathcal{F},I} + 2^{2k_1 - 4(n-k_0)+10}} \ +\ 2^{k_1 - 2(n-k_0)+5} .$$

Furthermore, the running-time of I is at most twice that of I' plus $O(k_1^3)$. ∎

For the proof, we construct a partial one-way adversary against $\mathcal{F}$ and then conclude the existence of an inverter I by the lemma. The former, which we call GetQuery, is depicted in Figure 5. The main games for the proof begin in Figure 2 and are continued in Figure 3.

Equation (3) of Theorem 5.2 follows from the following sequence of inequalities, which we will justify below:

$$
\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{priv-cpa}}_{\mathsf{DOAEP},A} &= \Pr\left[\, G_1^{A_{\mathrm{g}}} \Rightarrow b \,\right] && (7)\\
&\le \Pr\left[\, G_2^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \Pr[G_1^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_0\,] && (8)\\
&\le \Pr\left[\, G_2^{A_{\mathrm{g}}} \Rightarrow b \,\right] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (9)\\
&\le \Pr\left[\, G_3^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \Pr[G_2^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_1\,] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (10)\\
&\le \Pr\left[\, G_3^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (11)\\
&\le \Pr\left[\, G_4^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_2\,] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (12)\\
&\le \Pr\left[\, G_4^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (13)\\
&\le \Pr\left[\, G_5^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + \Pr[G_4^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] \\
&\quad + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (14)\\
&\le \Pr\left[\, G_6^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + \Pr[G_4^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] \\
&\quad + \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_4\,] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (15)\\
&= \Pr\left[\, G_7^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + \Pr[G_4^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] \\
&\quad + \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_4\,] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (16)\\
&\le \Pr\left[\, G_8^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + \Pr[G_4^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] \\
&\quad + \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_4\,] + \Pr[G_7^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_5\,] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (17)\\
&\le \frac{1}{2} + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + \Pr[G_4^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_3\,] \\
&\quad + \Pr[G_5^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_4\,] + \Pr[G_7^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_5\,] + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (18)\\
&\le \frac{1}{2} + \mathbf{Adv}^{\text{powf}}_{\mathcal{F},\mathsf{GetQuery}} + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}} && (19)\\
&\le \frac{1}{2} + q_{\mathrm{h}_2} v \cdot \sqrt{\mathbf{Adv}^{\text{owf}}_{\mathcal{F},I} + 2^{2k_1 - 4(n-k_0)+10}} + 2^{k_1 - 2(n-k_0)+5} \\
&\quad + \frac{q_{\mathrm{r}} v}{2^{k_0}} + \frac{q_{\mathrm{h}_1} q_{\mathrm{r}} v}{2^{\mu}} + 2q_{\mathrm{tot}} \cdot \mathsf{mpk}_{\mathcal{F}}\,. && (20)
\end{aligned}
$$

As we have seen, the advantage of A is 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 $b = 0$. Equation (7) follows from the fact that Game G_1 is just this game written in a convenient way. (The reasoning here is similar to in the proof of Theorem 5.1. We clarify that the two runs of A_{m} in the game have different coins but the same arrays H_1, R, H_2 are used for the ROs, so they have the same ROs.)

Games G_1, G_2 differ only in statements that follow the setting of bad_0, meaning are, in the terminology of [14], identical-until-bad_0 games. The Fundamental Lemma of Game Playing [14] thus applies to justify (8). Let us denote by $\mathsf{mpk}_{\mathcal{F}}$ the max public-key probability of RSA-DOAEP (which only depends on the underlying RSA trapdoor-permutation generator $\mathcal{F}$ and not on the rest of the

scheme). The probability $H_1[(N,e)\|x_{i,r}], R[(N,e)\|s_{i,0}]$ or $H_2[(N,e)\|t_{i,0}]$ for some $i \in \{1,\ldots,v\}$ is defined, meaning the array index was queried by $A_{\rm m}$ to its RO, is at most $2q \cdot \mathsf{mpk}_{\mathcal{F}}$ because $A_{\rm m}$, which makes a total of at most $q_{\rm tot}$ queries to all its ROs combined, gets no information about public key (N,e). (The factor of 2 is due to the fact that $A_{\rm m}$ is run twice.) This justifies (9).

As before, the Fundamental Lemma applies to justify (10). To bound the probability that Game G_2 when executing $A_{\rm g}$ sets bad_1, note that without $A_{\rm g}$ querying $(N,e)\|x_{i,r}$ (by which we mean for some $i \in \{1,\ldots v\}$) to oracle H_1 nor $(N,e)\|t_{i,0}$ to H_2 (i.e. up to the point that bad_1 is set), the values of $H^*_{i,1}, H^*_{i,2}, s_{i,1}, t_{i,0}$ are all random and independently distributed from $\mathbf{x}_b$ from its perspective. To be more precise, we define an auxilliary game called $G_{\rm rand}$, shown in Figure 4, in which all input to $A_{\rm g}$ and answers to its oracle queries are independent and random of $\mathbf{x}_b$ and the appropriate independent and dependent variables defined by the game are swapped. We claim that the probability G_2 when executing $A_{\rm g}$ sets bad_1 is at most the probability that $G_{\rm rand}$ does. To see this, we can wlog consider a common finite space of coins associated to the executions games G_2 and $G_{\rm rand}$ with $A_{\rm g}$ where if G_2 when executed using some particular sequence of coins from this space sets bad_1 then $G_{\rm rand}$ when executed using this same coin sequence also sets bad_1, because the executions of the games on these coin sequences are identical. So the probability of setting bad_1 in $G_{\rm rand}$ can only go up as compared to G_2. Now, since when executed with $G_{\rm rand}$, $A_{\rm g}$ gets no information about $s^*_{i,0}$ for any $1 \le i \le v$, the probability that $G_{\rm rand}$ when executing $A_{\rm g}$ sets bad_1 is at most $q_{\rm r} v/2^{k_0}$, giving (11).

Equation (12) is again obtained via the Fundamental Lemma in [14]. We bound the probability that G_3 when executing $A_{\rm g}$ sets bad_2 by showing that the latter implies $A_{\rm g}$ has "guessed" $\mathbf{x}_b[i]$ without being given any information about it, as follows. This probability is at most the probability that $G_{\rm rand}$ does, by an analogous argument to the above. But the probability of that $G_{\rm rand}$ sets bad_2 is, in turn, the same, over a common finite set of coins with which $A_{\rm g}$ is executed, as the probability that the "knowledge extractor" K shown in Figure 6 outputs a list containing the plaintext $\mathbf{x}_b[i]$ for some $1 \le i \le v$, where $\mathbf{x}_0, \mathbf{x}_1, b$ are defined as in $G_{\rm rand}$. The probability that K outputs such a list is at most $q_{\rm h_1} q_{\rm r} v/2^{\mu}$, because it gets no information about $\mathbf{x}_b$. So we have justified (13). (We remark that this was the step in the proof where we use the fact that the underlying padding transform of RSA-DOAEP consists of *three* Feistel rounds and not two.)

As usual, the Fundamental Lemma in [14] applies to justify all of (14), (15), and (17). We delay bounding the probabilities here until later.

Next consider when $A_{\rm g}$ executed with Game G_6 queries $(N,e)\|x_{i,r}$ to H_1 but prior to this has queried neither $(N,e)\|s_{i,0}$ to R nor $(N,e)\|t_{i,0}$ to H_2. Then, in reply to query $(N,e)\|x_{i,r}$, it receives $H^*_{i,1}$, which is random and independent from everything given to $A_{\rm g}$ so far. Then, after it queries $(N,e)\|x_{i,r}$ to H_1, we see from the code that the answers given to $A_{\rm g}$ in reply to any of its queries are likewise random and independent. This means that, instead replying to query $(N,e)\|x_{i,r}$ with the special string $H^*_{i,1}$ defined at the beginning of the game, we could simply reply with a random and independent string chosen "on the fly" during the particular invocation of the procedure to respond to H_1 queries. In other words, we may drop the "Else" statement in this procedure to result in an equivalent game G_6, which justifies (16).

