Message-Locked Encryption and Secure Deduplication

Mihir Bellare[1] Sriram Keelveedhi[2] Thomas Ristenpart[3]

October 2012

Abstract

We formalize a new cryptographic primitive, Message-Locked Encryption (MLE), where the key under which encryption and decryption are performed is itself derived from the message. MLE provides a way to achieve secure deduplication (space-efficient secure outsourced storage), a goal currently targeted by numerous cloud-storage providers. We provide definitions both for privacy and for a form of integrity that we call tag consistency. Based on this foundation, we make both practical and theoretical contributions. On the practical side, we provide ROM security analyses of a natural family of MLE schemes that includes deployed schemes. On the theoretical side the challenge is standard model solutions, and we make connections with deterministic encryption, hash functions secure on correlated inputs and the sample-then-extract paradigm to deliver schemes under different assumptions and for different classes of message sources. Our work shows that MLE is a primitive of both practical and theoretical interest.

Keywords: Convergent encryption, deduplication, deterministic encryption.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@eng.ucsd.edu. URL: http://cseweb.ucsd.edu/~mihir/. Supported in part by NSF grants CNS-0904380, CCF-0915675 and CNS-1116800.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: sriramkr@cs.ucsd.edu. URL: http://www.cs.ucsd.edu/users/skeelvee/. Supported in part by Bellare's grants.

[3] Department of Computer Sciences, University of Wisconsin–Madison, 1210 West Dayton Street, Madison, Wisconsin 53715, USA. Email: rist@cs.wisc.edu. URL: http://pages.cs.wisc.edu/~rist/. Supported in part by NSF grant CNS-1065134 and generous gifts from RSA Labs and Microsoft.

Contents

1 Introduction **3**
- 1.1 Background . . . 3
- 1.2 Definitions and Relations . . . 4
- 1.3 Practical Contributions . . . 5
- 1.4 Theoretical Contributions . . . 5
- 1.5 Further Remarks and Related Work . . . 7

2 Preliminaries **7**

3 Message-Locked Encryption **8**

4 Practical Contributions: The Security of Fast MLE Schemes **11**

5 Theoretical Contributions: Constructions without ROs **13**
- 5.1 Extract-Hash-Check . . . 13
- 5.2 Sample-Extract-Encrypt . . . 14

A Relations Between MLE Privacy Notions **19**

B Proof of PRV$-CDA for CE, HCE1, HCE2, RCE **21**

C Instantiations and Performance of CE, HCE2, and RCE **25**

D Proof of Theorem 5.1 **27**

E Proof of Theorem 5.2 **28**

1 Introduction

We introduce an intriguing new primitive that we call Message-Locked Encryption (MLE). An MLE scheme is a symmetric encryption scheme in which the key used for encryption and decryption is itself derived from the message. Instances of this primitive are seeing widespread deployment and application for the purpose of secure deduplication [1, 2, 4, 12, 17–19, 24, 26, 35, 37, 41, 45, 48], but in the absence of a theoretical treatment, we have no precise indication of what these methods do or do not accomplish.

We provide definitions of privacy and integrity peculiar to this domain. Now having created a clear, strong target for designs, we make contributions that may broadly be divided into two parts: (1) practical and (2) theoretical. In the first category we analyze existing schemes and new variants, breaking some and justifying others with proofs in the random-oracle-model (ROM) [10]. In the second category we address the challenging question of finding a standard-model MLE scheme, making connections with deterministic public-key encryption [5], correlated-input-secure hash functions [29] and locally-computable extractors [3, 36, 46] to provide schemes exhibiting different tradeoffs between assumptions made and the message distributions for which security is proven. From our treatment MLE emerges as a primitive that combines practical impact with theoretical depth and challenges, making it well worthy of further study and a place in the cryptographic pantheon. Below we begin with some background and then look more closely at our contributions.

1.1 Background

To save space, commercial cloud storage services such as Google Drive [28], Dropbox [23] and bitcasa [12] perform file-level deduplication across all their users. Say a user Alice stores a file M and Bob requests to store the same file M. Observing that M is already stored, the server, instead of storing a second copy of M, simply updates metadata associated to M to indicate that Bob and Alice both stored M. In this way, no file is stored more than once, moving storage costs for a file stored by u users from $\mathcal{O}(u \cdot |M|)$ to $\mathcal{O}(u + |M|)$ where the big-O notation hides implementation-dependent constants.

However, as users we may want our files to be encrypted. We may not want the storage provider to see our data. Even if we did trust the provider, we may legitimately worry about errant employees or the risk of server compromise by an external adversary. When users themselves are corporations outsourcing their data storage, policy or government regulation may mandate encryption.

Conventional encryption, however, makes deduplication impossible. Say Alice stores not her file M but its encryption C_A under her password pw_A. Bob would store C_B, the encryption of M under his password pw_B. Two issues arise: (1) how the server is to detect that the data underlying the two ciphertexts is the same, and (2) even if it can so detect, what can it store short of (C_A, C_B) that allows both parties, based on their separate respective passwords, to recover the data from what is stored. Standard IND-CPA encryption means even (1) is not possible. We might use some kind of searchable encryption [5, 15, 44] but it is still not clear how to solve (2). Just storing Alice's ciphertext, for example, does not work because Bob cannot later decrypt it to recover the file, and visa versa.

Douceur et. al. (DABST) [22] proposed a clever solution called convergent encryption (CE). Alice derives a key $K = H(M)$ from her message M and then encrypts the message as $C = E(K, M) = E(H(M), M)$, where H is a cryptographic hash function and E is a block cipher. (They assume the message is one block long.) The ciphertext is given to the server and the user retains K. Since encryption is deterministic, if Bob starts from the same message he would produce the same key and ciphertext. The server can now perform deduplication on the ciphertext C, checking, when it receives C, whether or not it is already stored, and, if the latter, as before, not re-storing but instead updating meta-data to indicate an additional owner. Both Alice and Bob can decrypt C since both have the same key K.

These ideas have been attractive enough to see significant usage, with CE or variants deployed in [2, 12, 17, 24, 26, 37, 41, 45, 48]. It is not however clear what precisely is the underlying security goal and whether deployed schemes achieve it.

1.2 Definitions and Relations

We introduce Message-Locked Encryption (MLE) —so named because the message is locked, as it were, under itself— with the goal of providing an encryption primitive that provably enables secure deduplication.

Syntax. As depicted in Figure 2, the key generation algorithm of an MLE scheme $\mathcal{K}$ maps a message M to a key K. The encryption algorithm $\mathcal{E}$ takes input the key K and a message M and produces a ciphertext C. The decryption algorithm $\mathcal{D}$ allows recovery of M from C given the key K. The tagging algorithm $\mathcal{T}$ maps the ciphertext C to a tag T used by the server to detect duplicates. (Tag correctness requires that tags corresponding to messages M_1, M_2 are likely to be the same iff M_1, M_2 are the same.) All algorithms may depend on a parameter P but the latter is public and common to all parties including the adversary, and thus is not a key.

Any MLE scheme enables deduplication of ciphertexts. CE is captured by our syntax as the MLE scheme that lets $K = H(M)$, $C = E(K, M)$ and tag $T = H(C)$.

MLE is trivially achieved by letting the key K equal the message M. (Set $C = T = \varepsilon$ to the empty string and have decryption simply return the key.) This degenerate solution is however useless for deduplication since the client stores as K the entire file and no storage savings result. We rule it out by requiring that keys be shorter than messages, ideally keys are of a fixed, short length.

Privacy. No MLE scheme can achieve semantic-security-style privacy in the spirit of [7, 27]. Indeed, if the target message M is drawn from a space S of size s then an adversary, given an encryption C of M, can recover M in $\mathcal{O}(s)$ trials. (For each candidate $M' \in S$ test whether $\mathcal{D}(\mathcal{K}(M'), C) = M'$ and if so return M'.) As with deterministic public-key encryption [5], we therefore ask for the best possible privacy, namely semantic security when messages are unpredictable (have high min-entropy). Adapting definitions from [5,6,8,16] we formalize a PRV-CDA notion where encryptions of two unpredictable messages should be indistinguishable. ("cda" stands for "chosen-distribution attack" [6].) We also formalize a stronger PRV$-CDA notion where the encryption of an unpredictable message must be indistinguishable from a random string of the same length (cf. [43]).

These basic notions are for non-adaptive adversaries. We also have corresponding adaptive versions PRV-CDA-A and PRV$-CDA-A. We show that PRV-CDA does not imply PRV-CDA-A but, interestingly, that PRV$-CDA does imply PRV$-CDA-A. (See Figure 2 for a comprehensive relations summary.) Thus PRV$-CDA emerges as the preferred target for designs because non-adaptive security is easier to prove yet adaptive security is implied.

Tag consistency. Suppose client Alice has a message M_A and client Bob has a different message M_B. Alice is malicious and uploads not an honest encryption of M_A but a maliciously-generated ciphertext C_A such that, when Bob tries to upload C_B, the server sees a tag match $\mathcal{T}(C_A) = \mathcal{T}(C_B)$. (This does not contradict the correctness requirement that tags are usually equal iff the messages are equal because that holds for honestly-generated ciphertexts.) The server thus keeps only C_A, deleting C_B. Yet later, when Bob downloads to get C_A, the decryption is M_A, not M_B, meaning the integrity of his data has been compromised.

This is a serious concern, and not mere speculation, for such "duplicate-faking" attacks have been found on some CE variants [45]. We define tag consistency to rule out these types of integrity violations. Notion TC asks that it be hard to create (M, C) such that $\mathcal{T}(C) = \mathcal{T}(\mathcal{E}(\mathcal{K}(M), M))$ but $\mathcal{D}(\mathcal{K}(M), C)$ is a string different from M. In words, an adversary cannot make an honest client recover an incorrect message, meaning one different from the one it uploaded. Notion STC ("S" for "strong") asks that it additionally be hard to create (M, C) such that $\mathcal{T}(C) = \mathcal{T}(\mathcal{E}(\mathcal{K}(M), M))$ but $\mathcal{D}(\mathcal{K}(M), C) = \perp$, meaning an adversary cannot erase an honest client's message. STC is strictly stronger than TC; we define both because, as we will see, some schemes meet only the weaker, but still meaningful, TC version.

1.3 Practical Contributions

The definitional framework outlined above puts us in a position to rigorously assess —a decade after its inception in [22]— the security of convergent encryption (CE). The task is complicated by the presence and deployment of numerous variants of the basic CE idea. We address this by formulating two MLE schemes, that we call CE and HCE1, that represent two major variants of CE and between them capture the prominent existing schemes. They each make use of a RO hash function H and a deterministic symmetric encryption scheme SE. CE with SE set to a blockcipher, for example, is the scheme of [22] and HCE1 with SE as a blockcipher in counter mode with fixed IV is the scheme of the Tahoe FileSystem [48].

CE sets $K = H(M)$, $C = \mathsf{SE}(K, M)$ and tag $T = H(C)$, while HCE1 sets $K = H(M)$, $C = \mathsf{SE}(K, M)\|H(K)$ and $T = H(K)$. The rationale for HCE1 is to offer better performance for the server who can simply read the tag as the second part of the ciphertext rather than needing to compute it by hashing the possibly long ciphertext. But we observe that HCE1 is vulnerable to duplicate faking attacks, meaning it does not even achieve TC security. We have reported this attack to the developers of the Tahoe FileSystem [48], who deemed it important enough that they propose to add patches.

We ask whether performance gains of the type offered by HCE1 over CE can be obtained without loss in consistency, and offer as answers two new schemes, HCE2 and RCE. The former is as efficient as HCE1. RCE however is even more efficient, needing just one concerted pass over the data to generate the key, encrypt the message and produce the tag. On the other hand, HCE2 needs two passes, one pass to generate the key and a second for encryption, while CE needs a third pass for producing the tag. RCE achieves this via a novel use of randomization (all previous schemes were deterministic). Roughly (see Figure 4), encryption picks a fresh random key L and then computes $\mathsf{SE}(L, M)$ and $K = H(M)$ in the same pass, finally placing an encryption of L under K, together with an appropriate tag, in the ciphertext. We have implemented all three schemes and the results (cf. Section C) show that RCE does indeed outperform the other two.

Figure 1 (table, first four rows) summarizes the findings of our security analysis of the four schemes. Under standard assumptions on the deterministic symmetric encryption scheme SE (one-time real-or-random ciphertext, or ROR, security as well as key-recovery security) and with H a RO, we show that all four MLE schemes meet our strong privacy notion PRV$-CDA. The consistency findings are more involved. As mentioned, HCE1 provides no tag consistency. The good news is that CE, HCE2 and RCE all achieve TC security, so that an adversary cannot make a client recover a file different from the one she uploaded. But only CE offers STC security, implying that the reduction in server cost offered by HCE1, HCE2 and RCE comes at a price, namely loss of STC-security. The conclusion is that designers will need to trade performance for strong tag consistency.