Now, we have that the probability that G_8 outputs the challenge bit b chosen randomly at the beginning of the game when executing $A_{\rm g}$ is at most 1/2, because this game does not give $A_{\rm g}$ any information about $\mathbf{x}_b$, giving (18).

Finally, observe that in each of the following cases, the probability that $A_{\rm g}$ causes the relevant game to set the flag also causes $G_{\rm rand}$ to do so with at least the same probability: game G_4 sets bad_3, game G_5 sets bad_4, and game G_7 sets bad_5. The argument here is analogous to the justification

of (11). Moreover, these cases exhaust all the possible sequences of queries made by $A_{\rm g}$ for which $A_{\rm g}$ queries $(N,e)\|t_{i,0}$ for some i to its H_2 oracle. Now the input to $A_{\rm g}$ and its oracle replies are distributed identically when executed with $G_{\rm rand}$ and when run by algorithm $\mathsf{GetQuery}$, except that the procedure to respond to queries to H_2 in $G_{\rm rand}$ explicitly checks whether a query made by $A_{\rm g}$ is equal to $t_{i,0}$ for some i, whereas algorithm $\mathsf{GetQuery}$ simply guesses whether this is the case by picking j when the first such query (which it hopes is $t_{w,0}$) will occur, at which point its output is determined. (Game $G_{\rm rand}$ also defines some additional variables, makes some "If" checks, and sets some flags omitted by $\mathsf{GetQuery}$, but none of these influence game output.) Since w, j are random and independent and $A_{\rm g}$ gets no information about them, we have

$$\Pr[G_3^{A_{\rm g}} \text{ sets } \mathsf{bad}_2\,] + \Pr[G_5^{A_{\rm g}} \text{ sets } \mathsf{bad}_3\,] + \Pr[G_7^{A_{\rm g}} \text{ sets } \mathsf{bad}_4\,] \;\le\; q_{\rm h_2} v \cdot \mathbf{Adv}^{\rm powf}_{\mathcal{F},\mathsf{GetQuery}} \;,$$

justifying (19). Equation (20) then follows by Lemma A.1.

Finally, we proceed to bound the max public-key probability of RSA-DOAEP as follows. We first recall the following fact, which can be derived from a proof of Chebyshev's theorem about the density of primes (see [50, Theorem 5.3]).

Fact A.2 Let $\pi(n)$ be the number of prime numbers less than or equal to an integer n. Then for all n

$$\pi(n) \;\ge\; \frac{2n}{\ln(n)} \;. \;\blacksquare$$

Claim A.3 Let $\mathsf{mpk}_{\mathcal{F}}$ denote the max public-key probability of scheme RSA-DOAEP with modulus length $k_1 = k$. Then

$$\mathsf{mpk}_{\mathcal{F}} \;\le\; \frac{\ln(2^{k_1/2}-1)}{2^{k_1/2}-1} \;. \;\blacksquare$$

Proof: Fix some $k_1/2$-bit prime numbers p', q'. We have that

$$\begin{aligned}
\mathsf{mpk}_{\mathcal{F}} &\le \Pr\left[\, p'\cdot q' = p\cdot q \;:\; ((p\cdot q,e),(p\cdot q,d)) \xleftarrow{\$} \mathcal{F}(1^k)\,\right] \\
&= \Pr\left[\, (p'=p \;\wedge\; q'=q) \;\vee\; (p'=q \;\wedge\; q'=p) \;:\; ((p\cdot q,e),(p\cdot q,d)) \xleftarrow{\$} \mathcal{F}(1^k)\,\right] \\
&\le \Pr\left[\, p'=p \;\vee\; p'=q \;:\; ((p\cdot q,e),(p\cdot q,d)) \xleftarrow{\$} \mathcal{F}(1^k)\,\right] \\
&\le 2\cdot\frac{\ln(2^{k_1/2}-1)}{2\cdot(2^{k_1/2}-1)} \;,
\end{aligned}$$

as desired. Above, the second line is by unique factorization and the last uses Fact A.2. $\blacksquare$

Applying Claim A.3 to (20) and re-arranging yields (3). To finish the proof, we justify the running-time analysis of I by taking into account the convention that the running-time of A includes that of its overlying experiment. Additional time-complexity for $\mathsf{GetQuery}$ here is for picking two random numbers between 1 and $q_{\rm h_2}, v$, respectively, and maintaining a counter up to value at most $q_{\rm h_2}$, incremented each time $A_{\rm g}$ makes a query to oracle H_2, which is $O(\log v + q_{\rm h_2}\log q_{\rm h_2})$. Then applying the running-time analysis in Lemma A.1, we have that the running-time of I twice that of $\mathsf{GetQuery}$ plus $O(k_1^3)$, as desired. $\blacksquare$

B Proof of Theorem 6.1

We actually define two adversaries, namely B, B', for the proof, as shown below. At the conclusion of the proof, we comment on how to extend it to the CCA case.

Adversary $B^{\mathcal{E}(1^k,pk,\mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$	**Adversary** $B'^{\mathcal{E}(1^k,pk,\mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$
Run A_{m} on input 1^k:	Run A_{m} twice on input 1^k:
On query Hash$(s\|x)$:	**On query Hash**$(s\|x)$:
If $H[s\|x]$ is undefined then	If $H[s\|x]$ is undefined then
$\quad H[s\|x] \xleftarrow{\$} \{0,1\}^l$	$\quad H[s\|x] \xleftarrow{\$} \{0,1\}^l$
Return $H[s\|x]$	Return $H[s\|x]$
Let $(\mathbf{x}_1, t_1)$ be the output of A_{m}	Let $(\mathbf{x}_0, t_0), (\mathbf{x}_1, t_1)$ be the outputs of A_{m}
$\mathbf{c} \xleftarrow{\$} \mathcal{E}(pk, \mathrm{LR}(\mathbf{x}_1, \mathbf{0}, b))$	$\mathbf{c} \xleftarrow{\$} \mathcal{E}(pk, \mathrm{LR}(\mathbf{x}_0, \mathbf{0}, b))$
For $i = 1$ to v do:	For $i = 1$ to v do:
$\quad$ If $H[pk\|\mathbf{x}[i]]$ is undefined then	$\quad$ If $H[pk\|\mathbf{x}_0[i]]$ is undefined then
$\quad\quad H[pk\|\mathbf{x}[i]] \xleftarrow{\$} \{0,1\}^l$	$\quad\quad H[pk\|\mathbf{x}_0[i]] \xleftarrow{\$} \{0,1\}^l$
$\quad \mathbf{c}[i] \leftarrow \mathbf{c}[i]\|H[pk\|\mathbf{x}_1[i]]$	$\quad \mathbf{c}[i] \leftarrow \mathbf{c}[i]\|H[pk\|\mathbf{x}_0[i]]$
Run A_{g} on input $1^k, pk, \mathbf{c}$:	Run A_{g} on input $1^k, pk, \mathbf{c}$:
On query Hash$(s\|x)$:	**On query Hash**$(s\|x)$:
If $H[s\|x]$ is undefined then	If $H[s\|x]$ is undefined then
$\quad H[s\|x] \xleftarrow{\$} \{0,1\}^l$	$\quad H[s\|x] \xleftarrow{\$} \{0,1\}^l$
Return $H[s\|x]$	Return $H[s\|x]$
Let g be the output of A_{g}	Let g be the output of A_{g}
If $g = t_1$ then return 1	If $g = t_0$ then return 1
Else return 0	Else return 0

Above, $\mathbf{0}$ denotes the size-v vector with each component 0^n. We clarify that the two runs of A_{m}, in each of B, B', have different coins but the same array H is used for the RO, so they have the same RO. The proof is a hybrid argument. Consider modified schemes $\mathsf{EaH}_i = (\mathcal{K}', \mathcal{E}_i)$ for $i \in \{1,2,3,4\}$, defined as follows. Each EaH_i has the same key-generation algorithm, namely $\mathcal{K}'$, which on input 1^k outputs pk, where (pk, sk) is the output of $\mathcal{K}(1^k)$ with random coins. The encryption algorithms operate on vectors of size v, and are defined as follows:

$$
\begin{aligned}
\mathcal{E}_1(1^k, pk, \mathbf{x}) &= \mathcal{E}(1^k, pk, \mathbf{x}) \| H(pk\|\mathbf{x}) \\
\mathcal{E}_2(1^k, pk, \mathbf{x}) &= \mathcal{E}(1^k, pk, \mathbf{0}) \| H(pk\|\mathbf{x}) \\
\mathcal{E}_3(1^k, pk, \mathbf{x}) &= \mathcal{E}(1^k, pk, \mathbf{0}) \| H(pk\|\mathbf{x}'), \text{ where } (\mathbf{x}', t') \xleftarrow{\$} A_{\mathrm{m}}(1^k) \\
\mathcal{E}_4(1^k, pk, \mathbf{x}) &= \mathcal{E}(1^k, pk, \mathbf{x}') \| H(pk\|\mathbf{x}'), \text{ where } (\mathbf{x}', t') \xleftarrow{\$} A_{\mathrm{m}}(1^k) \ .
\end{aligned}
$$

Above, $\mathbf{0}$ denotes the size-v vector with each component 0^n, and hash H is extended to vectors component-wise analogously to encryption, as is the in-fix "$\|$" operation where both arguments are vectors. Consider executing A against EaH_i for each $i \in \{1,2,3,4\}$ in the following game:

Game 1
$pk \xleftarrow{\$} \mathcal{K}'(1^k)$
$(\mathbf{x}, t) \xleftarrow{\$} A_{\mathrm{m}}(1^k)$
$g \xleftarrow{\$} A_{\mathrm{g}}(1^k, pk, \mathcal{E}_i(1^k, pk, \mathbf{x}))$
If $g = t$ then return 1
Else return 0

For the analysis, let us call a RO query *public-key prefixed* if it is of the form $pk\|y$ for some y. Let "BAD_1" denote the event that A_{m} makes a public-key prefixed query to its RO. Let "BAD_2" denote the event that A_{g} makes a RO query of the form $pk\|\mathbf{x}[i]$ for some and $i \in \{1, \ldots, v\}$. Lastly, let "E" denote the event that Game 1 outputs 1, and let $\Pr_i[\,\cdot\,]$ be the probability of the argument when Game 1 is executed with A, EaH_i. We first claim that

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH},A} \quad = \quad \Pr_1[\,\mathsf{E}\,] - \Pr_4[\,\mathsf{E}\,] \ .$$

This is clear by construction of $\mathsf{EaH}_1, \mathsf{EaH}_4$, taking into account the definition of the advantage of A. Now expand the above as

$$\Pr_1[\,\mathsf{E}\,] - \Pr_4[\,\mathsf{E}\,] \quad = \quad (\Pr_1[\,\mathsf{E}\,] - \Pr_2[\,\mathsf{E}\,]) + (\Pr_2[\,\mathsf{E}\,] - \Pr_3[\,\mathsf{E}\,]) + (\Pr_3[\,\mathsf{E}\,] - \Pr_4[\,\mathsf{E}\,]) \ . \quad (21)$$

Towards bounding the right-hand side, we next claim that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \quad \geq \quad \Pr_1[\,\mathsf{E}\,] - \Pr_2[\,\mathsf{E}\,] \ .$$

This follows from comparing runs of B and of Game 1 with $\mathsf{EaH}_1, \mathsf{EaH}_2$, taking into account the definition of the advantage of B. Next we claim that

$$\Pr_2[\,\mathsf{E}\,] - \Pr_3[\,\mathsf{E}\,] \quad \leq \quad \Pr_2[\,\mathsf{BAD}_1\,] \ + \ \Pr_2[\,\mathsf{BAD}_2\,] \ .$$

We will justify this below. Assuming it for the moment, let us next argue that

$$\Pr_2[\,\mathsf{BAD}_1\,] \ \leq \ q_{\mathrm{h}} \cdot \mathsf{mpk}_{\mathsf{AE}} \ ; \quad \Pr_2[\,\mathsf{BAD}_2\,] \ \leq \ q_{\mathrm{h}} v \cdot 2^{-\mu} \ .$$

To see the first, note that A_{m}, which makes at most q_{h} RO queries, when executed in Game 1 against EaH_2 (or in fact any of the schemes) does not get any information about pk. To see the second, observe that when A_{g} is executed against EaH_2, it is not given any information about $\mathbf{x}$ until BAD_2 occurs. Similarly to before, we also have that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B'} \quad \geq \quad \Pr_3[\,\mathsf{E}\,] - \Pr_4[\,\mathsf{E}\,] \ .$$

Now we substitute all these into (21) to get

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH},A} \quad \leq \quad \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B} \ + \ q_{\mathrm{h}} \cdot \mathsf{mpk}_{\mathsf{AE}} \ + \ q_{\mathrm{h}} v \cdot 2^{-\mu} \ + \ \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B'} \ .$$

Assuming wlog that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B'} \quad \leq \quad \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B}$$

and combining like-terms then yields (4). So it remains to prove the following.

Claim B.1 The claim is that

$$\Pr_2[\,\mathsf{E}\,] \quad \leq \quad \Pr_3[\,\mathsf{E}\,] + \Pr_2[\,\mathsf{BAD}_1\,] + \Pr_2[\,\mathsf{BAD}_2\,] \ .$$

Proof: The claim follows from the following sequence of inequalities:

$$\begin{aligned}
\Pr_2[\,\mathsf{E}\,] &= \Pr_2\left[\,\mathsf{E} \wedge \overline{\mathsf{BAD}}_1 \wedge \overline{\mathsf{BAD}}_2\,\right] + \Pr_2[\,\mathsf{E} \wedge (\mathsf{BAD}_1 \vee \mathsf{BAD}_2)\,] \\
&\leq \Pr_2\left[\,\mathsf{E} \mid \overline{\mathsf{BAD}}_1 \wedge \overline{\mathsf{BAD}}_2\,\right] \cdot \Pr_2\left[\,\overline{\mathsf{BAD}}_1 \wedge \overline{\mathsf{BAD}}_2\,\right] + \Pr_2[\,\mathsf{BAD}_1 \vee \mathsf{BAD}_2\,] \\
&= \Pr_3\left[\,\mathsf{E} \mid \overline{\mathsf{BAD}}_1 \wedge \overline{\mathsf{BAD}}_2\,\right] \cdot \Pr_3\left[\,\overline{\mathsf{BAD}}_1 \wedge \overline{\mathsf{BAD}}_2\,\right] + \Pr_2[\,\mathsf{BAD}_1 \vee \mathsf{BAD}_2\,] \\
&= \Pr_3\left[\,\mathsf{E} \wedge \overline{\mathsf{BAD}}_1 \wedge \overline{\mathsf{BAD}}_2\,\right] + \Pr_2[\,\mathsf{BAD}_1 \vee \mathsf{BAD}_2\,] \\
&\leq \Pr_3[\,\mathsf{E}\,] + \Pr_2[\,\mathsf{BAD}_1\,] + \Pr_2[\,\mathsf{BAD}_2\,] \ .
\end{aligned}$$

To see the third line above, consider runs of Game 1 with $\mathsf{EaH}_2, \mathsf{EaH}_3$ and A over a common finite set of coins; note that a sequence of coins (which includes the coins for A) that cause Game 1 with EaH_2 and A to output 1 and for which both $\mathsf{BAD}_1, \mathsf{BAD}_2$ are not true and just those that cause Game 1 with EaH_3 and A to output 1 and for which both $\mathsf{BAD}_1, \mathsf{BAD}_2$ are not true, because if $\mathsf{BAD}_1, \mathsf{BAD}_2$ are not true then the executions of each are identical. ∎

Extending the proof to CCA. To extend the above proof to show PRIV-CCA security of EaH assuming IND-CCA security of the starting encryption scheme AE, adversaries B, B' would additionally have access to decryption oracle $\mathcal{D}(1^k, pk, sk, \cdot)$ and would simulate answers to queries of $A_{\rm g}$ to oracle $\mathcal{HD}(1^k, pk, sk, \cdot)$ as follows: On query $c\|h$, check if $c\|h'$ for any string h' occurs in input $\mathbf{c}$ to $A_{\rm g}$; if so, return $\perp$. Otherwise, compute $\mathcal{HD}(1^k, pk, sk, c\|h)$ by using oracle $\mathcal{D}(1^k, pk, sk, \cdot)$ to decrypt c and return the result to $A_{\rm g}$. This "works" because hash function H is deterministic. So, if $A_{\rm g}$ submits decryption query $c\|h$ such that some $c\|h'$ occurs in its input $\mathbf{c}$, then either $h' = h$ (which is not a query $A_{\rm g}$ is allowed to make), or else h' is not the hash of the decryption of c, and therefore $\mathcal{D}(1^k, pk, sk, c\|h)$ outputs $\perp$. The rest of the proof remains essentially identical. ∎

C Proof of Theorem 6.2

For simplicity, we analyze a modified scheme $\mathsf{EaH}' = (\mathcal{K}, \mathcal{HE}')$ where the ciphertext is dropped from the output, namely we just have $\mathcal{HE}'^{H}(pk, x) = H(pk\|x)$. (This is a "hiding scheme" as defined in Section 7.) Our analysis then implies Theorem 6.2 by a simple hybrid argument. (Moreover, essentially the same hybrid argument can be used to treat PRIV-CCA security of EaH assuming IND-CCA security of AE.) As a warm-up, we start with the case that $A_{\rm m}$ puts a uniform distribution on a finite set of plaintexts S.