1.4 Theoretical Contributions

Is standard-model MLE possible? This emerges as the natural and most basic theoretical question in this domain. Another question is, how does MLE relate to other (existing) primitives? MLE has in common with Deterministic Public-Key Encryption (D-PKE) [5] and Correlated-input-secure Hash Functions (CI-H) [29] a goal of privacy on unpredictable but possibly related inputs, so it is in particular natural to ask about the relation of MLE to these primitives. The two questions are related, for showing that a primitive X implies MLE yields a construction of an MLE scheme based on X. In exploring these questions it is instructive to distinguish between D-MLE (where encryption is deterministic) and R-MLE (where encryption may be randomized). The connections we now discuss are summarized by the picture on the right side of Figure 1:

- D-PKE $\Rightarrow$ D-MLE: We show how to construct an MLE scheme from any D-PKE scheme that is PRIV-secure in the sense of [5]. The first idea that may come to mind is to make public a public key pk for the D-PKE scheme DE and MLE-encrypt M as $\mathsf{DE}(pk, M)$. But this does not make sense because sk is needed to decrypt and the latter is not derived from M. Our XtDPKE ("extract-then-D-PKE" solution, specified and proven in Section 5, is quite different and does not exploit the decryptability

Scheme	Model	D/R	Privacy		Integrity	
			PRV-CDA	PRV$-CDA	TC	STC
CE	RO	D	✓	✓	✓	✓
HCE1	RO	D	✓	✓	✗	✗
HCE2	RO	D	✓	✓	✓	✗
RCE	RO	R	✓	✓	✓	✗
XtCIH	STD	D	✓	✓	✓	✓
XtDPKE	STD	D	✓	✗	✓	✓
XtESPKE	STD	R	✓	✗	✓	✓
SXE	STD	D	✓	✓	✓	✓

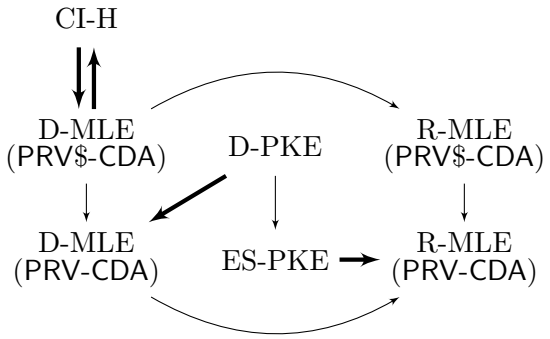


Figure 1: **Left:** For each MLE scheme that we construct, we indicate whether it is in the RO or standard model; whether it is deterministic or randomized; and which security properties it is proven to possess. The assumptions for XtCIH, XtDPKE and XtESPKE are, respectively, a CI-H function, a D-PKE scheme and an ES-PKE scheme, while the others assume only a symmetric encryption scheme. **Right:** An arrow X $\rightarrow$ Y means we can construct primitive Y from primitive X. Dark arrows are our results while light arrows indicate trivial or known implications.

of DE at all. We apply a strong randomness extractor to M to get the MLE key K and then encrypt M bit-by-bit, the encryption of the i-th bit $M[i]$ being $C[i] = \mathsf{DE}(pk, K\|i\|M[i])$. Decryption, given K, is done by re-encrypting, for each i, both possible values of the i-th message bit and seeing which ciphertext matches $C[i]$. We assume a trusted generation of pk in which nobody retains sk. XtDPKE has PRV-CDA privacy and provides STC (strong) tag consistency.

- CI-H $\Leftrightarrow$ D-MLE: Our XtCIH ("extract-then-CI-Hash") scheme derives a D-MLE scheme from any CI-H hash function [29] by using the latter in place of the D-PKE scheme in the above. XtCIH is PRV$-CDA private while retaining STC consistency. Conversely, any PRV$-CDA D-MLE scheme can be used to construct a CI-H hash function, making the primitives equivalent.

We believe these results are interesting as connections between prominent primitives. However, they do not, right now, yield MLE schemes under standard assumptions because providing the required D-PKE schemes or CI-H functions under such assumptions is still open and deemed challenging. Indeed, Wichs [47] shows that secure D-PKE schemes or CI-H functions may not be obtained via blackbox reductions from any assumption that may be modeled as a game between an adversary and a challenger. We note that his result applies to D-MLE as well but, as far as we can tell, not to R-MLE. One potential route to MLE with standard assumptions may thus be to exploit randomization but we are unaware of how to do this beyond noting that XtDPKE extends to a R-MLE scheme XtESPKE based on any ES-PKE (Efficiently Searchable PKE) scheme [5], a weaker primitive than D-PKE.

In the D-PKE domain, progress was made by restricting attention to special message distributions. In particular D-PKE under standard assumptions have been achieved for independent messages or block sources [8, 14, 16, 25]. CI-H functions have been built for messages given by polynomials evaluated at the same random point [29]. It is thus natural to ask whether we can obtain MLE under standard assumptions for special message distributions. One might think that this follows from our D-PKE $\Rightarrow$ D-MLE and CI-H $\Rightarrow$ D-MLE constructions and the known results on D-PKE and CI-H, but this is not the case because our constructions do not preserve the message distribution.

The final contribution we mention here is MLE schemes under standard assumptions for certain classes of message distributions . Our SXE (Sample-extract-encrypt) MLE scheme is inspired by locally-computable extractors [3, 36, 46] and the sample-then-extract paradigm [40, 46]. The idea is to put a random subset of the message bits through an extractor to get a key used to encrypt the rest of the bits, and the only assumption made is a standard, ROR-secure symmetric encryption scheme.

1.5 Further Remarks and Related Work

Recall we introduced an indistinguishability-from-random notion PRV$-CDA for MLE and showed that it implied its adaptive counterpart. This is of broader interest for the parent settings of deterministic and hedged encryption. Here achieving adaptive security has been challenging [6]. We suggest that progress can be made by defining and then targeting indistinguishability-from-random style definitions.

Mironov, Pandey, Reingold and Segev [39] suggest deduplication as a potential application of their incremental deterministic public-key encryption scheme. But this will not work for multi-user deduplication settings, since all users would have to share the secret key.

Recent work showed that client-side deduplication gives rise to side-channel attacks because users are told if another user already uploaded a file [32]. MLE is compatible with either client- or server-side deduplication (the latter prevents such side-channels). MLE targets a different class of threats than proofs of ownership [31], which were proposed for deduplication systems in order to mitigate abuse of services for surreptitious content distribution.

2 Preliminaries

Notations and Conventions. The empty string is denoted by ε. If $\mathbf{x}$ is a vector then $|\mathbf{X}|$ denotes the number of components in $\mathbf{x}$, $\mathbf{x}[i]$ denotes the i-th component, and $\mathbf{x}[i,j] = \mathbf{x}[i] \ldots \mathbf{x}[j]$ for $1 \le i \le j \le |\mathbf{x}|$. A (binary) string x is identified with a vector over $\{0,1\}$ so that $|x|$ is its length, $x[i]$ is its i-th bit and $x[i,j] = x[i] \ldots x[j]$ for $1 \le i \le j \le |x|$. If S is a finite set then $|S|$ denotes its size and $s \leftarrow\!\!\$\, S$ denotes picking an element uniformly from S and assigning it to s. For $i \in \mathbb{N}$ we let $[i] = \{1, \ldots, i\}$. We denote by $\lambda \in \mathbb{N}$ the security parameter and by 1^λ its unary representation.

Algorithms are randomized unless otherwise indicated. "PT" stands for "polynomial-time." By $y \leftarrow A(x_1, \ldots; R)$, we denote the operation of running algorithm A on inputs $x_1, \ldots$ and coins R and letting y denote the output. By $y \leftarrow\!\!\$\, A(x_1, \ldots)$, we denote the operation of letting $y \leftarrow A(x_1, \ldots; R)$ with R chosen at random. We denote by $[A(x_1, \ldots)]$ the set of points that have positive probability of being output by A on inputs $x_1, \ldots$. Adversaries are algorithms or tuples of algorithms. In the latter case, the running time of the adversary is the sum of the running times of all the algorithms in the tuple.

The guessing probability $\mathbf{GP}(X)$ and min-entropy $\mathbf{H}_\infty(X)$ of a random variable X are defined via $\mathbf{GP}(X) = \max_x \Pr[X = x] = 2^{-\mathbf{H}_\infty(X)}$. The conditional guessing probability $\mathbf{GP}(X \,|\, Y)$ and conditional min-entropy $\mathbf{H}_\infty(X \,|\, Y)$ of a random variable X given a random variable Y are defined via $\mathbf{GP}(X \,|\, Y) = \sum_y \Pr[Y = y] \cdot \max_x \Pr[X = x | Y = y] = 2^{-\mathbf{H}_\infty(X \,|\, Y)}$. By $\mathsf{SD}(X; Y)$ we denote the statistical distance between random variables X and Y.

Game playing framework. For our security definitions and proofs we use the code-based game playing framework of [11], though adopting some of the syntax and semantics of [42]. A game G (Figure 3, for example) has a main procedure, and possibly other procedures. G begins by executing main, which runs an adversary A after some initialization steps. A can make oracle calls to procedures permitted by G. When A finishes executing, G performs further steps with A's output to produce some output itself. We denote by $G^A \Rightarrow y$ the event that an execution of G with A outputs y. We abbreviate $G^A \Rightarrow \mathsf{true}$ as G^A. Boolean flags are assumed initialized to false and sets to $\emptyset$. The running time of the adversary A is a function of λ defined as the total running time of the game with the adversary in expectation, including the procedures of the game, and will be denoted by $\mathbf{T}_A(\cdot)$ when the game is clear from the context.

CR hashing. A family of hash functions $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ is a pair of PT algorithms, the second deterministic. The key generation algorithm takes input 1^λ and returns a hashing key K_h. The hashing algorithm $\mathcal{H}$ takes K_h and a message M and returns its hash $H \leftarrow \mathcal{H}(K_h, M)$. We say that H is injective if the function $\mathcal{H}(K_h, \cdot)$ is injective for every K_h. Let game CR_H choose a key $K_h \leftarrow\!\!\$\, \mathcal{HK}(1^\lambda)$, run an adversary $A(K_h)$, and check if A's output of M, M' are such that $M \neq M'$ and $\mathcal{H}(K_h, M) = \mathcal{H}(K_h, M')$. Advantage is defined as $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},A}(\lambda) = \Pr[\mathrm{CR}^A_\mathsf{H}(\lambda)]$ and we say that H is CR-secure if $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},A}(\cdot)$ is a negligible function for any PT A.

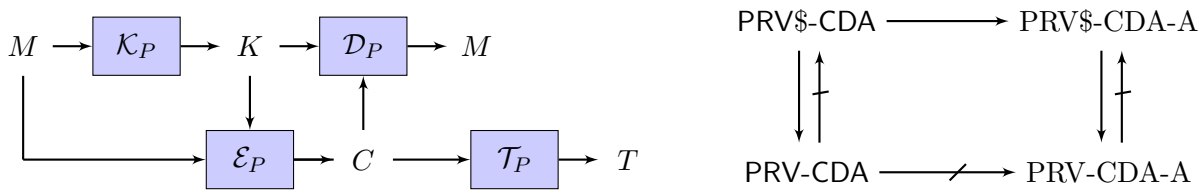


Figure 2: **Left:** Depiction of syntax of MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$. The parameter generation algorithm is not shown. **Right:** Relations between notions of privacy for MLE schemes. An arrow from A to B means that any A-secure MLE scheme is also B-secure. A barred arrow means there is an A-secure MLE scheme that is not B-secure.

3 Message-Locked Encryption

Syntax and correctness. An MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ is a five-tuple of PT algorithms, the last two deterministic — see Figure 2. On input 1^λ the parameter generation algorithm $\mathcal{P}$ returns a public parameter P. On input P and a message M, the key-generation algorithm $\mathcal{K}$ returns a message-derived key $K \leftarrow\!\!\$\ \mathcal{K}_P(M)$. On inputs P, K, M the encryption algorithm $\mathcal{E}$ returns a ciphertext $C \leftarrow\!\!\$\ \mathcal{E}_P(K, M)$. On inputs P, K and a ciphertext C, the decryption algorithm $\mathcal{D}$ returns $\mathcal{D}_P(K, C) \in \{0,1\}^* \cup \{\perp\}$. On inputs P, C the tag generation algorithm returns a tag $T \leftarrow \mathcal{T}_P(C)$. Associated to the scheme is a *message space* $\mathrm{MsgSp}_{\mathsf{MLE}}$ that associates to any $\lambda \in \mathbb{N}$ a set $\mathrm{MsgSp}_{\mathsf{MLE}}(\lambda) \subseteq \{0,1\}^*$. We require that there is a function Cl such that, for all $\lambda \in \mathbb{N}$, all $P \in [\mathcal{P}(1^\lambda)]$ and all $M \in \{0,1\}^*$, any output of $\mathcal{E}_P(\mathcal{K}_P(M), M)$ has length $\mathsf{Cl}(P, \lambda, |M|)$, meaning the length of a ciphertext depends on nothing about the message other than its length. The *decryption correctness* condition requires that $\mathcal{D}_P(K, C) = M$ for all $\lambda \in \mathbb{N}$, all $P \in [\mathcal{P}(1^\lambda)]$, all $M \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$, all $K \in [\mathcal{K}_P(M)]$ and all $C \in [\mathcal{E}_P(K, M)]$. The *tag correctness* condition requires that there is a negligible function $\delta\colon \mathbb{N} \to [0,1]$, called the false negative rate, such that $\Pr[\mathcal{T}_P(C) \neq \mathcal{T}_P(C')] \leq \delta(\lambda)$ for all $\lambda \in \mathbb{N}$, all $P \in [\mathcal{P}(1^\lambda)]$ and all $M \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$, where the probability is over $C \leftarrow\!\!\$\ \mathcal{E}_P(\mathcal{K}_P(M), M)$ and $C' \leftarrow\!\!\$\ \mathcal{E}_P(\mathcal{K}_P(M), M)$. We say that MLE is deterministic if $\mathcal{K}$ and $\mathcal{E}$ are deterministic. We observe that if MLE is deterministic then it has perfect tag correctness, meaning a false negative rate of 0.

Discussion. In the application to secure deduplication, the server publishes P and maintains a database that we view as a table Da, initially everywhere $\perp$. In the UPLOAD protocol, the client, having P, M, computes $K \leftarrow\!\!\$\ \mathcal{K}_P(M)$ and $C \leftarrow\!\!\$\ \mathcal{E}_P(K, M)$. The client stores K securely. (It may do so locally or store K encrypted under its password on the server, but the implementation is not relevant here.) It sends C to the server. The latter computes $T \leftarrow \mathcal{T}_P(C)$. If $\mathsf{Da}[T] = \perp$ then it lets $\mathsf{Da}[T] \leftarrow C$. The server provides the client with a filename or pointer that we may, for simplicity, just view as the tag T. In the DOWNLOAD protocol, the client sends the server a tag T and the server returns $\mathsf{Da}[T]$. If Alice uploads M and Bob later does the same, tag correctness means that their tags will most likely be equal and the server will store a single ciphertext on their behalf. Downloads will return to both this common ciphertext C, and decryption correctness guarantees that both can decrypt C under their respective (although possibly different) keys to recover M.