The uniform case. Say that MR adversary A is *uniform* if $A_{\rm m}$ puts a uniform distribution on some finite set of plaintexts S. For a uniform MR adversary A we claim that

$$\mathbf{Adv}^{\rm priv}_{\mathsf{EaH}',A} \quad \leq \quad 6 \cdot \frac{2^l}{|S| - 1} \ . \tag{22}$$

Note that this is quite a good bound because we expect the advantage of A in this case to be about the inverse of the size of the "bucket" in which the challenge message chosen by $A_{\rm m}$ lies (because $A_{\rm g}$ can do no better than simply guessing at random a message from this bucket), and the expected bucket size is $|S|/2^l$. So the bound says that we are basically off by at most a small constant factor. To deduce it, we consider a simplified game that works as follows:

Game 1
$h_0, h_1, \ldots, h_M \xleftarrow{\$} \{0,1\}^l$
$m \xleftarrow{\$} \{m_0, \ldots, m_M\}$
$C \leftarrow \{i \ \geq \ 1 \ : \ h_i = h_0\}$
Return $|C|$

Regarding h_0 as the hash of the challenge message m, the game is, intuitively, returning how many other plaintexts besides the challenge message have hash value equal to that of the challenge message. To bound A's advantage, we would like to know how large we expect the game's output to be. For this, we define some random variables. Let $S = \{m_0, m_1, \ldots, m_M\}$. For $i = 1$ to M define the random variable

$$X_i(h_0, \ldots, h_M) = \begin{cases} 1 & \text{if } h_i = h_0, \\ 0 & \text{O.W.}, \end{cases}$$

and define $X = \sum_{i=1}^{M} X_i$. So Game 1 returns the value of X. Note that the X_i are pairwise independent. Some simple computations that we will need are (where free variables have implicit universal quantification):

$$\mathbf{E}[\,X_i\,] \ = \ 2^{-l}; \quad \mathbf{E}[\,X\,] \ = \ \sum_{i=1}^{m} \mathbf{E}[\,X_i\,] \ = \ M \cdot 2^{-l}\,,$$

where the second is from linearity of expectation. Now, for some $0 \leq \alpha \leq 1$ to be determined later, we claim the following inequalities, which we will justify below:

$$\begin{aligned}
\mathbf{Adv}^{\text{priv}}_{\mathsf{EaH}',A} &= \Pr\left[\,\mathbf{Exp}^{\text{priv-1}}_{\mathsf{EaH}',A} \Rightarrow 1\,\right] - \Pr\left[\,\mathbf{Exp}^{\text{priv-0}}_{\mathsf{EaH}',A} \Rightarrow 1\,\right] \\
&\leq \Pr\left[\,\mathbf{Exp}^{\text{priv-1}}_{\mathsf{EaH}',A} \Rightarrow 1\,\right] \\
&\leq \mathbf{E}\left[\,\frac{1}{1+X}\,\right] \\
&\leq \Pr[\,X \ \leq \ \alpha \cdot \mathbf{E}[\,X\,]\,] \cdot \frac{1}{1} \ + \ \Pr[\,X \ > \ \alpha \cdot \mathbf{E}[\,X\,]\,] \cdot \frac{1}{\alpha \cdot \mathbf{E}[\,X\,]} \\
&\leq \Pr[\,X \ \leq \ \alpha \cdot \mathbf{E}[\,X\,]\,] \ + \ \frac{1}{\alpha \cdot \mathbf{E}[\,X\,]} \\
&\leq \Pr[\,X \ \leq \ \alpha \cdot \mathbf{E}[\,X\,]\,] \ + \ \frac{2^l}{\alpha \cdot M}\,.
\end{aligned}$$

Of these, the first is by definition and the second is because dropping the second term on the right-hand side can only cause the latter to go up.

For the third, we can consider runs of Game 1 and the priv-1 experiment, over a common finite set of coins, respectively. In a run of the experiment and A, the best strategy for A_{g} to recover challenge message x is to output at random one of the messages from S whose hash value (when concatenated with pk) is equal to its input $H(pk\|x)$. Its success probability is therefore at most $1/n$, where n is the number of such messages. Now look at a run of Game 1 on the same coins. Viewing h_0 as the hash of the challenge message for A_{g}, it returns exactly $n-1$. As X takes this return value, the probability (over the random coins of the experiment and A) that A recovers the challenge message when executed in the priv-1 experiment is at most $\mathbf{E}[\,1/(1+X)\,]$.

The fourth line is a conditioning argument, where we use that if $X \leq \alpha \cdot \mathbf{E}[\,X\,]$ then $1/(1+X)$ is at most $1/(1+0)$ and if $X > \alpha \cdot \mathbf{E}[\,X\,]$ then $1/(1+X)$ is at most $1/(1+\alpha \cdot \mathbf{E}[\,X\,]) \leq 1/\alpha \cdot \mathbf{E}[\,X\,]$.

The fifth is dropping a positive factor less than 1 and the sixth is substitution according to our previous calculation of $\mathbf{E}[X]$.

So it remains to bound the probability on the last line. For this, we claim the following.

Claim C.1 For any $0 \leq \alpha \leq 1$, we have

$$\Pr[X \leq \alpha \cdot \mathbf{E}[X]] \leq \frac{1}{(1-\alpha)^2} \cdot \frac{2^l}{M} .$$

Proof: The claim follows from the following inequalities, which we return to justify below:

$$\begin{aligned}
\Pr[X \leq \alpha \cdot \mathbf{E}[X]] &\leq \Pr[|X - \mathbf{E}[X]| \geq (1-\alpha) \cdot \mathbf{E}[X]] \\
&\leq \frac{\mathbf{Var}[X]}{(1-\alpha)^2 \cdot \mathbf{E}[X]^2} \\
&\leq \frac{\mathbf{E}[X]}{(1-\alpha)^2 \cdot \mathbf{E}[X]^2} \\
&= \frac{1}{(1-\alpha)^2} \cdot \frac{2^l}{M} .
\end{aligned}$$

The first and last are just algebraic manipulation and substitution according to our previous calculations. The second is by Chebyshev's inequality. For the third, we use:

$$\begin{aligned}
\mathbf{Var}[X] &= \sum_{i=1}^{M} \mathbf{Var}[X_i] \\
&\leq \sum_{i=1}^{M} \mathbf{E}[X_i] \\
&= \mathbf{E}[X] .
\end{aligned}$$

Of these, the first is by pairwise independence, and the second follows from the fact that for any binary-valued random variable Y, $\mathbf{Var}[Y] \leq \mathbf{E}[Y]$. The last we have already seen. ∎

Now, substituting according to the claim gives

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH}',A} \leq \frac{1}{(1-\alpha)^2} \cdot \frac{2^l}{M} + \frac{2^l}{\alpha \cdot M} .$$

Setting α to $1/2$ and replacing M with $|S| - 1$ yields (22). This completes our analysis of the uniform case. ∎

The non-uniform case. Now consider the more general case of a MR privacy adversary A where A_{m} does not necessarily put the uniform distribution, but instead some arbitrary distribution $\mathcal{D}$, on set S of plaintexts. Let μ be the min-entropy, and set $p = 2^{-\mu}$. This time, let $|S| = M$. For $i \in \{1, \ldots, M\}$, let p_i be the probability given to m_i according to $\mathfrak{D}$. In this case, we claim that

$$\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH}',A} \leq 12 \cdot \mathsf{Coll}[\mathfrak{D}] \cdot |S| \cdot \frac{2^{-\mu} \cdot 2^l}{(1-2^{-\mu})^2} . \tag{23}$$

To help interpret the above, first ignore the factor $\mathsf{Coll}[\mathfrak{D}] \cdot |S|$ in the expression. Then the above is similar to what we had in the uniform case, where $|S| = 2^\mu$; the difference is that there is a

factor of $(1-2^{-\mu})^2$ now, but as long as the min-entropy is not very close to zero, the bound is still good. Looking at the $\mathsf{Coll}[\mathfrak{D}] \cdot |S|$ factor now, we note it is a fact (which follows from the Cauchy-Schwarz inequality) that $1/|S| \leq \mathsf{Coll}[\mathfrak{D}]$, with equality just when A_{m} puts a uniform distribution on S. The closer these quantities, the better our bound.

To derive (23), we generalize our previous approach and consider a simplified game that works as follows:

Game 2
$h_1, \ldots, h_M \xleftarrow{\$} \{0,1\}^l$
$I \xleftarrow{\mathfrak{D}} \{1, \ldots, M\}$
$C \leftarrow \{i \neq I \,:\, h_i = h_I\}$
Return $\sum_{i \in C} p_i$

Above, "$I \xleftarrow{\mathfrak{D}} \{1, \ldots, M\}$" means that I is assigned a value from $\{1, \ldots, M\}$ according to distribution $\mathfrak{D}$. Intuitively, regarding I as the index of the challenge message, Game 2 returns the "weight" of the set of messages that have the same hash as the challenge message (not including the latter) with respect to $\mathfrak{D}$. As before, this will help us determine A's maximal advantage. Now define for all $i = 1$ to M and $j = 1$ to M subject to $j \neq i$ the random variables

$$X_{i,j}(h_1, \ldots, h_M) = \begin{cases} p_i & \text{if } h_i = h_j, \\ 0 & \text{O.W.} \end{cases}$$

and $X_i(h_1, \ldots, h_M) = \sum_{j \neq i} X_{i,j}(h_1, \ldots, h_M)$. So Game 2 returns the value of X_I. Note that, for fixed i, the $X_{i,j}$ are pairwise independent (as j varies). Some basic calculations we will need are as follows (where, as before, all free variables have implicit universal quantification):

$$\mathbf{E}[X_{i,j}] = \frac{p_j}{2^l}\,; \quad \frac{1-p}{2^l} \leq \mathbf{E}[X_i] = \frac{\sum_{j \neq i} p_j}{2^l} \leq \frac{1}{2^l}\,.$$

$$\mathbf{Var}[X_{i,j}] = \frac{p_j^2}{2^l} - \frac{p_j^2}{2^{2l}} \leq \frac{p_j^2}{2^l}\,; \quad \mathbf{Var}[X_i] = \sum_{j \neq i} \mathbf{Var}[X_{i,j}] \leq \frac{\sum_{j \neq i} p_j^2}{2^l}\,.$$

Of these, the top left is a simple computation. In the top right, we use linearity of expectation for the equality, and the fact that $\sum_{i=1}^{M} p_i = 1$ for the upper bound and $p_i \leq p$ for the lower. Moving to the bottom left, we first apply the definition of variance and computations of $\mathbf{E}[X]$ and $\mathbf{E}\left[X^2\right]$, and then drop the second term for the upper-bound. In the bottom right, we use pairwise independence for the equality and the previous calculation for the upper-bound.

Now for some $0 \le \alpha \le 1$ to be determined later, we claim the following inequalities, which we then return to justify below:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH}',A} &= \Pr\left[\, \mathbf{Exp}^{\text{priv-1}}_{\mathsf{EaH}',A} \Rightarrow 1 \,\right] - \Pr\left[\, \mathbf{Exp}^{\text{priv-0}}_{\mathsf{EaH}',A} \Rightarrow 1 \,\right] \\
&\le \Pr\left[\, \mathbf{Exp}^{\text{priv-1}}_{\mathsf{EaH}',A} \Rightarrow 1 \,\right] \\
&\le \mathbf{E}\left[\, \frac{p}{p_I + X_I} \,\right] \\
&= p \cdot \sum_{i=1}^{M} \mathbf{E}\left[\, \frac{1}{p_i + X_i} \;\mid\; I = i \,\right] \cdot \Pr[\, I = i \,] \\
&= p \cdot \sum_{i=1}^{M} \mathbf{E}\left[\, \frac{1}{p_i + X_i} \;\mid\; I = i \,\right] \cdot p_i \\
&\le p \cdot \sum_{i=1}^{M} \left(\Pr[\, X_i \;\le\; \alpha \cdot \mathbf{E}[\, X_i \,]\,] \cdot \frac{1}{p_i} \;+\; \Pr[\, X_i \;>\; \alpha \cdot \mathbf{E}[\, X_i \,]\,] \cdot \frac{1}{\alpha \cdot \mathbf{E}[\, X_i \,]} \right) p_i \\
&\le p \cdot \sum_{i=1}^{M} \Pr[\, X_i \;\le\; \alpha \cdot \mathbf{E}[\, X_i \,]\,] \;+\; \frac{p_i}{\alpha \cdot \mathbf{E}[\, X_i \,]} \\
&\le p \cdot \sum_{i=1}^{M} \Pr[\, X_i \;\le\; \alpha \cdot \mathbf{E}[\, X_i \,]\,] \;+\; \frac{p \cdot 2^l}{\alpha \cdot (1-p)} \,.
\end{aligned}$$

Of these, the first line is by definition and the second is just dropping the second term.

The third line follows by considering runs of Game 2 and the priv-1 experiment over a common finite set of coins. When executed by the experiment, the best strategy for A_{g} to guess challenge message x is output a plaintext x' from S whose hash value (when concatenated with pk) is equal to its input $H(pk\|x)$ and that has the highest probability of being output by A_{m} over all such plaintexts. Regarding S as indexed from 1 to M and I as the index of the challenge message, we see that the probability of success of this strategy is exactly $p_I / \sum_j p_j$, where j in the sum here and below is taken over all indices such that $H(pk\|x_j) = H(pk\|x_I)$. Now when executed on the same coins, Game 2 returns $\sum_j p_j - p_I$, the value of X_I. So the probability that the experiment outputs 1 is at most $\mathbf{E}[\, p/(p_I + X_I)\,]$ (using $p_I \le p$ in the numerator).

The fourth line is expansion by conditional expectation, and the fifth is by definition of p_i. For the sixth, we use a conditioning argument, noting that if $X_i \le \alpha \cdot \mathbf{E}[\, X_i \,]$ then $p_i/(p_i + X_i)$ is at most p_i/p_i, while if $X_i > \alpha \cdot \mathbf{E}[\, X_i \,]$ then $p_i/(p_i + X_i)$ is at least $p_i/(\alpha \cdot \mathbf{E}[\, X_i \,])$. The seventh drops a positive factor less than 1. Finally, the last uses $p_i \le p$ and our previous lower-bound on $\mathbf{E}[\, X_i \,]$.

So we need to bound the probability on the last line. For this, we claim the following.