A trivial construction of an MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ may be obtained by setting the key to the message. In more detail, let $\mathcal{P}(1^\lambda) = \varepsilon$; let $\mathcal{K}_\varepsilon(M) = M$; let $\mathcal{E}_\varepsilon(M, M) = \mathcal{T}_\varepsilon(C) = \varepsilon$; let $\mathcal{D}_\varepsilon(M, C) = M$. This will meet the decryption and tag correctness conditions besides meeting the security requirements (privacy and tag consistency) we will formalize below. However, this scheme is of no use for deduplication because the client stores the entire file as the key and no storage savings are gleaned. To avoid this kind of degenerate scheme, we insist that an MLE scheme have keys that are shorter than the message. Formally, there must be a constants $c, d < 1$ such that the function that on input $\lambda \in \mathbb{N}$ returns $\max_{P,M} \Pr[|\mathcal{K}_P(M)| > d \cdot |M|^c]$ is negligible where the probability is over the

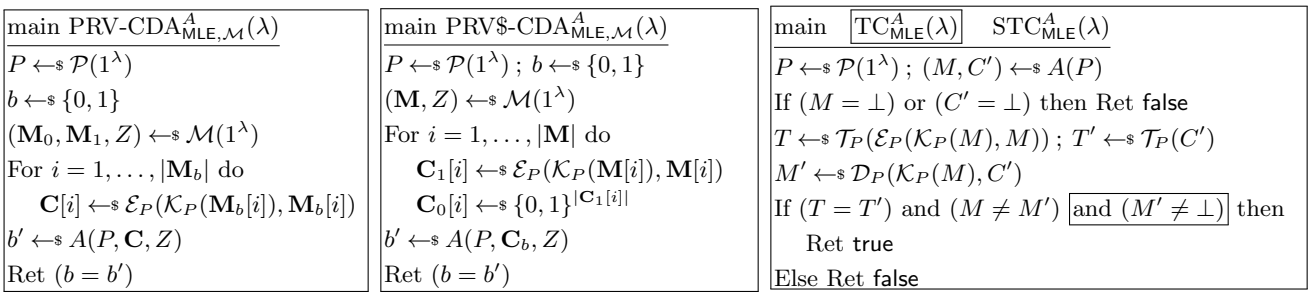


Figure 3: Games defining PRV-CDA, PRV$-CDA privacy and TC, STC tag consistency security of MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$.

choices of $\mathcal{K}$ and the maximum is over all $P \in [\mathcal{P}(1^\lambda)]$ and all $M \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$. Particular schemes we construct or analyze, however, do much better, with the key-length for most of them depending only on the security parameter.

Our formulation of search via tag comparison enables fast search: the server can use the tag to index directly into a table or perform a logarithmic-time binary search as in [5]. These requirements could be relaxed to define MLE variants where search was allowed linear time (cf. [15]) or search ability was not even provided. MLE does not appear easy to achieve even in the last case.

Privacy. As we noted in Section 1, no MLE scheme can provide privacy for predictable messages, meaning ones drawn from a space of polynomial size, in particular ruling out classical semantic security. We now formalize two notions of privacy for unpredictable messages.

A *source* is a PT algorithm $\mathcal{M}$ that on input 1^λ returns $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z)$ where $\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}$ are vectors over $\{0,1\}^*$ and $Z \in \{0,1\}^*$. Here $n \geq 1$ is a constant called the arity of the source. (We will only consider $n \in \{1, 2\}$.) We require that all the vectors have the same length $m(\lambda)$ for some function m called the number of messages of the source. We require that there is a function len, called the message length of the source, such that the string $\mathbf{M}_j[i]$ has length $\mathsf{len}(\lambda, i)$ for all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$. We require that $\mathbf{M}_j[i_1] \neq \mathbf{M}_j[i_2]$ for all distinct $i_1, i_2 \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$, meaning the entries of each vector are distinct. We refer to Z as the auxiliary information. The guessing probability $\mathbf{GP}_{\mathcal{M}}$ of source $\mathcal{M}$ is defined as the function which on input $\lambda \in \mathbb{N}$ returns $\max_{i,j} \mathbf{GP}(\mathbf{M}_j[i] \,|\, Z)$ where the probability is over $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z) \leftarrow\!\$\ \mathcal{M}(1^\lambda)$ and the maximum is over all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$. We say that $\mathcal{M}$ is *unpredictable* if $\mathbf{GP}_{\mathcal{M}}(\cdot)$ is negligible. (Meaning, messages are unpredictable given the auxiliary information. We do *not* require that the components $\mathbf{M}_j[1], \ldots, \mathbf{M}_j[m(\lambda)]$ of a vector are independent, just that each, individually, is unpredictable.) We refer to $-\log(\mathbf{GP}_{\mathcal{M}}(\cdot))$ as the min-entropy of the source. We say that $\mathcal{M}$ is MLE-valid if $\mathbf{M}_j[i] \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$ for all $\lambda \in \mathbb{N}$, all $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z) \in [\mathcal{M}(1^\lambda)]$, all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$.

In the games of Figure 3, "CDA" stands for "Chosen-Distribution Attack," referring to the distribution on messages imposed by the source $\mathcal{M}$, which in game PRV-CDA has arity 2 and in game PRV$-CDA has arity 1. The source is assumed MLE-valid. If A is an adversary we let $\mathbf{Adv}^{\mathsf{prv\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\text{PRV-CDA}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)] - 1$ and $\mathbf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\text{PRV\$-CDA}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)] - 1$. We say that MLE is PRV-CDA (resp. PRV$-CDA) secure over a class $\overline{\mathcal{M}}$ of PT, MLE-valid sources if $\mathbf{Adv}^{\mathsf{prv\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\cdot)$ (resp. $\mathbf{Adv}^{\mathsf{prv\$\text{-}cda}}_{\mathsf{MLE},\mathcal{M},A}(\cdot)$) is negligible for all PT A and all $\mathcal{M} \in \overline{\mathcal{M}}$. We say that MLE is PRV-CDA (resp. PRV$-CDA) secure if it is PRV-CDA (resp. PRV$-CDA) secure over the class of all PT, unpredictable MLE-valid sources. PRV-CDA asks for indistinguishability of encryptions of two unpredictable messages and is based on formalizations of deterministic [5,8,16] and hedged [6] PKE. PRV$-CDA is a new variant, asking for the stronger property that encryptions of unpredictable messages are indistinguishable from random strings, an adaption to this setting of the corresponding notion for symmetric encryption

from [43].

The source is not given the parameter P as input, meaning privacy is only assured for messages that do not depend on the parameter. This is analogous to the restriction that messages do not depend on the public key in D-PKE [5], and without this restriction, privacy is not possible. However, the adversary A does get the parameter.

The notions here are non-adaptive in the sense that the distribution of the next message does not depend on the previous ciphertext. In Appendix A we give corresponding adaptive definitions PRV-CDA-A and PRV$-CDA-A, and prove the relations summarized in Figure 2. The one we highlight is that the non-adaptive PRV$-CDA implies its adaptive counterpart. This is not true for PRV-CDA and makes PRV$-CDA preferable to achieve.

TAG CONSISTENCY. Consider the games of Figure 3 and let A be an adversary. Game $\mathrm{TC}_{\mathsf{MLE}}$ includes the boxed statement, while $\mathrm{STC}_{\mathsf{MLE}}$ does not. We let $\mathbf{Adv}^{\mathsf{TC}}_{\mathsf{MLE},A}(\lambda) = \Pr[\mathrm{TC}^{A}_{\mathsf{MLE}}(\lambda)]$ and $\mathbf{Adv}^{\mathsf{STC}}_{\mathsf{MLE},A}(\lambda) = \Pr[\mathrm{STC}^{A}_{\mathsf{MLE}}(\lambda)]$. We say that MLE is TC (resp. STC) secure if $\mathbf{Adv}^{\mathsf{TC}}_{\mathsf{MLE},A}(\cdot)$ (resp. $\mathbf{Adv}^{\mathsf{STC}}_{\mathsf{MLE},A}(\cdot)$) is negligible.

Tag consistency (TC) aims to provide security against duplicate faking attacks in which a legitimate message is undetectably replaced by a fake one. In such an attack we imagine the adversary A creating and uploading C'. Later, an honest client, holding M (the formalism allows A to pick M) computes $K \leftarrow\!\!\$\ \mathcal{K}_P(M)$ and uploads $C \leftarrow\!\!\$\ \mathcal{E}_P(K, M)$. The server finds that the tags of C and C' are equal and thus continues to store only C'. Later, the honest client downloads C' and decrypts under K. It expects to recover M, but in a successful duplicate-faking attack it recovers instead some message $M' \neq M$. The integrity of its data has thus been violated. TC security protects against this. Note that TC explicitly excludes an attack in which $M' = \perp$. Thus TC secure schemes may still admit erasure attacks, in which a client can detect corruption but no longer be able to recover their message. STC (strong tag consistency) aims to additionally provide security against erasure attacks. In terms of implications, STC implies TC but TC does not imply STC.

Duplicate faking attacks are not just a theoretical concern. They were first discussed in [45], yet currently deployed schemes are still vulnerable, as we'll see in the next section. Discussions with practitioners suggest that security against them is viewed in as an important requirement in practice.

Given any TC secure scheme, we can prevent all of the attacks above by having a client, upon being informed that her ciphertext is already stored, download it immediately and check that decryption yields his message. If not, she complains. This however is not optimal, being expensive and complex and leading to deduplication side-channels (cf. [32]).

If an MLE scheme is deterministic, letting the tag equal the ciphertext will result in a scheme that is STC secure. (In a strong sense, for the advantage of even a computationally unbounded adversary is 0 in either case. We omit the simple proof.) This provides a relatively easy way to ensure resistance to duplicate faking attacks, but the price paid is that the tag is as long as the ciphertext. CR-hashing the ciphertext (still for a D-MLE scheme) preserves STC, but for efficiency other, less effective options have been employed in practice, as we will see.

ROM. A RO [10] is a game procedure H that maintains a table $\mathtt{H}[\cdot,\cdot]$, initially everywhere $\perp$. Given a query x, k with $x \in \{0,1\}^*$ and $k \in \mathbb{N}$, it executes: If $\mathtt{H}[x,k] = \perp$ then $\mathtt{H}[x,k] \leftarrow\!\!\$\ \{0,1\}^k$. It then returns $\mathtt{H}[x,k]$. As this indicates, the RO can provide outputs of any desired length. We will at times omit the length when a single value is of import and that length is clear from context. In the ROM, both scheme algorithms and the adversary will have access to H. However, in the privacy definitions, we do not give the source access to H. This is to simplify our proofs. Our methods and proofs can be extended to handle sources that query the RO.

4 Practical Contributions: The Security of Fast MLE Schemes

We investigate four MLE schemes, two that correspond to in-use schemes and two new schemes. Figure 4 provides pseudocode for all four schemes.

Ingredients. The schemes are built from a one-time symmetric encryption scheme and a hash function family $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$. The former is a tuple $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ of algorithms: key generation $\mathcal{SK}$, on input 1^λ, outputs a key K of length $k(\lambda)$; deterministic encryption $\mathcal{SE}$ maps a key K and plaintext M to a ciphertext C; and deterministic decryption $\mathcal{SD}$ maps a key K and ciphertext C to a message M. We require that $\Pr[\mathcal{SD}(K, \mathcal{SE}(K, M)) = M] = 1$ for all $\lambda \in \mathbb{N}$, all $K \in [\mathcal{SK}(1^\lambda)]$, and all $M \in \{0,1\}^*$. We assume that there exists a function $\mathsf{cl}_{\mathsf{SE}}$ such that for all $\lambda \in \mathbb{N}$ and all $M \in \{0,1\}^*$ any output of $\mathcal{SE}(K, M)$ has length $\mathsf{cl}_{\mathsf{SE}}(\lambda, |M|)$. For simplicity we assume that $\mathcal{H}$ and SE are compatible: $\mathcal{H}(K_h, M)$ outputs a message of length $k(\lambda)$ for any $K_h \in [\mathcal{HK}(1^\lambda)]$.

We require schemes that provide both key recovery security and one-time real-or-random security [43]. Let game $\mathrm{KR}_{\mathsf{SE}}$, on input 1^λ, run the adversary A. The latter can query at most once to an encryption oracle Enc a message M to which the game replies with an encryption of M under a freshly chosen key K. Adversary A outputs a bit string K' and wins if $K' = K$. Advantage is defined as $\mathbf{Adv}^{\mathrm{kr}}_{\mathsf{SE},A}(\lambda) = \Pr[\mathrm{KR}^A_{\mathsf{SE}}(1^\lambda)]$ and we say that SE is KR-secure if $\mathbf{Adv}^{\mathrm{ror}}_{\mathsf{SE},A}(\cdot)$ is a negligible function for any PT A.

Let game $\mathrm{ROR}_{\mathsf{SE}}$ on input 1^λ, first choose a random bit b, and then run the adversary A. Adversary A can make multiple queries to an encryption oracle Enc, each query a plaintext $M \in \{0,1\}^*$. If $b = 1$, then Enc will first choose a random key $K \leftarrow\!\!\$\ \mathcal{SK}(1^\lambda)$ and return $C \leftarrow \mathcal{SE}(K, M)$. Note that each encryption query chooses a fresh key. If $b = 0$, then Enc will just return a random bit string of length $\mathsf{cl}_{\mathsf{SE}}(\lambda, |M|)$. To win, the adversary should guess b. Advantage is defined as $\mathbf{Adv}^{\mathrm{ror}}_{\mathsf{SE},A}(\lambda) = 2 \cdot \Pr[\mathrm{ROR}^A_{\mathsf{SE}}(1^\lambda)] - 1$ and we say that SE is ROR-secure $\mathbf{Adv}^{\mathrm{ror}}_{\mathsf{SE},A}(\cdot)$ is a negligible function for any PT A.

The four schemes. The first scheme, that we simply call convergent encryption (CE), generalizes the original scheme of DABST [22]. CE encrypts by hashing the message to generate a symmetric key K, which is then used to encrypt the message using $\mathcal{SE}$. Tags are computed by hashing the entire ciphertext.[1]

The second scheme, HCE1 (Hash-and-CE 1), is a popular variant of the CE scheme used in a number of systems [18, 19, 24, 48]. Compared to CE, HCE1 computes tags during encryption by hashing the per-message key and including the result in the ciphertext. Tag generation just extracts this embedded tag. This offloads work from the server to the client and reduces the number of cryptographic passes needed to encrypt and generate a tag from three to two.

HCE1 is vulnerable to attacks that break TC security, as first discussed in [45]. The attack is straightforward: adversary A chooses two messages $M \neq M'$, computes $C \leftarrow \mathsf{SE}(\mathcal{H}(P, M), M')$ and $T \leftarrow \mathcal{H}(P, \mathcal{H}(P, M))$, and finally outputs $(M, C \,\|\, T)$. This means an adversary, given knowledge of a user's to-be-stored message, can undetectably replace it with an arbitrary message of the adversary's choosing.

We suggest a new scheme, HCE2, that modifies HCE1 to include a mechanism that we refer to as guarded decryption. Namely, during decryption, it checks the tag embedded in the ciphertext by recomputing the tag using the just-decrypted message. If the check fails, then $\perp$ is returned. As we argue below, this provably ensures TC security (but not STC).

In practice, the important operation is deriving the ciphertext and tag from the message. This involves generating the key, followed by encryption and tag generation. CE requires three full cryptographic passes to perform encryption and tag generation, while HCE1 and HCE2 require two. This is fundamental: deterministic MLE schemes that output bits of ciphertext before processing most of the message will not achieve PRV-CDA security.

The third scheme, Randomized Convergent Encryption (RCE), therefore takes advantage of randomization to give a version of HCE2 that can generate the key, encrypt the message, and produce the tag, all together, in a single pass. RCE accomplishes this by first picking a random symmetric encryption key

[1] One could alternatively use the ciphertext itself as the tag, but this is typically not practical.

Scheme	Key generation	Encrypt	Tag generation	Decrypt
	$\mathcal{K}_P(M)$	$\mathcal{E}_P(K, M)$	$\mathcal{T}_P(C)$	$\mathcal{D}_P(K, C)$
$\mathsf{CE}[\mathsf{SE}, \mathsf{H}]$ Convergent Encryption	$K \leftarrow \mathcal{H}(P, M)$ Ret K	$C \leftarrow \mathcal{SE}(K, M)$ Ret C	Ret $\mathcal{H}(P, C)$	Ret $\mathcal{SD}(K, C)$
$\mathsf{HCE1}[\mathsf{SE}, \mathsf{H}]$ Hash and CE w/o tag check		$T \leftarrow \mathcal{H}(P, K)$ $C \leftarrow \mathcal{SE}(K, M)$ Ret $C \,\|\, T$	Parse C as $C_1 \,\|\, T$ Ret T	Parse C as $C_1 \,\|\, T$ $M \leftarrow \mathcal{SD}(K, C)$ Ret M
$\mathsf{HCE2}[\mathsf{SE}, \mathsf{H}]$ Hash and CE w/ tag check				Parse C as $C_1 \,\|\, T$ $M \leftarrow \mathcal{SD}(K, C)$ $T' \leftarrow \mathcal{H}(P, \mathcal{H}(P, M))$ If $T' \neq T$ then $\perp$ Ret M
$\mathsf{RCE}[\mathsf{SE}, \mathsf{H}]$ Randomized Convergent Encryption		$L \leftarrow\!\!\$\ \{0,1\}^{k(\lambda)}$ $T \leftarrow \mathcal{H}(P, K)$ $C_1 \leftarrow \mathcal{SE}(L, M)$ $C_2 \leftarrow L \oplus K$ Ret $C_1 \,\|\, C_2 \,\|\, T$	Parse C as $C_1 \,\|\, C_2 \,\|\, T$ Ret T	Parse C as C_1, C_2, T $L \leftarrow C_2 \oplus K$ $M \leftarrow \mathcal{SD}(L, C_1)$ $T' \leftarrow \mathcal{H}(P, \mathcal{H}(P, M))$ If $T' \neq T$ then Ret $\perp$ Ret M

Figure 4: MLE schemes built using symmetric encryption scheme $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ and hash function family $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$. All schemes inherit their message space from SE (typically $\{0,1\}^*$), use as parameter generation $\mathcal{HK}$, and share a common key generation algorithm. HCE1 and HCE2 additionally use the same encryption and tag generation algorithms.

L and then encrypting the message with L, and deriving the MLE key K in a single pass. Finally it encrypts L using K as a one-time pad, and derives the tag from K. Like HCE2 it uses guarded decryption.

It is easy to verify decryption correctness. Tag correctness follows as the tags are all deterministic.

Privacy. We prove the PRV$-CDA security of the four schemes when modeling H as a RO. In Appendix B we give a concrete-security statement together with proof that covers all four schemes. The following theorem ends up an easy corollary.

Theorem 4.1. *Let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme with key length* $k(\cdot)$ *and let* H *be a RO. Then if* SE *is both* KR*-secure and* ROR*-secure, the scheme* XXX[SE, H] *for* $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE1}, \mathsf{HCE2}, \mathsf{RCE}\}$ *is* PRV$-CDA*-secure.* □

Tag consistency. We turn to security in the sense of tag consistency. As discussed in Section 3, any deterministic scheme is STC-secure when tags are CR-hashes of the ciphertext. So too with CE. For HCE2 and RCE, a straightforward reduction establishes the following theorem. We omit the details.

Theorem 4.2. *Let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme and* $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ *be a hash function family. If* H *is* CR*-secure then* HCE2[SE, H] *and* RCE[SE, H] *are* TC*-secure.* □

HCE2 and RCE are not STC-secure, by the same attack as used against the TC security of HCE1. (The tag check makes it so that decryption outputs $M' = \perp$.) One could achieve STC security using non-interactive zero-knowledge proofs [13], but this would obviate the speedups offered by the schemes compared to CE. Finding fast, STC-secure schemes with $\mathcal{O}(1)$ tag generation is an interesting open problem surfaced by our definitions and results above.

Discussion. The above schemes use a hash function family H. In practice, we might use SHA-256 or SHA-3, and key them appropriately by choosing a uniform bit string to prepend to messages. Note that SHA-256 is not indifferentiable from a RO [38]. We may also want to prove security for sources that have access to the RO, a setting for which indifferentiability-based composition does not apply (cf., [42]). In these cases one must perform direct proofs of security that account for the iterative structure of the hash function. In Appendix C we explore various instantiations of the MLE schemes, including ones that

implement the hash function by way of a block cipher. This specifically yields MLE schemes entirely built from AES, which provides efficiency benefits due to widespread hardware support for AES (i.e., AES-NI). We also report on performance there.

5 Theoretical Contributions: Constructions without ROs

We present a paradigm for constructing MLE schemes that we call Extract-Hash-Check. In particular it yields standard model constructions of MLE-schemes from D-PKE schemes and CI-H hash functions. We then present an MLE scheme based on a weaker assumption, namely any ROR symmetric encryption scheme, for particular classes of sources, based on a method we call Sample-Extract-Encrypt. It also uses an extractor.

5.1 Extract-Hash-Check

Overview. It is natural to aim to build MLE from a D-PKE scheme or a CI-H function because the latter primitives already provide privacy on unpredictable messages. However, in attempting to build MLE from these primitives, several problems arise. One is that neither of the base primitives derives the decryption key from the message. Indeed, in both, keys must be generated upfront and independently of the data. A related problem is that it is not clear how an MLE scheme might decrypt. CI-H functions are not required to be efficiently invertible. D-PKE does provide decryption, but it requires the secret key, and it is not clear how this can yield message-based decryption.

Our solution will in fact not use the decryptability of the D-PKE scheme, but rather view the latter as providing a CI-H function keyed by the public key. We apply an extractor (its seed S will be in the parameters of the MLE scheme) to the message M to get the MLE key K. Given S, M, this operation is deterministic. The scheme encrypts the message bit by bit, creating from $M = M[1] \ldots M[|M|]$ the ciphertext $C = C[1] \ldots C[|M|]$ in which $C[i]$ is a hash of $K \| \langle i \rangle \| M[i]$. (The key for the hash function is also in the parameters.) To decrypt $C[i]$ given K, hash both $K \| \langle i \rangle \| 1$ and $K \| \langle i \rangle \| 0$ and see which equals $C[i]$. (This is the "check" part.) The proof of privacy relies on the fact that each input to each application of the hash function will have a negligible guessing probability even given the parameters. The reduction will take an MLE source and build a source for the hash function that itself computes K and produces the inputs to the hash function. We now proceed to the details.

Ingredients. The first tool we need is a family of hash functions $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$. We define privacy in the same manner as for MLE. Namely, game PRV-CDA$_{\mathsf{H}}$ is the same as PRV-CDA$_{\mathsf{MLE}}$ of Figure 3 except that it uses $\mathcal{HK}$ and $\mathcal{H}$ instead of $\mathcal{P}$ and $\mathcal{E}$. Likewise for PRV\$-CDA$_{\mathsf{H}}$. If A is an adversary we let $\mathbf{Adv}^{\text{prv-cda}}_{\mathsf{H},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\text{PRV-CDA}^{A}_{\mathsf{H},\mathcal{M}}(\lambda)] - 1$ and $\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\text{PRV\$-CDA}^{A}_{\mathsf{H},\mathcal{M}}(\lambda)] - 1$. We say that H is PRV-CDA (resp. PRV\$-CDA) secure if $\mathbf{Adv}^{\text{prv-cda}}_{\mathsf{H},\mathcal{M},A}(\cdot)$ (resp. $\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},\mathcal{M},A}(\cdot)$) is negligible for all PT $\mathcal{M}, A$ such that $\mathcal{M}$ is unpredictable. The PRV-CDA formulation follows [5, 8, 29] while the PRV\$-CDA formulation follows [29].

The second tool we need is a family of extractors. This is a family $\text{Ext} = \{\text{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ where $\text{Ext}_\lambda \colon \{0,1\}^{s(\lambda)} \times \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{k(\lambda)}$ for each $\lambda \in \mathbb{N}$. We refer to s, ℓ, k as the seed, input and output lengths respectively. We require that the map $1^\lambda, S, X \mapsto \text{Ext}_\lambda(S, X)$ be PT computable. For all $\lambda \in \mathbb{N}$ and all random variables $(X, Z), S, K$ we require that $\mathsf{SD}((S, \text{Ext}_\lambda(S, X), Z); (S, K, Z)) \leq \sqrt{2^{k(\lambda)} \cdot \mathbf{GP}(X \mid Z)}$ under the following conditions: $(X, Z), S, K$ are independent, S is uniformly distributed over $\{0,1\}^{s(\lambda)}$, and K is uniformly distributed over $\{0,1\}^{k(\lambda)}$. A construction with this guarantee may be obtained via the (average-case version of the) Leftover Hash Lemma (LHL) [20, 33].

Extract-Hash-Check construction. Let $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ be a family of hash functions. Let $\text{Ext} = \{\text{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of extractors with seed length s, input length ℓ and output length k. Our construction associates to them the MLE scheme $\text{XHC}[\mathsf{H}, \text{Ext}] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ whose constituent algorithms are defined in Figure 5. The message space of this scheme is defined by $\text{MsgSp}(\lambda) = \{0,1\}^{\ell(\lambda)}$ for all $\lambda \in \mathbb{N}$.

$\underline{\mathcal{P}(1^\lambda)}$
$S \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$
$K_h \leftarrow\!\!\$\ \mathcal{HK}(1^\lambda)$
Ret $S\|K_h$

$\underline{\mathcal{K}_{S\|K_h}(M)}$
$K \leftarrow \mathrm{Ext}_\lambda(S, M)$
Ret K

$\underline{\mathcal{E}_{S\|K_h}(K, M)}$
For $i = 1$ to $|M|$ do
  $C[i] \leftarrow \mathcal{H}(K_h, K\|\langle i\rangle\|M[i])$
Ret C

$\underline{\mathcal{T}_{S\|K_h}(C)}$
Ret C

$\underline{\mathcal{D}_{S\|K_h}(K, C)}$
For $i = 1$ to $|C|$ do
  If $C[i] = \mathcal{H}(K_h, K\|\langle i\rangle\|1)$ then $M[i] \leftarrow 1$
  Else if $C[i] = \mathcal{H}(K_h, K\|\langle i\rangle\|0)$ then $M[i] \leftarrow 0$
  Else Ret $\perp$
Ret M

Figure 5: MLE scheme XHC[H, Ext] associated to hash family $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ and extractor family $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$.

We let $\langle i\rangle$ denote the encoding of $i \in \mathbb{N}$ as a λ-bit string. (We are assuming $\ell(\lambda) \le 2^\lambda$ for all $\lambda \in \mathbb{N}$.) The ciphertext C is a $\ell(\lambda)$-vector over $\{0,1\}^*$.

This MLE scheme is deterministic, hence provides perfect tag correctness. It satisfies decryption correctness as long as H is injective. (A weaker, computational decryption correctness condition is met if H is not injective but is collision-resistant.) As a consequence of being deterministic and using ciphertexts for tags, it also has unconditional consistency, namely perfect STC security. (We are, for simplicity, using the ciphertext as the tag. For greater efficiency one could CR-hash it. STC would still hold, but now computationally.) The main task is to prove privacy, which is done by the following, whose proof is in Appendix D. Here, when $\mathsf{MLE} = \mathrm{XHC}[\mathsf{H}, \mathrm{Ext}]$, we denote by $\overline{\mathcal{M}}_{\mathsf{MLE}}$ the class of all PT, MLE-valid sources $\mathcal{M}$ such that $2^{k(\cdot)} \cdot \mathbf{GP}_{\mathcal{M}}(\cdot)$ is negligible, where k is the output length of Ext.

Theorem 5.1. *Let* $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ *be a family of hash functions. Let* $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a family of extractors. Let* $\mathsf{MLE} = \mathrm{XHC}[\mathsf{H}, \mathrm{Ext}]$ *be the MLE scheme associated to them via our Extract-Hash-Check construction. Then (1) If* H *is* PRV-CDA*-secure then* MLE *is* PRV-CDA*-secure over* $\overline{\mathcal{M}}_{\mathsf{MLE}}$ *, and (2) If* H *is* PRV$-CDA*-secure then* MLE *is* PRV$-CDA*-secure over* $\overline{\mathcal{M}}_{\mathsf{MLE}}$ *.*

We can directly instantiate H by a CI-H function as defined in [29]. If $\mathsf{DPKE} = (\mathcal{DK}, \mathcal{DE}, \mathcal{DD})$ is a D-PKE scheme, we obtain a hash family $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ as follows. Let $\mathcal{HK}(1^\lambda)$ run $\mathcal{DK}(1^\lambda)$ to get (pk, sk) and return pk as the hash key. Let $\mathcal{H}(pk, M) = \mathcal{DE}(pk, M)$. (We note that we must assume a trusted setup which executes $\mathcal{HK}$ as shown and discards sk, for if the server knows sk it can break the scheme. The parameters must be generated by a third party or via a secure computation protocol so that the server does not learn sk.) If DPKE meets the PRIV-security condition of [5, 8] appropriately extended to handle auxiliary inputs as above, then $\mathcal{H}$ will be PRV-CDA-secure. Note that this hash family is injective. Finally we note that the construction can be adapted to turn an efficiently-searchable PKE scheme [5] into an MLE scheme, but since the former may be randomized, the latter may be as well. These constructions account for the schemes called XtCIH, XtDPKE and XtESPKE in the table of Figure 1 and the corresponding implication arrows in the picture.