Claim C.2 For all $0 \le \alpha \le 1$ and $0 \le i \le M$, we have

$$\Pr[\, X_i \;\le\; \alpha \cdot \mathbf{E}[\, X_i \,]\,] \;\le\; \frac{\mathsf{Coll}[\, \mathfrak{D} \,] \cdot 2^l}{(1-\alpha)^2 \cdot (1-p)^2} \,.$$

Proof: We have

$$\begin{aligned}
\Pr[\, X_i \;\le\; \alpha \cdot \mathbf{E}[\, X_i \,]\,] &\le \Pr[\, |X_i - \mathbf{E}[\, X_i \,]| \;\ge\; (1-\alpha) \cdot \mathbf{E}[\, X_i \,]\,] \\
&\le \frac{\mathbf{Var}[\, X_i \,]}{(1-\alpha)^2 \cdot \mathbf{E}[\, X \,]^2} \\
&\le \frac{2^{2l} \cdot \sum_{j \neq i} p_j^2}{2^l \cdot (1-\alpha)^2 \cdot (1-p)^2} \\
&\le \frac{2^l \cdot \mathsf{Coll}[\, \mathfrak{D} \,]}{(1-\alpha)^2 \cdot (1-p)^2}
\end{aligned}$$

Above, the first line is algebraic manipulation. The second line is by Chebyshev's inequality. The third is substitution according to previous calculations. The last uses the fact that $\sum_{j \neq i} p_j^2 \;\le\; \mathsf{Coll}[\, \mathfrak{D} \,]$ for all i (because the former omits a positive term of the latter). ▮

Substituting according to the claim, we get

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{priv}}_{\mathsf{EaH}',A} &\le p \cdot \sum_{i=1}^{M} \frac{\mathsf{Coll}[\, \mathfrak{D} \,] \cdot 2^l}{(1-\alpha)^2 \cdot (1-p)^2} \;+\; \frac{p_i \cdot 2^l}{\alpha \cdot (1-p)} \\
&\le \frac{p \cdot \mathsf{Coll}[\, \mathfrak{D} \,] \cdot 2^l \cdot M}{(1-\alpha)^2 \cdot (1-p)^2} \;+\; \frac{p \cdot 2^l}{\alpha \cdot (1-p)} \;.
\end{aligned}$$

Then, setting $\alpha = 1/2$, using $(1-p)^2 \le 1-p$ to make denominators the same and $p \cdot 2^l \;\le\; p \cdot \mathsf{Coll}[\, \mathfrak{D} \,] \cdot 2^l \cdot M$ to make the numerators the same, and combining terms yields (23). This completes our analysis of the non-uniform case. ▮

D Proof of Theorem 7.1

Since EwH is deterministic, we assume wlog that the plaintext components of a vector $\mathbf{x}$ are always distinct for all $(\mathbf{x}, t)$ output by $A_{\mathrm{m}}(1^k)$. Ind-cpa adversary B for the proof is depicted in Figure 7. We consider the games depicted in Figure 8. (To prove Theorem 5.1 instead, the procedure in the games and the code of B to respond to A's decryption queries and the assignments to array E would simply be dropped.)

Equation (5) of Theorem 7.1 follows from the following sequence of inequalities, which we will justify below:

$$
\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{priv-cca}}_{\mathsf{DPE},A} &= \Pr\left[\, G_1^{A_{\mathrm{g}}} \Rightarrow b \,\right] && (24)\\
&\le \Pr\left[\, G_2^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \Pr[G_1^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_0\,] && (25)\\
&\le \Pr\left[\, G_2^{A_{\mathrm{g}}} \Rightarrow b \,\right] + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} && (26)\\
&\le \Pr\left[\, G_3^{A_{\mathrm{g}}} \Rightarrow b \,\right] + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} && (27)\\
&\le \Pr\left[\, G_4^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \Pr[G_3^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_1\,] + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} && (28)\\
&\le \Pr\left[\, G_4^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{h}}v}{2^{\mu}} + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} && (29)\\
&= \Pr\left[\, G_5^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{h}}v}{2^{\mu}} + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} && (30)\\
&\le \Pr\left[\, G_6^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{h}}v}{2^{\mu}} + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} + \Pr[G_6^{A_{\mathrm{g}}} \text{ sets } \mathsf{bad}_1\,] && (31)\\
&\le \Pr\left[\, G_6^{A_{\mathrm{g}}} \Rightarrow b \,\right] + \frac{q_{\mathrm{h}}v}{2^{\mu}} + q_{\mathrm{d}} \cdot \mathsf{mc}_{\mathsf{AE}} + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}} && (32)\\
&= \frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AE},B}(k) + \frac{q_{\mathrm{h}}v}{2^{\mu}} + q_{\mathrm{d}} \cdot \mathsf{mc}_{\mathsf{AE}} + 4q_{\mathrm{h}}v \cdot \mathsf{mpk}_{\mathsf{AE}}\,. && (33)
\end{aligned}
$$

The advantage of A is, as we have seen, 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. Intuitively, the game tries to pick values of $H[pk\|\mathbf{x}_i[j]]$ for $i \in \{0,1\}$ and $j \in \{1,\ldots,v\}$ used as coins for encryption upfront before running A_{g}, but in the case that A_{m} has already queried one of the indices to its RO the game uses its previous reply instead. We clarify that the two runs of A_{m} have different coins but the same array H is used for the RO, so they have the same RO. Game G_1 also sets a few flags, but they do not influence game output. We have justified (24).

Games G_1, G_2 differ only in statements that follow the setting of bad_0, meaning are, in the terminology of [14], identical-until-bad_0 games. The Fundamental Lemma of Game Playing [14] thus applies to justify (25). The probability that $H[pk\|\mathbf{x}_i[j]]$ for fixed $i \in \{0,1\}$ and $j \in \{1,\ldots,v\}$ is defined, meaning $pk\|\mathbf{x}_i[j]$ was queried by A_{m} to its RO, is at most $2q_{\mathrm{h}} \cdot \mathsf{mpk}_{\mathsf{AE}}$ because A_{m} gets no information about pk. (The factor of 2 is due to the fact that A_{m} is run twice.) Summing over all possible values of i, j, we obtain (26).

The Finalize procedure of Game G_3 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_2. 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_3 assigns d differently from G_2, the assignment made by G_3 is correct, meaning equal to b. Additionally, G_3 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 (27).

As before, the Fundamental Lemma of Game Playing [14] applies to justify (28). The probability that A_{g} makes a hash query $x \in \mathbf{x}_{1-b}$ when executed with G_3 is at most $q_{\mathrm{h}}v/2^{\mu}$ because A_{g} gets no information about $\mathbf{x}_{1-b}$. This justifies (29).

As in the proof of the Fundamental Lemma in [14], 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 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 (30). (To prove Theorem 5.1, we would basically be done at this point.)

Again, the Fundamental Lemma of Game Playing [14] applies to justify (31). When executed in Game G_5, the probability that a decryption query y made by A_{g} to is a valid ciphertext for some message m such that A_{g} has not queried $pk\|x$ to its hash oracle is at most $\mathsf{mc}_{\mathsf{AE}}$. This is because, without any information about $H[pk\|x]$, $H[pk\|x]$ and the coins used by $\mathcal{E}(pk, x)$ have the same distribution from the perspective of A_{g}. This implies (32).

Notice that the assignment to x to $\mathcal{D}(sk, y)$ in the procedure to respond to decryption queries in Game G_6 does not influence the output of the procedure. To make this explicit, we drop this assignment and the associated code from this procedure to yield an equivalent game G_7. Similarly to before, Game G_7 is now just the game defining the advantage of B written in a convenient way. The code for B does not set flags $\mathsf{bad}_0, \mathsf{bad}_1$ whereas G_7 makes some "If" checks and sets these flags, but these flags do not affect the game output. We have justified (33). Re-arranging yields (5).