D-MLE IMPLIES CI-H. Finally we justify the claim of Figure 1 that deterministic MLE implies CI-H. (Combined with the above, this makes the primitives equivalent.) Given a deterministic MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ we define the family of hash functions $\mathsf{H} = (\mathcal{P}, \mathcal{H})$ as follows. Algorithm $\mathcal{H}$, given key P and message M, lets $K \leftarrow \mathcal{K}_P(M)$ and $C \leftarrow \mathcal{E}_P(K, M)$, and returns C. (Both $\mathcal{K}$ and $\mathcal{E}$ are deterministic by assumption.) It is easy to see that if MLE is PRV$-CDA-secure then so is H.

5.2 Sample-Extract-Encrypt

OVERVIEW. We now give a construction of an MLE scheme that relies only on a standard and weak assumption, namely a one-time symmetric encryption scheme as defined in Section 4, which can be built from any one-way function. The tradeoff is that the scheme only works for a limited class of sources.

Stepping back, if we are to consider special sources, the obvious starting point is uniform and independent messages. Achieving MLE here is easy because we can use part of the message as the key to

$\underline{\mathsf{Proj}(M,T)}$
$M_1 \leftarrow \varepsilon$; $i_1 \leftarrow 0$
For $i = 1$ to $|T|$ do
  If $T[i] = 1$ then $i_1 \leftarrow i_1 + 1$; $M_1[i_1] \leftarrow M[i]$
Return M_1

$\underline{\mathsf{Merge}(M_1, M_2, T)}$
$M \leftarrow \varepsilon$; $i_1 \leftarrow 0$; $i_2 \leftarrow 0$
For $i = 1$ to $|T|$
  If $T[i] = 1$ then $i_1 \leftarrow i_1 + 1$; $M[i] \leftarrow M_1[i_1]$
  Else $i_2 \leftarrow i_2 + 1$; $M[i] \leftarrow M_2[i_2]$
Return M

$\underline{\mathcal{P}(1^\lambda)}$
$S \leftarrow\!\!\$\ \{0,1\}^{s(\lambda)}$; $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$
Ret $S \| U$

$\underline{\mathcal{K}_{S\|U}(M)}$
$K \leftarrow \mathsf{Proj}(M, U)$; Ret K

$\underline{\mathcal{E}_{S\|U}(K,M)}$
$L \leftarrow \mathrm{Ext}_\lambda(S, \mathsf{pad}_\lambda(K))$
$M_2 \leftarrow \mathsf{Proj}(M, \overline{U})$
$C \leftarrow\!\!\$\ \mathcal{SE}(L, M_2)$
Ret C

$\underline{\mathcal{D}_{S\|U}(K,C)}$
$L \leftarrow \mathrm{Ext}_\lambda(S, \mathsf{pad}_\lambda(K))$
$M_2 \leftarrow \mathcal{SD}(L, C)$
$M \leftarrow \mathsf{Merge}(K, M_2, U)$
Ret M

Figure 6: **Top:** The Proj and Merge algorithms. **Bottom:** MLE scheme $\mathsf{SXE}[\mathsf{SE}, \mathrm{Ext}, p]$ associated to symmetric encryption scheme $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$, extractor family $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$ and $p\colon\ \mathbb{N} \to [0,1]$. The tag algorithm $\mathcal{T}_{S\|U}(C)$ simply returns C.

encrypt the other part. The next obvious target is block sources, where each message is assumed to have negligible guessing probability given the previous ones. D-PKE for such sources was achieved in [14]. We might hope, via the above XHC construction, to thus automatically obtain MLE for the same sources, but XHC does not preserve the block source restriction because the inputs to the hash function for different bits of the same message are highly correlated.

Our Sample-Extract-Encrypt (SXE) construction builds an MLE scheme for certain classes of block sources where a random subset of the bits of each message remains unpredictable even given the rest of the bits and previous messages. For example, if a message has some subset of uniform bits embedded within it. The scheme then uses a random subset of the message bits as a key, applies an extractor, and then symmetrically encrypts the rest of the message.

Preliminaries. First, some notation. If $T \in \{0,1\}^*$, we let $\mathsf{hw}(T)$ denote the Hamming weight of T and $\overline{T}$ the bitwise complement of T. For $T \in \{0,1\}^{|M|}$, function $\mathsf{Proj}(M,T)$ of Figure 6 returns the $\mathsf{hw}(T)$-length string formed by selecting from M the coordinates i in which $T[i] = 1$. If $q \in [0,1]$ we let $U \leftarrow_q \{0,1\}^n$ mean that we let $U[i] = 1$ with probability q and 0 with probability $1-q$, independently for each $i \in [n]$. Note that the expected Hamming weight of U is then qn. Looking ahead, we will be interested in sources for which one can "gather" sufficient min-entropy by randomly taking a subset of message bits. The fraction q controls the bias with which we select any particular bit.

In this section, all sources have arity 1. Let $\mathcal{M}$ be a source with number of messages m and message length len such that $\mathsf{len}(\lambda, i) = n(\lambda)$ for all $\lambda, i \in \mathbb{N}$, meaning all messages $\mathbf{M}[1], \ldots, \mathbf{M}[m(\lambda)] \in [\mathcal{M}(1^\lambda)]$ have the same length $n(\lambda)$. For $p\colon\ \mathbb{N} \to [0,1]$ and $\lambda \in \mathbb{N}$ we let

$$\mathbf{GP}_{\mathcal{M},p}(\lambda) = \max_i \mathbf{GP}(\mathsf{Proj}(\mathbf{M}[i], U) \mid (\mathbf{M}[1], \ldots, \mathbf{M}[i-1], \mathsf{Proj}(\mathbf{M}[i], \overline{U}), Z))$$

where the probability is over $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$ and $(\mathbf{M}, Z) \leftarrow \mathcal{M}(1^\lambda)$ and the maximum is over all $i \in [m(\lambda)]$. In other words, picking U at random from our $p(\lambda)$-biased distribution, we are measuring the probability of guessing the projection of the i-th message onto U, given the other bits of the message as well as the previous messages. We say that $\mathcal{M}$ is p-unpredictable if $\mathbf{GP}_{\mathcal{M},p}(\cdot)$ is negligible. Note that a blocksource is a source that is 1-unpredictable. Considering smaller values of p thus relaxes the usual blocksource requirement.

Some natural sources are, or can be shown to be, p-unpredictable for small p. One obvious example is uniform messages. Short of that is any source that is sufficiently dense, meaning that a large enough fraction of the message bits have high min-entropy conditioned on all other bits. A concrete example would be sources that embed uniform bits in arbitrary locations within a message. For such a source, one can use a Chernoff bound to show that it is a p-unpredictable source for a reasonable value of p related to the density of the message.

THE SAMPLE-EXTRACT-ENCRYPT CONSTRUCTION. Let $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ be a (deterministic) one-time symmetric encryption scheme with key length $k(\cdot)$. Let $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of extractors with seed length s, output length k and input length ℓ. (We've assumed that the output length of Ext is equal to the key length of SE.) We let $n(\cdot) = \ell(\cdot) - 1$. Let p: $\mathbb{N} \to [0,1]$. Our construction associates to them the MLE scheme $\mathsf{SXE}[\mathsf{SE}, \mathrm{Ext}, p] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ whose constituent algorithms are defined in Figure 6. Here we let $\mathsf{pad}_\lambda(K) = K\|1\|0^{n(\lambda)-|K|}$ so that outputs of pad_λ are valid inputs for $\mathrm{Ext}_\lambda(S,\cdot)$. The message space of this scheme is defined by $\mathrm{MsgSp}(\lambda) = \{0,1\}^{n(\lambda)}$ for all $\lambda \in \mathbb{N}$.

This MLE scheme is deterministic, hence provides perfect tag correctness. It satisfies decryption correctness due to the corrections of SE. Note that the scheme is not strictly non-trivial, since it could be that $\mathsf{hw}(U) = n(\lambda)$. However, it is non-trivial in expectation whenever $p(\lambda) < 1$, since $\mathbf{E}[\mathsf{hw}(U)] = p(\lambda)\cdot n(\lambda)$. As a consequence of being deterministic, and using ciphertext as the tag, it also has perfect STC security. (We are, for simplicity, using the ciphertext as the tag. For greater efficiency one could CR-hash it. STC would still hold, but now computationally.) The following theorem establishes privacy. Here, when $\mathsf{MLE} = \mathsf{SXE}[\mathsf{SE}, \mathrm{Ext}, p]$, we denote by $\overline{\mathcal{M}}_{\mathsf{MLE}}$ the class of all PT, MLE-valid sources $\mathcal{M}$ such that $2^{k(\cdot)} \cdot \mathbf{GP}_{\mathcal{M},p}(\cdot)$ is negligible, where k is the output length of Ext. The proof is in Appendix E.

Theorem 5.2. *Let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme providing* ROR *security. Let* $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a family of extractors Let* p: $\mathbb{N} \to [0,1]$. *Let* $\mathsf{MLE} = \mathsf{SXE}[\mathsf{SE}, \mathrm{Ext}, p]$ *be the MLE scheme associated to them via our Sample-Extract-Encrypt construction. Then* MLE *is* $\mathsf{PRV\$\text{-}CDA}$*-secure over* $\overline{\mathcal{M}}_{\mathsf{MLE},p}$.

The key in this scheme has expected length $p(\cdot)\cdot n(\cdot)$. If we increase p, we get security for a larger class of sources at the cost of a larger key length, so the construction can be seen as trading key size for security.

Acknowledgments

The authors thank James Lentini for pointing out the STC attack against EMK; Ananth Raghunathan for early discussions about convergent encryption; and the TahoeFS developers for informative discussions regarding duplicate faking attacks in TahoeFS.

References

[1] A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. *ACM SIGOPS Operating Systems Review*, 36(SI):1–14, 2002. (Cited on page 3.)

[2] P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted de-duplication. In *Proc. of USENIX LISA*, 2010. (Cited on page 3.)

[3] Z. Bar-Yossef, O. Reingold, R. Shaltiel, and L. Trevisan. Streaming computation of combinatorial objects. In *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on*, pages 133–142. IEEE, 2002. (Cited on page 3, 6.)

[4] C. Batten, K. Barr, A. Saraf, and S. Trepetin. pStore: A secure peer-to-peer backup system. *Unpublished report, MIT Laboratory for Computer Science*, pages 130–139, 2001. (Cited on page 3.)

[5] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Aug. 2007. (Cited on page 3, 4, 5, 6, 9, 10, 13, 14, 21.)

[6] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249. Springer, Dec. 2009. (Cited on page 4, 7, 9, 21.)

[7] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997. (Cited on page 4.)

[8] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 360–378. Springer, Aug. 2008. (Cited on page 4, 6, 9, 13, 14.)

[9] M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In Y. Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 216–233. Springer, Aug. 1994. (Cited on page 25.)

[10] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. (Cited on page 3, 10.)

[11] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 7, 22.)

[12] Bitcasa. Bitcasa, a file-storage and sharing service. http://www.bitcasa.com/. (Cited on page 3.)

[13] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988. (Cited on page 12.)

[14] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, Aug. 2008. (Cited on page 6, 15.)

[15] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, May 2004. (Cited on page 3, 9.)

[16] Z. Brakerski and G. Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 543–560. Springer, Aug. 2011. (Cited on page 4, 6, 9.)

[17] Ciphertite. Ciphertite data backup. http://www.ciphertite.com/. (Cited on page 3.)

[18] J. Cooley, C. Taylor, and A. Peacock. ABS: the apportioned backup system. *MIT Laboratory for Computer Science*, 2004. (Cited on page 3, 11.)

[19] L. P. Cox, C. D. Murray, and B. D. Noble. Pastiche: making backup cheap and easy. *SIGOPS Oper. Syst. Rev.*, 36:285–298, Dec. 2002. (Cited on page 3, 11.)

[20] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008. (Cited on page 13.)

[21] Y. Dodis, T. Ristenpart, and T. Shrimpton. Salvaging Merkle-Damgård for practical applications. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 371–388. Springer, Apr. 2009. (Cited on page 25.)

[22] J. Douceur, A. Adya, W. Bolosky, P. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on*, pages 617–624. IEEE, 2002. (Cited on page 3, 5, 11.)

[23] Dropbox. Dropbox, a file-storage and sharing service. `http://www.dropbox.com/`. (Cited on page 3.)

[24] Flud. The flud backup system. `http://flud.org/wiki/Architecture`. (Cited on page 3, 11.)

[25] B. Fuller, A. O'Neill, and L. Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 582–599. Springer, Mar. 2012. (Cited on page 6.)

[26] GNUnet. GNU's framework for secure peer-to-peer networking. `https://gnunet.org/`. (Cited on page 3.)

[27] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on page 4.)

[28] Google. Google drive, a file-storage and sharing service. `http://drive.google.com/`. (Cited on page 3.)

[29] V. Goyal, A. O'Neill, and V. Rao. Correlated-input secure hash functions. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 182–200. Springer, Mar. 2011. (Cited on page 3, 5, 6, 13, 14.)

[30] S. Gueron. Advanced encryption standard (AES) instructions set. *Intel Corporation*, 25, 2008. (Cited on page 25.)

[31] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In Y. Chen, G. Danezis, and V. Shmatikov, editors, *ACM CCS 11*, pages 491–500. ACM Press, Oct. 2011. (Cited on page 7.)

[32] D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *Security & Privacy, IEEE*, 8(6):40–47, 2010. (Cited on page 7, 10.)

[33] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (Cited on page 13.)

[34] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *Journal of Cryptology*, 14(1):17–35, 2001. (Cited on page 25.)

[35] M. Killijian, L. Courtès, D. Powell, et al. A survey of cooperative backup mechanisms, 2006. (Cited on page 3.)

[36] C.-J. Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 257–271. Springer, Aug. 2002. (Cited on page 3, 6.)

[37] L. Marques and C. Costa. Secure deduplication on mobile devices. In *Proceedings of the 2011 Workshop on Open Source and Design of Communication*, pages 19–26. ACM, 2011. (Cited on page 3.)

[38] U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Feb. 2004. (Cited on page 12.)

[39] I. Mironov, O. Pandey, O. Reingold, and G. Segev. Incremental deterministic public-key encryption. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 628–644. Springer, Apr. 2012. (Cited on page 7.)

[40] N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996. (Cited on page 6.)

[41] A. Rahumed, H. Chen, Y. Tang, P. Lee, and J. Lui. A secure cloud backup system with assured deletion and version control. In *Parallel Processing Workshops (ICPPW), 2011 40th International Conference on*, pages 160–167. IEEE, 2011. (Cited on page 3.)

[42] T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, May 2011. (Cited on page 7, 12, 25.)

[43] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01*, pages 196–205. ACM Press, Nov. 2001. (Cited on page 4, 10, 11.)

[44] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000. (Cited on page 3.)

[45] M. Storer, K. Greenan, D. Long, and E. Miller. Secure data deduplication. In *Proceedings of the 4th ACM international workshop on Storage security and survivability*, pages 1–10. ACM, 2008. (Cited on page 3, 4, 10, 11.)

[46] S. P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 61–77. Springer, Aug. 2003. (Cited on page 3, 6.)

[47] D. Wichs. Barriers in cryptography with weak, correlated and leaky sources. Cryptology ePrint Archive, Report 2012/459, 2012. `http://eprint.iacr.org/`. (Cited on page 6.)

[48] Z. Wilcox-O'Hearn and B. Warner. Tahoe: The least-authority filesystem. In *Proceedings of the 4th ACM international workshop on Storage security and survivability*, pages 21–26. ACM, 2008. (Cited on page 3, 5, 11.)

A Relations Between MLE Privacy Notions

We first define the adaptive versions of PRV-CDA and PRV$-CDA, and then go on to show the relations between PRV-CDA, PRV$-CDA and their adaptive versions, as described in Figure 2. We first generalize sources to take inputs.

A *source with input* is a PT algorithm $\mathcal{M}$ that on input 1^λ and a string d returns $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z)$ where $\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}$ are vectors over $\{0,1\}^*$ and $Z \in \{0,1\}^*$. As before $n \geq 1$ is called the arity of the source. We require that

- all the vectors have the same length $m(\lambda)$, for some function m called the number of messages;
- there is a function len, called the message length of the source, such that the string $\mathbf{M}_j[i]$ has length $\mathsf{len}(\lambda, i)$ for all $i \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$; and
- $\mathbf{M}_j[i_1] \neq \mathbf{M}_j[i_2]$ for all distinct $i_1, i_2 \in [m(\lambda)]$ and all $j \in \{0, \ldots, n-1\}$, meaning the entries of each vector are distinct.

These requirements are as before for sources (without inputs). As before, we refer to Z as the auxiliary information.

The guessing probability $\mathbf{GP}_{\mathcal{M}}$ of source with input $\mathcal{M}$ is defined as the function which on input $\lambda \in \mathbb{N}$ returns $\max_{i,j,d} \mathbf{GP}(\mathbf{M}_j[i] \,|\, Z)$ where the probability is over $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda, d)$ and the maximum is over all $i \in [m(\lambda)]$, all $j \in \{0, \ldots, n-1\}$ and all $d \in \{0,1\}^*$. (The domain for the latter being infinite, the max here is interpreted as a sup and it is an implicit assumption that for a source-with-input to be valid, this sup must exist.) As compared to a source (without input) the important point here is that we maximize over d as well, so only the coins underlying the execution of $\mathcal{M}$ can contribute to the guessing probability. We refer to $-\log(\mathbf{GP}_{\mathcal{M}}(\cdot))$ as the min-entropy of $\mathcal{M}$. We say that $\mathcal{M}$ is *unpredictable* if $\mathbf{GP}_{\mathcal{M}}(\cdot)$ is negligible. We say that $\mathcal{M}$ is MLE-valid if $\mathbf{M}_j[i] \in \mathrm{MsgSp}_{\mathsf{MLE}}(\lambda)$ for all $\lambda \in \mathbb{N}$, all $(\mathbf{M}_0, \ldots, \mathbf{M}_{n-1}, Z) \in [\mathcal{M}(1^\lambda, d)]$, all $i \in [m(\lambda)]$, all $j \in \{0, \ldots, n-1\}$ and all $d \in \{0,1\}^*$.

Adaptive privacy notions. Let $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be an MLE scheme and $\mathcal{M}$ be MLE-valid. The adaptive chosen distribution attack games PRV-CDA-$\mathrm{A}_{\mathsf{MLE},\mathcal{M}}$ and PRV$-CDA-$\mathrm{A}_{\mathsf{MLE},\mathcal{M}}$ for MLE are detailed in Figure 7. For the former $\mathcal{M}$ will have arity one and for the latter, arity two. In these games, the adversary A can make multiple, adaptive queries to its Enc oracle, specifying each time an input d for $\mathcal{M}$. It can then ask for the parameter with a reveal query, after which no further Enc queries are allowed. We define advantage as $\mathbf{Adv}^{\text{prv-cda-a}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\text{PRV-CDA-A}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)] - 1$ and $\mathbf{Adv}^{\text{prv\$-cda-a}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) = 2 \cdot \Pr[\text{PRV\$-CDA-A}^A_{\mathsf{MLE},\mathcal{M}}(\lambda)] - 1$. We say that MLE is PRV-CDA-A (resp. PRV$-CDA-A) secure over a class $\overline{\mathcal{M}}$ of PT, MLE-valid sources if $\mathbf{Adv}^{\mathsf{prv\text{-}cdaa}}_{\mathsf{MLE},\mathcal{M},A}(\cdot)$ (resp. $\mathbf{Adv}^{\mathsf{prv\$\text{-}cdaa}}_{\mathsf{MLE},\mathcal{M},A}(\cdot)$) is negligible for all PT A and all $\mathcal{M} \in \overline{\mathcal{M}}$. We say that MLE is PRV-CDA-A (resp. PRV$-CDA-A) secure if it is PRV-CDA-A (resp. PRV$-CDA-A) secure over the class of all PT, unpredictable MLE-valid sources.

Relations between the notions. We have introduced four notions: PRV-CDA, PRV$-CDA, PRV-CDA-A and PRV$-CDA-A. Figure 2 states the relations between the four notions. The two trivial implications are that security in the adaptive sense implies security in the corresponding non-adaptive sense: PRV-CDA-A $\Rightarrow$ PRV-CDA and PRV$-CDA-A $\Rightarrow$ PRV$-CDA. (The double-arrow notation meaning security in the left sense implies security in the right.) We now show through a series of propositions the other implications and separations. Most are simple; we start with the most interesting and useful, that PRV$-CDA $\Rightarrow$ PRV$-CDA-A. In the following, let $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$ be an MLE scheme.

Proposition A.1. *If* MLE *is* PRV$-CDA *secure, then it is* PRV$-CDA-A *secure.*

Proof. We prove the above proposition by showing that for every adversary A and source $\mathcal{M}$, there exists another adversary B and source $\mathcal{M}'$ such that for any $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\text{prv\$-cda-a}}_{\mathsf{MLE},\mathcal{M},A}(\lambda) \leq q_e(\lambda) \cdot \mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{MLE},\mathcal{M}',B}(\lambda) \ ,$$

where function q_e is a bound on the number of encryption queries made by A. Moreover, $\mathcal{M}'$ has the same guessing probability as $\mathcal{M}$, and $\mathbf{T}_{\mathcal{M}'} = \mathcal{O}(\mathbf{T}_{\mathcal{M}})$, and $\mathbf{T}_B = \mathcal{O}(\mathbf{T}_A)$.

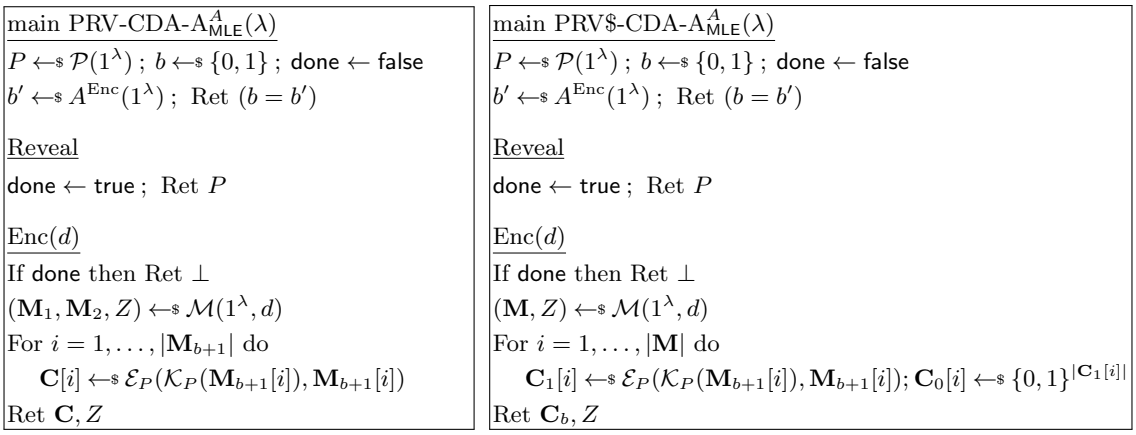


Figure 7: The PRV-CDA-A and PRV$-CDA-A games.

We use a hybrid argument. Consider game H with behavior intermediate between PRV$-CDA-A with $b = 0$ and with $b = 1$. In H, a random $g \leftarrow\!\$\ [q_e(\lambda)]$ and $b \leftarrow\!\$\ \{0,1\}$ are chosen. Up to the g-th encryption query, the adversary gets random bits as replies. The g-th query gets real ciphertexts or random bits depending on b being 1 or 0. Further queries get real ciphertexts as replies. From a standard hybridization argument, it follows that

$$\Pr[H^A(\lambda)] \geq \frac{1}{q_e(\lambda)} \mathbf{Adv}^{\mathsf{prv\$\text{-}cdaa}}_{\mathsf{MLE},\mathcal{M},A}(\lambda).$$

Consider adversary B which simulates A on game H as follows. Let $\mathcal{M}'$ be the algorithm that chooses random coins R and $g \in q_e(\lambda)$ and simulates A up to the g-th encryption query by using bits from R to supply for the randomness required by A. When A makes a query to Enc, then B runs $\mathcal{M}$ with fresh random coins from R to get $\mathbf{M}, Z$ and replies to A with random bits (from R) for ciphertexts, along with Z. When A makes its g-th encryption query d_g, then $\mathcal{M}'$ evaluates $\mathcal{M}(d)$ on fresh random coins — not from R — and outputs the resulting vector of plaintexts $\mathbf{M}$ and side information Z, R, g.

Now, consider the PRV$-CDA game with $\mathcal{M}'$ as the source. Importantly, the side information output by $\mathcal{M}'$, which includes the random bits R, does not reduce the conditional entropy of the output $\mathbf{M}$ of $\mathcal{M}(d_g)$ as the coins used to run $\mathcal{M}(d_g)$ to get $\mathbf{M}$ were picked at random, not using R. The PRV$-CDA game then invokes adversary $B(P, \mathbf{C}, Z, R, g)$, where $\mathbf{C}$ is either an encryption of $\mathbf{M}$ or a vector of random bits, depending on the choice of the bit b in the CDR game. This bit b actually corresponds to the bit in the H game as well. Adversary B then continues to run A with the same random coins R. A continues to make Enc queries which are now handled by B which replies to them since it knows the parameter P. When A makes a reveal query and B releases the parameter. Finally A finishes execution and outputs a bit b which is echoed by B. We have

$$\Pr[H^A(\lambda)] = \Pr[\text{PRV\$-CDA}^B_{\mathsf{MLE},\mathcal{M}'}(\lambda)].$$

Moreover, the sum of the running times of $\mathcal{M}'$ and B are both proportional to the running time of A, which includes the running time of $\mathcal{M}$ on the inputs provided by A. This proves the proposition. □

Security in the PRV$-CDA sense also implies security in the PRV-CDA sense, with sources of comparable entropy.

Proposition A.2. *If* MLE *is* PRV$-CDA *secure, then it is* PRV-CDA *secure.*

Proof. We can prove this proposition by showing that for every adversary A, for every source $\mathcal{M}$, there exists another adversary B, and another source $\mathcal{M}'$ with $\mathbf{GP}_{\mathcal{M}}(\cdot) = \mathbf{GP}_{\mathcal{M}'}(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{PRV\$\text{-}CDA}}_{\mathsf{MLE},\mathcal{M}',B}(\lambda) \geq \frac{1}{2} \mathbf{Adv}^{\mathsf{PRV\text{-}CDA}}_{\mathsf{MLE},\mathcal{M},A}(\lambda),$$

for all $\lambda \in \mathbb{N}$ and $\mathbf{T}_B = \mathcal{O}(\mathbf{T}_A)$, and $\mathbf{T}_{\mathcal{M}'} = \mathcal{O}(\mathbf{T}_{\mathcal{M}})$, from which the proposition follows immediately. Towards that, consider source $\mathcal{M}'$ which picks a random bit b, runs $\mathcal{M}$ to get $\mathbf{M}_0, \mathbf{M}_1, Z$, and returns $\mathbf{M}_b, (Z, b)$. Clearly, $\mathcal{M}'$ has the same guessing probability as $\mathcal{M}$ and the running time of $\mathcal{M}'$ is proportional to $\mathcal{M}$, depending on implementation constants. Adversary B runs A with Z to get b', and returns 1 if $b = b'$ and 0 otherwise. The relation between the advantages follows. □

A similar argument can be used to show the adaptive equivalent, PRV$-CDA-A ⇒ PRV-CDA-A. The following proposition shows that the converse is not true: PRV-CDA ⇏ PRV$-CDA.

Proposition A.3. *There exists a scheme* MLE′ *that is* PRV-CDA *secure but not* PRV$-CDA *secure.*

Proof. Assume there exists an PRV-CDA secure $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$. Then, on top of MLE, we build another scheme $\mathsf{MLE}' = (\mathcal{P}, \mathcal{K}, \mathcal{E}', \mathcal{D}', \mathcal{T})$ such that (1) for every adversary A and source $\mathcal{M}$ there exists another adversary B running in comparable time with $\mathbf{Adv}^{\text{prv-cda}}_{\mathsf{MLE},\mathcal{M},B}(\lambda) = \mathbf{Adv}^{\text{prv-cda}}_{\mathsf{MLE}',\mathcal{M},A}(\lambda)$, for all $\lambda \in \mathbb{N}$; and (2) there exists an efficient adversary C such that $\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{MLE}',\mathcal{M},C}(\lambda) \geq \frac{1}{2}$ for all $\lambda \in \mathbb{N}$, for all sources $\mathcal{M}$.