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, in addition to the time to run A, B's time-complexity is dominated by the time needed to create a hash table containing the elements of $\mathbf{x}_0, \mathbf{x}_1$, which is $O(vn)$, as well as for encrypting each hash query made by A_{g}. So its additional overhead is $O(vn + q_{\mathrm{h}} T_{\mathcal{E}})$, as asserted. ▌

procedure Initialize Games $\boxed{G_1}$, $G_2 - G_8$

$b \xleftarrow{\$} \{0,1\}$
Run A_{m} on input 1^k twice:
  On query $H_1(z\|x)$:
  If $H_1[z\|x]$ is undefined then
    $H_1[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
  Return $H_1[z\|x]$
  On query $R(z\|x)$:
  If $R[z\|x]$ is undefined then
    $R[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
  Return $R[z\|x]$
  On query $H_2(z\|x)$:
  If $H_2[z\|x]$ is undefined then
    $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
  Return $H_2[z\|x]$
Let $(t_0, \mathbf{x}_0), (t_1, \mathbf{x}_1)$ be the outputs of A_{m}
$((N,e),(N,d)) \xleftarrow{\$} \mathcal{F}(1^k)$
For $i = 1$ to v do:
  $x_{i,l} \leftarrow \mathbf{x}_b[i][1 \ldots k_0]$
  $x_{i,r} \leftarrow \mathbf{x}_b[i][k_0 + 1 \ldots n]$
  $H^*_{i,1}, H^*_{i,2} \xleftarrow{\$} \{0,1\}^{k_0}$; $R^*_i \xleftarrow{\$} \{0,1\}^{n-k_0}$
  If $H_1[(N,e)\|x_{i,r}]$ is defined then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{H^*_{i,1} \leftarrow H_1[(N,e)\|x_{i,r}]}$
  If $R[(N,e)\|s_{i,0}]$ is defined then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{R^*_i \leftarrow R[(N,e)\|s_{i,0}]}$
  If $H_2[(N,e)\|t_{i,0}]$ is defined then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{H^*_{i,2} \leftarrow H_1[(N,e)\|t_{i,0}]}$
  $s_{i,0} \leftarrow H^*_{i,1} \oplus x_{i,l}$; $t_{i,0} \leftarrow R^*_i \oplus x_{i,r}$
  $s_{i,1} \leftarrow H^*_{i,2} \oplus s_{i,0}$
  $p_{i,l} \leftarrow (s_{i,1}\|t_{i,0})[1 \ldots n - k_1]$
  $y_i \leftarrow ((s_{i,1}\|t_{i,0})[n - k_1 + 1 \ldots n])^e \bmod N$
  $\mathbf{c}[i] \leftarrow p_{i,l}\|y_i$
Return $(1^k, (N,e), \mathbf{c})$

On query $H_1(z\|x)$: Games $\boxed{G_1, G_2}$, G_3

If $H_1[z\|x]$ is undefined then
  $H_1[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $z = (N,e)$ then
  If $\exists i$ such that $x = x_{i,r}$ then
    If $H_{i,2}[T_0]$ is defined then
      $H_1[z\|x] \leftarrow H^*_{i,1}$
    If $R[s_{i,0}]$ is defined then
      $\boxed{H_1[z\|x] \leftarrow H^*_{i,1}}$ /* bad_1 must be set */
Return $H_1[z\|x]$

On query $R(z\|x)$: Games $\boxed{G_1, G_2}$, G_3

If $R[z\|x]$ is undefined then
  $R[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
If $z = (N,e)$ then
  If $\exists i$ such that $x = s_{i,0}$ then
    If $H_2[t_{i,0}]$ is defined then
      $R[z\|x] \leftarrow R^*_i$
    If $H_1[z\|x_r]$ is undefined then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boxed{R[z\|x] \leftarrow R^*_i}$
Return $R[z\|x]$

On query $R(z\|x)$: Games $\boxed{G_3}$, G_4

If $R[z\|x]$ is undefined then
  $R[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
If $z = (N,e)$
  If $\exists i$ such that $x = s_{i,0}$ then
    If $H_2[z\|t_{i,0}]$ is defined then
      $R[z\|x] \leftarrow R^*_i$
    If $H_1[z\|x_{i,r}]$ is defined then
      $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boxed{R[z\|x] \leftarrow R^*_i}$
Return $R[z\|x]$

Figure 2: Start of the games for the proof of Theorem 5.2. Games whose names are boxed include the boxed statements, while un-boxed games do not. The games are continued in Figure 3. All games have the same Finalize procedure, namely **procedure Finalize**(g): If $g = t_0$ then Return 1, Else Return 0.

On query $H_2(z\|x)$: Games $\boxed{G_1 - G_4}$, G_5

If $H_2[z\|x]$ is undefined then
  $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $z = (N, e)$ then
  If $\exists i$ such that $x = t_{i,0}$ then
    If $R[z\|s_{i,0}]$ is defined then
      $\mathsf{bad}_3 \leftarrow \mathsf{true}$; $\boxed{H_2[z\|x] \leftarrow H^*_{i,2}}$
    Else $H_2[z\|x] \leftarrow H^*_{i,2}$
Return $H_2[z\|x]$

On query $H_2(z\|x)$: Games $\boxed{G_5}$, G_6

If $H_2[z\|x]$ is undefined then
  $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $z = (N, e)$ then
  If $\exists i$ such that $x = t_{i,0}$ then
    If $H_1[z\|x_{i,r}]$ is defined then
      $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boxed{H_2[z\|x] \leftarrow H^*_{i,2}}$
    Else $H_2[z\|x] \leftarrow H^*_{i,2}$
Return $H_2[z\|x]$

On query $H_1(z\|x)$: Games $\boxed{G_6}$, G_7, G_8

If $H_1[z\|x]$ is undefined then
  $H_1[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $z = (N, e)$ then
  If $\exists i$ such that $x = x_{i,r}$ then
    If $H_{i,2}[z\|T_0]$ is defined then
      $H_1[z\|x] \leftarrow H^*_{i,1}$
    Else $\boxed{H_1[z\|x] \leftarrow H^*_{i,1}}$
Return $H_1[z\|x]$

On query $H_2(z\|x)$: Games $\boxed{G_7}$, G_8

If $H_2[z\|x]$ is undefined then
  $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $z = (N, e)$ then
  If $\exists i$ such that $x = t_{i,0}$ then
    If $R[z\|s_{i,0}], H_1[z\|x_{i,r}]$ are undefined then
      $\mathsf{bad}_5 \leftarrow \mathsf{true}$; $\boxed{H_2[z\|x] \leftarrow H^*_{i,2}}$
Return $R[z\|x]$

Figure 3: Continuation of games for the proof of Theorem 5.2, started in Figure 2. Games whose names are boxed include the boxed statements, while un-boxed games do not. All games have the same Finalize procedure, namely **procedure Finalize**(g): If $g = t_0$ then Return 1, Else Return 0.

procedure Initialize

$b \xleftarrow{\$} \{0,1\}$
Run $A_{\rm m}$ on input 1^k twice:
  On query $H_1(z\|x)$:
  If $H_1[z\|x]$ is undefined then
    $H_1[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
  Return $H_1[z\|x]$
  On query $R(z\|x)$:
  If $R[z\|x]$ is undefined then
    $R[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
  Return $R[z\|x]$
  On query $H_2(z\|x)$:
  If $H_2[z\|x]$ is undefined then
    $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
  Return $H_2[z\|x]$
Let $(t_0, \mathbf{x}_0), (t_1, \mathbf{x}_1)$ be the outputs of $A_{\rm m}$
$((N,e),(N,d)) \xleftarrow{\$} \mathcal{F}$
For $i = 1$ to v do:
  $x_{i,l} \leftarrow \mathbf{x}_b[i][1 \ldots k_0]$
  $x_{i,r} \leftarrow \mathbf{x}_b[i][k_0+1 \ldots n]$
  $s_{i,1} \xleftarrow{\$} \{0,1\}^{k_0}$; $t_{i,0} \xleftarrow{\$} \{0,1\}^{n-k_0}$
  $p_{i,l} \leftarrow (s_{i,1}\|t_{i,0})[1 \ldots n-k_1]$
  $y_i \leftarrow ((s_{i,1}\|t_{i,0})[n-k_1+1 \ldots n])^e \bmod N$
  $\mathbf{c}[i] \leftarrow p_{i,l}\|y_i$
  $H^*_{i,2} \xleftarrow{\$} \{0,1\}^{k_0}$; $s_{i,0} \leftarrow H^*_{i,2} \oplus s_{i,1}$
  $R^*_i \leftarrow t_{i,0} \oplus x_{i,r}$; $H^*_{i,1} \leftarrow s_{i,0} \oplus x_{i,l}$
Return $(1^k, (N,e), \mathbf{c})$

On query $H_1(z\|x)$:
If $H_1[z\|x]$ is undefined then
  $H_1[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
Return $H_1[z\|x]$

On query $R(z\|x)$:
If $R[z\|x]$ is undefined then
  $R[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
If $\exists i$ such that $x = s_{i,0}$ then
  If $H_1[z\|x_r]$ is undefined then
    $\mathsf{bad}_1 \leftarrow \mathsf{true}$
  Else $\mathsf{bad}_2 \leftarrow \mathsf{true}$
Return $R[z\|x]$

On query $H_2(z\|x)$:
If $H_2[z\|x]$ is undefined then
  $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $\exists i$ such that $x = t_{i,0}$ then
  If $R[z\|s_{i,0}]$ is defined
    $\mathsf{bad}_3 \leftarrow \mathsf{true}$;
  Else If $H_1[z\|x_{i,r}]$ is defined then
    $\mathsf{bad}_4 \leftarrow \mathsf{true}$
  Else $\mathsf{bad}_5 \leftarrow \mathsf{true}$
Return $H_2[z\|x]$

---

procedure Finalize(g)
If $g = t_0$ then Return 1
Else Return 0

Figure 4: Game $G_{\rm rand}$ for the proof of Theorem 5.2.