Let MLE′ be such that $\mathcal{E}'$ runs $\mathcal{E}$ on its inputs and appends a 0-bit to the resulting ciphertext, and $\mathcal{D}'$ chops off the trailing bit of its ciphertext input and returns the result of $\mathcal{D}$ on the result. Given adversary A, adversary B simply appends a 0-bit to its input ciphertexts and runs A. Adversary C outputs 1 if the last bit of the first component of its input ciphertext is 0 and it outputs 0 otherwise. □

The adaptive analogue PRV-CDA-A ⇏ PRV$-CDA-A follows from a similar proposition and proof as above. The following proposition shows that PRV-CDA ⇏ PRV-CDA-A as one-time security does not imply adaptive security in the PRV-CDA-A sense. The proof follows from results for deterministic encryption [5] and hedged encryption [6] which rule out public-key dependent security.

Proposition A.4. *There exists a scheme* MLE′ *which is* PRV-CDA *secure but not* PRV-CDA-A *secure.*

Proof. We prove the proposition by taking an PRV-CDA-secure MLE scheme MLE, and modifying it so that it is still PRV-CDA secure, but no longer PRV-CDA-A secure. Specifically, we show that for every adversary A and source $\mathcal{M}$ there exists another adversary B running in comparable time with

$$\mathbf{Adv}^{\mathsf{PRV\text{-}CDA}}_{\mathsf{MLE},\mathcal{M},B}(\lambda) = \mathbf{Adv}^{\mathsf{PRV\text{-}CDA}}_{\mathsf{MLE}',\mathcal{M},A}(\lambda),$$

for all $\lambda \in \mathbb{N}$ and there exists an efficient adversary C and source $\mathcal{M}'$ with $\mathbf{GP}_{\mathcal{M}'}(\cdot) \leq 2\mathbf{GP}_{\mathcal{M}}(\cdot)$ such that $\mathbf{Adv}^{\mathsf{prv\text{-}cdaa}}_{\mathsf{MLE}',\mathcal{M}',C}(\lambda) \geq 1/2$, for all $\lambda \in \mathbb{N}$. Consider the MLE scheme MLE′ such that $\mathcal{E}'$ runs $\mathcal{E}$ on its inputs and appends the parameter P to the ciphertext returned by $\mathcal{E}$. $\mathcal{D}'$ chops off the trailing parameter P of its ciphertext input and returns the result of $\mathcal{D}$ on the result. Given adversary A, adversary B, when invoked with parameter P, ciphertext vector $\mathbf{C}$ and side-information Z, simply appends the parameter to each element of $\mathbf{C}$ and calls 0-bit to its input ciphertexts and runs A, echoing its output. In the PRV-CDA-A game, adversary C receives the parameter after the first query to Enc, and it can provide this parameter to $\mathcal{M}'$, which can now generate parameter-dependent messages. Now, we can show that $\mathcal{M}'$ can fix one or more bits of the tag, and thus communicate a single bit to the adversary. We omit the details, noting that this scenario of insecurity when the source has the parameter is similar to others arising in deterministic encryption [5] and hedged encryption [6] which rule out public-key dependent security in these settings. □

B Proof of PRV$-CDA for CE, HCE1, HCE2, RCE

We state a concrete security version of Theorem 4.1; the latter is a corollary of the concrete version.

Theorem B.1. *Let* SE *be a symmetric encryption scheme with key length* $k(\cdot)$ *and let* H *be a random oracle. Let* $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE1}, \mathsf{HCE2}, \mathsf{RCE}\}$ *be constructed using* SE *and* H*. For any source* $\mathcal{M}$ *with min-entropy* $\mu(\cdot)$ *and number of messages* $m(\cdot)$ *and any adversary* A *making* $q(\cdot)$ *queries to* H*, there exists adversaries* B_1, B_2 *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{XXX},\mathcal{M},A}(\lambda) \;\le\; qm \cdot \mathbf{Adv}^{\text{kr}}_{\mathsf{SE},B_1}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{ror}}_{\mathsf{SE},B_2}(\lambda) + \frac{4m^2}{2^k} + \frac{qm}{2^\mu}$$

where $q = q(\lambda)$, $\mu = \mu(\lambda)$, $m = m(\lambda)$, *and* $k = k(\lambda)$. *The running time of adversary* B_1 *and* B_2 *are each at most* $\mathbf{T}_A + \mathbf{T}_{\mathcal{M}} + cq(\lambda)$, *where* c *is a small implementation-dependent constant.* B_2 *makes at most* m *queries to its oracle.* □

Proof. We prove the theorem for the case of RCE; the case of $\mathsf{HCE2}$ proceeds similarly. The $\mathsf{PRV\$\text{-}CDA}$ security of $\mathsf{HCE2}$ implies immediately both the security of $\mathsf{HCE1}$ and the security of CE, the latter by way of a straightforward reduction. In this setting, where $\mathcal{M}$ does not have access to H, the parameter is unnecessary, and so we omit it in the rest of the proof.

Let $\text{PRV\$-CDA1}_{\mathsf{RCE},\mathcal{M}}$ be the $\text{PRV-CDA}_{\mathsf{RCE},\mathcal{M}}$ game with $b = 1$ and let $\text{PRV\$-CDA0}_{\mathsf{RCE},\mathcal{M}}$ be the $\text{PRV\$-CDA}_{\mathsf{RCE},\mathcal{M}}$ game with $b = 0$. A standard argument yields

$$\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{RCE},\mathcal{M},A}(\lambda) = \Pr\left[\,\text{PRV\$-CDA1}^A_{\mathsf{RCE},\mathcal{M}}(1^\lambda)\,\right] - \Pr\left[\,\text{PRV\$-CDA0}^A_{\mathsf{RCE},\mathcal{M}}(1^\lambda)\,\right].$$

Let $m = m(\lambda)$, $k = k(\lambda)$, $q = q(\lambda)$, and $\mu = \mu(\lambda)$. The first game G_1 (Figure 8) is identical to $\text{PRV\$-CDA1}_{\mathsf{RCE},\mathcal{M}}$. Note that while the loop in main does not check if $\mathtt{H}[\mathbf{M}[i]]$ was already defined by a previous iteration, but this does not change the implementation of a random oracle H because all sources ensure that $\mathbf{M}[i] \neq \mathbf{M}[j]$ for $i \neq j$. The game sets a flag bad, however, should a value $\mathbf{K}[i]$ collide with a value $\mathbf{K}[j]$ or $\mathbf{M}[j]$ for $j < i$. Game G_2 removes the boxed statement after bad. The two games are identical-until-bad, and so from the fundamental lemma of game-playing [11],

$$\Pr\left[\,\mathrm{G}_1^A\,\right] \le \Pr\left[\,\mathrm{G}_2^A\,\right] + \Pr\left[\,\mathrm{G}_2^A \text{ sets } \mathsf{bad}\,\right].$$

Because $\mathbf{K}[i]$ are chosen uniformly and independently of $\mathbf{M}$, we can bound the probability of bad being set via a union bound, giving that $\Pr[\mathrm{G}_2^A \text{ sets } \mathsf{bad}] \le 4m^2/2^k$.

Game G_3 defers updating of the table $\mathtt{H}$ with regards to the points $\mathbf{K}[i]$ and $\mathbf{T}[i]$ until they are needed due to a query to H. This change is invisible to the adversary, and we have $\Pr[\mathrm{G}_2^A] = \Pr[\mathrm{G}_3^A]$. Game G_3 sets a flag bad or bad' should such an H query occur. Game G_4 removes the boxed statements of G_3, which occur after bad or bad' is set. Games G_3 and G_4 are identical-until-bad or bad' and so

$$\Pr\left[\,\mathrm{G}_3^A\,\right] \le \Pr\left[\,\mathrm{G}_4^A\,\right] + \Pr\left[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}\,\right] + \Pr\left[\,\mathrm{G}_4 \text{ sets } \mathsf{bad}'\,\right].$$

In game G_5, we change the way in which $\mathbf{K}[i]$ is selected. Now, $\mathbf{C}_2[i]$ is sampled uniformly, and $\mathbf{K}[i]$ is set to be $\mathbf{C}_2[i] \oplus \mathbf{L}[i]$. This, in particular, makes the ciphertexts in $\mathbf{C}$ independent of the symmetric keys in $\mathbf{L}$. We have that $\Pr[\mathrm{G}_4^A] = \Pr[\mathrm{G}_5^A]$ as well as $\Pr[\mathrm{G}_4^A \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_5^A \text{ sets } \mathsf{bad}]$ and $\Pr[\mathrm{G}_4^A \text{ sets } \mathsf{bad}'] = \Pr[\mathrm{G}_5^A \text{ sets } \mathsf{bad}']$.

In game G_6 we defer the computation of $\mathbf{K}[i]$ values and the setting of bad or bad' until after A finishes execution. We have that $\Pr[\mathrm{G}_5^A] = \Pr[\mathrm{G}_6^A]$ as well as $\Pr[\mathrm{G}_5^A \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_6^A \text{ sets } \mathsf{bad}]$ and $\Pr[\mathrm{G}_5^A \text{ sets } \mathsf{bad}'] = \Pr[\mathrm{G}_6^A \text{ sets } \mathsf{bad}']$.

We now bound in game G_6 the probability that bad' is set, which corresponds to the adversary A querying one of the $\mathbf{K}[i]$ values. The adversary doing so reveals the symmetric key $\mathbf{L}[i]$ associated to that query, by way of $\mathbf{C}_2[i] \oplus \mathbf{K}[i]$. Let B_1 be $\mathrm{KR}_{\mathsf{SE}}$ adversary that works as shown in Figure 8. It guesses a hash query j^* and an encryption index i^*. After A finishes executing, it outputs $L = \mathbf{C}_2[i^*] \oplus \mathbf{X}[j^*]$ should $|X^{j^*}| = k$. A hybrid argument gives that

$$\Pr\left[\,\mathrm{G}_6^A \text{ sets } \mathsf{bad}'\,\right] \le qm \cdot \mathbf{Adv}^{\text{kr}}_{\mathsf{SE},B_1}(\lambda).$$

Game G_7 (Figure 9) is the same as G_6 except that the setting of bad' is dropped, and so $\Pr[\mathrm{G}_6^A] = \Pr[\mathrm{G}_7^A]$ as well as $\Pr[\mathrm{G}_6^A \text{ sets } \mathsf{bad}] = \Pr[\mathrm{G}_7^A \text{ sets } \mathsf{bad}]$ Game G_8 adds adds the boxed statement, replacing the

main $\boxed{\text{G}_1}$, G_2

$(\mathbf{M}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$

For $i = 1, \ldots, m$

  $\mathbf{L}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}_1[i] \leftarrow \mathcal{SE}(\mathbf{L}[i], \mathbf{M}[i])$

  $\mathbf{K}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathtt{H}[\mathbf{M}[i]] \leftarrow \mathbf{K}[i]$

  $\mathbf{C}_2[i] \leftarrow \mathbf{L}[i] \oplus \mathbf{K}[i]$

  $\mathbf{T}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  If $\mathtt{H}[\mathbf{K}[i]] \neq \perp$ then

    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathbf{T}[i] \leftarrow \mathtt{H}[\mathbf{K}[i]]}$

  $\mathtt{H}[\mathbf{K}[i]] \leftarrow \mathbf{T}[i]$

  $\mathbf{C}[i] \leftarrow \mathbf{C}_1[i] \,\|\, \mathbf{C}_2[i] \,\|\, \mathbf{T}[i]$

$b' \leftarrow\!\!\$\ A^{\mathrm{H}}(\mathbf{C}, Z)$; Ret b'

proc. H(X) // G_1, G_2

If $\mathtt{H}[X] \neq \perp$ then Ret $\mathtt{H}[X]$

$Y \leftarrow\!\!\$\ \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Ret Y

main $\boxed{\text{G}_3}$, G_4

$(\mathbf{M}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$

For $i = 1, \ldots, m$

  $\mathbf{L}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}_1[i] \leftarrow \mathcal{SE}(\mathbf{L}[i], \mathbf{M}[i])$

  $\mathbf{K}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}_2[i] \leftarrow \mathbf{L}[i] \oplus \mathbf{K}[i]$

  $\mathbf{T}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}[i] \leftarrow \mathbf{C}_1[i] \,\|\, \mathbf{C}_2[i] \,\|\, \mathbf{T}[i]$

$b' \leftarrow\!\!\$\ A^{\mathrm{H}}(\mathbf{C}, Z)$; Ret b'

proc. H(X)

For $i = 1, \ldots, m$

  If $X = \mathbf{M}[i]$ then

    $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\text{Ret } \mathbf{K}[i]}$

  If $X = \mathbf{K}[i]$ then

    $\mathsf{bad}' \leftarrow \mathsf{true}$; $\boxed{\text{Ret } \mathbf{T}[i]}$

If $\mathtt{H}[X] \neq \perp$ then Ret $\mathtt{H}[X]$

$Y \leftarrow\!\!\$\ \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Ret Y

main G_5

$(\mathbf{M}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$

For $i = 1, \ldots, m$

  $\mathbf{L}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}_1[i] \leftarrow \mathcal{SE}(\mathbf{L}[i], \mathbf{M}[i])$

  $\mathbf{C}_2[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{K}[i] \leftarrow \mathbf{C}_2[i] \oplus \mathbf{L}[i]$

  $\mathbf{T}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}[i] \leftarrow \mathbf{C}_1[i] \,\|\, \mathbf{C}_2[i] \,\|\, \mathbf{T}[i]$

$b' \leftarrow\!\!\$\ A^{\mathrm{H}}(\mathbf{C}, Z)$; Ret b'

proc. H(X)

For $i = 1, \ldots, m$

  If $X = \mathbf{M}[i]$ then $\mathsf{bad} \leftarrow \mathsf{true}$

  If $X = \mathbf{K}[i]$ then $\mathsf{bad}' \leftarrow \mathsf{true}$

If $\mathtt{H}[X] \neq \perp$ then Ret $\mathtt{H}[X]$

$Y \leftarrow\!\!\$\ \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Ret Y

main G_6

$(\mathbf{M}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$

For $i = 1, \ldots, m$

  $\mathbf{L}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}_1[i] \leftarrow \mathcal{SE}(\mathbf{L}[i], \mathbf{M}[i])$