Algorithm $\mathsf{GetQuery}(1^k, (N, e), y)$
$ctr \leftarrow 0$
$j \xleftarrow{\$} \{1, \ldots, q_{H_2}\}$; $w \xleftarrow{\$} \{1, \ldots, v\}$
$\mathbf{c} \xleftarrow{\$} (\{0,1\}^n)^{\times v}$ /* pick random v-size vector */
$c' \xleftarrow{\$} \{0,1\}^{n-k_1}$
$\mathbf{c}[w] \leftarrow c' \| y$
Run A_{g} on input $1^k, (N, e), \mathbf{c}$, replying to its oracle queries as follows:
On query $H_1(z\|x)$:
If $H_1[z\|x]$ is undefined then
$H_1[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
Return $H_1[z\|x]$
On query $R(z\|x)$:
If $R[z\|x]$ is undefined then
$R[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
Return $R[z\|x]$
On query $H_2(z\|x)$:
$ctr \leftarrow ctr + 1$
If $H_2[z\|x]$ is undefined then
$H_2[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
If $ctr = j$ then
$T \leftarrow x$
Return $H_2[z\|x]$
Until A_{g} halts
Return T

Figure 5: Algorithm $\mathsf{GetQuery}$ for proof of Theorem 5.2.

Algorithm $\mathsf{K}(1^k)$
$((N,e),(N,d)) \xleftarrow{\$} \mathcal{F}(1^k)$
$\mathbf{c} \xleftarrow{\$} (\{0,1\}^n)^{\times v}$ /* pick random v-size vector */
Run A_{g} on input $1^k, f, \mathbf{c}$, replying to its oracle queries as follows:
    On query $H_1(z\|x)$:
    If $H_1[z\|x]$ is undefined then
        If $z = (N,e)$ then add x to L_1
        $H_1[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
    Return $H_1[z\|x]$
    On query $R(z\|x)$:
    If $R[z\|x]$ is undefined then
        For all $z \in L_1$ add $x \oplus H_1[(N,e)\|z]\|z$ to L_2
        $R[z\|x] \xleftarrow{\$} \{0,1\}^{k_0}$
    Return $R[z\|x]$
    On query $H_2(z\|x)$:
    If $H_2[z\|x]$ is undefined then
        $H_2[z\|x] \xleftarrow{\$} \{0,1\}^{n-k_0}$
    Return $H_2[z\|x]$
Until A_{g} halts
Return L_2

Figure 6: "Knowledge extractor" K used in the proof of Theorem 5.2.

Adversary $B^{\mathcal{E}(1^k,pk,\mathrm{LR}(\cdot,\cdot,b))}(1^k, pk)$
Run A_{m} on input 1^k twice, replying to its oracle queries as follows:
  On query Hash$(s\|x)$:
  If $H[s\|x]$ is undefined then
    $H[s\|x] \xleftarrow{\$} \mathsf{Coins}_s(|x|)$
  Return $H[s\|x]$
Let $(\mathbf{x}_0, t_0), (\mathbf{x}_1, t_1)$ be the outputs of A_{m}
$\mathbf{c} \xleftarrow{\$} \mathcal{E}(1^k, pk, \mathrm{LR}(\mathbf{x}_0, \mathbf{x}_1, b))$
Run A_{g} on input $1^k, pk, \mathbf{c}$, replying to its oracle queries as follows:
  On query Hash$(s\|x)$:
  If $H[s\|x]$ is undefined then
    $H[s\|x] \xleftarrow{\$} \mathsf{Coins}_s(|x|)$
  If $s = pk$ then
    $E[z] \leftarrow \mathcal{E}(1^k, pk, z; H[s\|x])$
    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 $H[pk\|x]$
  On query Decrypt$(1^k, pk, sk, y)$:
  If $\exists z$ such that $E[z] = y$ then return z
  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 7: Ind-cpa adversary B for proof of Theorem 7.1.

procedure Initialize Games $\boxed{G_1}$, G_2, G_3–G_7

$b \xleftarrow{\$} \{0,1\}$
$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$
Run A_{m} on input 1^k twice:
  On query Hash$(s\|x)$:
  If $H[s\|x]$ is undefined then
    $H[s\|x] \xleftarrow{\$} \mathsf{Coins}_s(|x|)$
  Return $H[s\|x]$
Let $(\mathbf{x}_0, t_0), (\mathbf{x}_1, t_1)$ be the outputs of A_{m}
For $i = 0$ to 1 do:
  For $j = 1$ to v do:
  $R_{i,j} \xleftarrow{\$} \mathsf{Coins}_{pk}(|\mathbf{x}_i[j]|)$
  If $H[pk\|\mathbf{x}_i[j]]$ is defined then
    $\mathsf{bad}_0 \leftarrow \mathsf{true}$; $\boxed{R_{i,j} \leftarrow H[pk\|\mathbf{x}_i[j]]}$
  $\mathbf{y}_i[j] \leftarrow \mathcal{E}(1^k, pk, \mathbf{x}_i[j]; R_{i,j})$
Return $1^k, pk, \mathbf{y}_b$

On query Hash$(s\|x)$:
Games $\boxed{G_1 - G_4}$, G_5–G_7

If $H[s\|x]$ is undefined then
  $H[s\|x] \xleftarrow{\$} \mathsf{Coins}_s(|x|)$
If $s = pk$ then
  $E[z] \leftarrow \mathcal{E}(1^k, pk, x; H[s\|x])$
  If $\exists i$ such that $x = \mathbf{x}_0[i]$ then
    If $\mathsf{one} = \mathsf{false}$ then $\mathsf{zer} \leftarrow \mathsf{true}$
    $\boxed{H[s\|x] \leftarrow R_{0,i}}$
  If $\exists j$ such that $x = \mathbf{x}_1[j]$ then
    If $\mathsf{zer} = \mathsf{false}$ then $\mathsf{one} \leftarrow \mathsf{true}$
    $\boxed{H[s\|x] \leftarrow R_{1,j}}$
Return $H[s\|x]$

On query Decrypt(y):
Games $\boxed{G_1 - G_5}$, G_6

If $\exists z$ such that $E[z] = y$ then return z
$x \leftarrow \mathcal{D}(1^k, pk, sk, y)$
If $x = \perp$ then return $\perp$
If $H[pk\|x]$ is undefined then
  $H[pk\|x] \xleftarrow{\$} \mathsf{Coins}_{pk}(|x|)$
  $E[x] \leftarrow \mathcal{E}(1^k, pk, x; H[pk\|x])$
  If $E[x] = y$ then
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boxed{\text{Return } x}$
Return $\perp$

On query Decrypt(y): Game G_7

If $\exists z$ such that $E[z] = y$ then return z
Else return $\perp$

---

procedure Finalize(g) Games G_1, G_2

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

---

procedure Finalize(g) Games $\boxed{G_3}$, 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 $d \leftarrow 0$
If $(b = 1 \wedge \mathsf{zer} = \mathsf{true}) \vee (b = 0 \wedge \mathsf{one} = \mathsf{true})$
  then $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boxed{d \leftarrow b}$
Return d

Figure 8: Games for the proof of Theorem 7.1. There are a total of 7 games. The games whose names are boxed include the boxed statements, while un-boxed games do not.