  $\mathbf{C}_2[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{T}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}[i] \leftarrow \mathbf{C}_1[i] \,\|\, \mathbf{C}_2[i] \,\|\, \mathbf{T}[i]$

$b' \leftarrow\!\!\$\ A^{\mathrm{H}}(\mathbf{C}, Z)$

For $i = 1, \ldots, m$

  $\mathbf{K}[i] \leftarrow \mathbf{C}_2[i] \oplus \mathbf{L}[i]$

  If $\mathbf{M}[i] \in \mathcal{X}$ then $\mathsf{bad} \leftarrow \mathsf{true}$

  If $\mathbf{K}[i] \in \mathcal{X}$ then $\mathsf{bad}' \leftarrow \mathsf{true}$

Ret b'

proc. H(X)

$\mathcal{X} \leftarrow \mathcal{X} \cup \{X\}$

If $\mathtt{H}[X] \neq \perp$ then Ret $\mathtt{H}[X]$

$Y \leftarrow\!\!\$\ \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Ret Y

adversary B_1^{Enc}:

$(\mathbf{M}, Z) \leftarrow\!\!\$\ \mathcal{M}(1^\lambda)$

$i^* \leftarrow\!\!\$\ [1\,..\,m]$; $j^* \leftarrow\!\!\$\ [1\,..\,q]$

For $i = 1, \ldots, m$

  If $i = i^*$ then

    $\mathbf{C}_1[i] \leftarrow \mathrm{Enc}(\mathbf{M}[i])$

  Else

    $\mathbf{K}[i] \leftarrow\!\!\$\ \{0,1\}^k$

    $\mathbf{C}_1[i] \leftarrow \mathcal{SE}(\mathbf{K}[i], \mathbf{M}[i])$

  $\mathbf{C}_2[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{T}[i] \leftarrow\!\!\$\ \{0,1\}^k$

  $\mathbf{C}[i] \leftarrow \mathbf{C}_1[i] \,\|\, \mathbf{C}_2[i] \,\|\, \mathbf{T}[i]$

$b' \leftarrow\!\!\$\ A^{\mathrm{H}}(\mathbf{C}, Z)$

If $|\mathbf{X}[j^*]| = k$ then

  Ret $\mathbf{C}_2[i^*] \oplus \mathbf{X}[j^*]$

Ret $\perp$

proc. H(X)

$j \leftarrow j + 1$; $\mathbf{X}[j] \leftarrow X$

If $\mathtt{H}[X] \neq \perp$ then Ret $\mathtt{H}[X]$

$Y \leftarrow\!\!\$\ \{0,1\}^k$; $\mathtt{H}[X] \leftarrow Y$; Ret Y

Figure 8: Games used in the proof of Theorem 4.1.

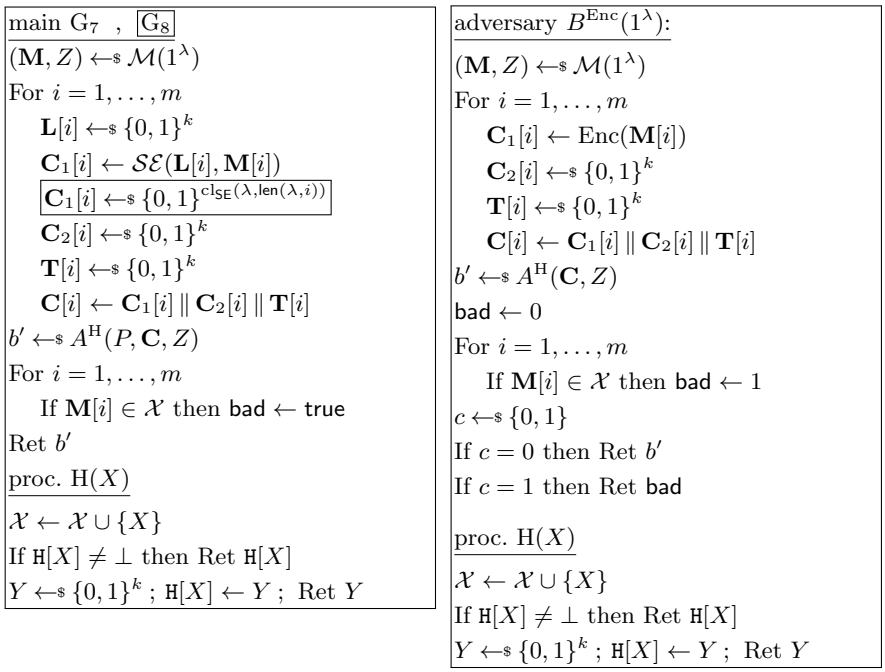


Figure 9: Games used in the proof of Theorem 4.1.

symmetric encryption ciphertext with random bits.

We now upper bound the transition from G_7 to G_8 by reducing to an $\mathrm{ROR}_{\mathsf{SE}}$ adversary B_2. The adversary is shown in Figure 9. It executes exactly G_7^A except that it uses its own Enc oracle to generate $\mathbf{C}_1[i]$ ciphertexts and it computes its output differently, randomly choosing whether to return b' or bad. (Here bad is set to 0 or 1 as opposed to true or false so B_2 can output the value of the flag as its return value.) We have that

$$\mathbf{Adv}^{\mathrm{ror}}_{\mathsf{SE},B_2}(\lambda) = \Pr\left[\mathrm{ROR}^{B_2}_{\mathsf{SE}}\right] = \frac{1}{2}\left(\Pr\left[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \mid c = 1\right] + \Pr\left[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \mid c = 2\right]\right) . \tag{1}$$

We investigate each conditional probability in the sum in turn. Let $\mathrm{ROR1}_{\mathsf{SE}}$ (resp. $\mathrm{ROR0}_{\mathsf{SE}}$) be the $\mathrm{ROR}_{\mathsf{SE}}$ game but with the challenge bit b set to 1 (resp. 0). Then

$$\begin{aligned}\Pr\left[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \mid c = 0\right] &= \Pr\left[\mathrm{ROR1}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \mid c = 0\right] - \Pr\left[\mathrm{ROR0}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \mid c = 0\right] \\ &= \Pr\left[\mathrm{G}_7^A\right] - \Pr\left[\mathrm{G}_8^A\right]\end{aligned}$$

Likewise we have that

$$\begin{aligned}\Pr\left[\mathrm{ROR}^{B_2}_{\mathsf{SE}} \mid c = 1\right] &= \Pr\left[\mathrm{ROR1}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \mid c = 1\right] - \Pr\left[\mathrm{ROR0}^{B_2}_{\mathsf{SE}} \Rightarrow 1 \mid c = 1\right] \\ &= \Pr\left[\mathrm{G}_7^A \text{ sets } \mathsf{bad}\right] - \Pr\left[\mathrm{G}_8^A \text{ sets } \mathsf{bad}\right]\end{aligned}$$

By rearranging and substituting into Equation (1) we get that

$$\Pr\left[\mathrm{G}_7^A\right] + \Pr\left[\mathrm{G}_7^A \text{ sets } \mathsf{bad}\right] = \Pr\left[\mathrm{G}_8^A\right] + \Pr\left[\mathrm{G}_8^A \text{ sets } \mathsf{bad}\right] + 2 \cdot \mathbf{Adv}^{\mathrm{ror}}_{\mathsf{SE},B_2}(\lambda) .$$

In game G_8, the choice of $\mathbf{M}$ is independent of the generation of $\mathbf{C}$. We can therefore apply the min-entropy of $\mathcal{M}$ in order to bound the probability that any hash query by A equals $P \,\|\, \mathbf{M}[i]$ for some i. Specifically, a union bound gives that

$$\Pr\left[\mathrm{G}_8^A \text{ sets } \mathsf{bad}\right] \leq \frac{qm}{2^\mu} .$$

Moreover, G_8 implements for A exactly the oracles of the $\mathrm{PRV\$\text{-}CDA0}_{\mathsf{RCE}}$ experiment. □

$\underline{\mathrm{MD}[E](M)}$	$\underline{\mathsf{CTR}[E](K, M)}$
$\mathbf{Y}[0] \leftarrow 0^n$; $\mathbf{B} \leftarrow \mathsf{split}(\mathsf{pad}(M,k),k)$; $\ell \leftarrow \lvert\mathbf{B}\rvert$	$\mathbf{B} \leftarrow \mathsf{split}(M,n)$; $\ell \leftarrow \lvert\mathbf{B}\rvert$
For $i = 1, \ldots, \ell$ do $\mathbf{Y}[i] \leftarrow E(\mathbf{B}[i], \mathbf{Y}[i-1]) \oplus \mathbf{Y}[i-1]$	For $i = 1, \ldots, \ell - 1$ do $\mathbf{C}[i] \leftarrow E(K, \langle i \rangle_n) \oplus \mathbf{B}[i]$
Ret $\mathbf{Y}[\ell]$	$X \leftarrow E(K, \langle \ell \rangle_n)$; $\mathbf{C}[\ell] \leftarrow \mathbf{B}[\ell] \oplus X[0, \lvert\mathbf{B}[\ell]\rvert]$
$\underline{\mathsf{split}(M,k)}$	Ret $\mathbf{C}$
For $i = 1, \ldots, \lfloor \lvert M\rvert/k \rfloor$ do $\mathbf{B}[i] \leftarrow M[ik, (i+1)k]$	$\underline{\mathsf{pad}(M,k)}$
$\mathbf{B}[\lfloor \lvert M\rvert/k \rfloor + 1] \leftarrow M[\lfloor \lvert M\rvert/k \rfloor k, \lvert M\rvert]$; Ret $\mathbf{B}$	$\ell \leftarrow k(\lfloor \lvert M\rvert/k \rfloor + 1) - k$; Ret $M \,\|\, 1 \,\|\, 0^\ell \,\|\, \langle \lvert M\rvert \rangle_k$

Figure 10: The Merkle-Damgard transform $\mathrm{MD}[E] : \{0,1\}^* \to \{0,1\}^n$ over (k,n)-block cipher E in Davies-Meyer mode for messages of length $< 2^k$, and $\mathsf{CTR}[E]$, the counter mode of operation of E. If $\ell \in \mathbb{N}$, then $\langle \ell \rangle_n$ denotes an n-bit encoding of ℓ.

C Instantiations and Performance of CE, HCE2, and RCE

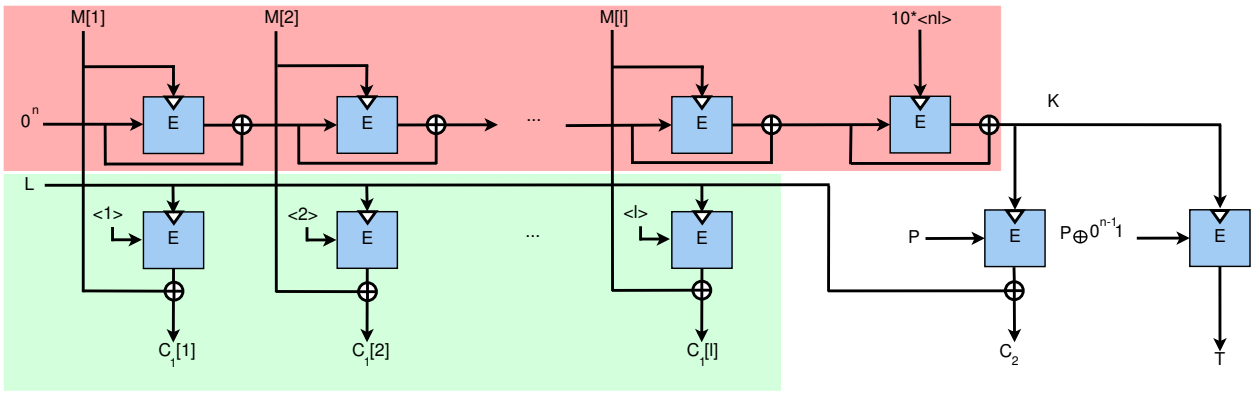
Block-cipher-only MLE. We define variants of CE, HCE1, HCE2, RCE that rely solely on a block cipher, such as AES. This has significant efficiency benefits given the widespread hardware support for AES. A block cipher is a map E: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ for which $E(K,\cdot) = E_K(\cdot)$ is a permutation with inverse $D(K,\cdot) = D_K(\cdot)$. Both E and D must be efficiently computable. We have for simplicity assumed that the key size and block size are equal. This is true of AES, and one can extend the constructions below to other cases.

In Figure 10 we define the hash function $\mathrm{MD}[E]$ with output length n bits and the one-time symmetric encryption algorithm $\mathsf{CTR}[E]$. The former is the Merkle-Damgard with strengthening transform applied to the Davies-Meyer compression function using E. The latter is standard CTR mode with a fixed IV. We define the MLE scheme $\mathsf{RCE}[E] = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{T}, \mathcal{D})$ via the algorithms below. A diagram is shown in Figure 11. The scheme $\mathsf{CE}[E]$ is derived from $\mathsf{RCE}[E]$ by: (1) setting $L = 0^n$, using C_2 as the symmetric key for $\mathsf{CTR}[E]$, and not including C_2 in the ciphertext; and (2) having tag generation hash the ciphertext by $E(\mathrm{MD}[E](M), P)$. Scheme $\mathsf{HCE2}[E]$ is the same except that encryption uses as key for $\mathsf{CTR}[E]$ the value C_2 and C_2 is not included in the ciphertext. The PRV$-CDA security of these variants can be shown in the ideal cipher model [34] using a proof that uses the preimage-awareness [21] of $\mathrm{MD}[E]$ together with an adaptation of the proof techniques used in the analysis of deterministic encryption schemes given in [42]. We omit it for the sake of brevity. The STC and TC security of the schemes are analogous to the results in Section 4.

Speed comparisons. We implemented the schemes

- $\mathsf{XXX}[\mathsf{CTR}[\mathsf{AES128}], \mathsf{H}]$ for $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE2}, \mathsf{RCE}\}$ and $\mathsf{H} \in \{\mathsf{SHA256}, \mathsf{SHA512}\}$
- $\mathsf{XXX}[E]$ for $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE2}, \mathsf{RCE}\}$ and $E \in \{\mathsf{AES128}, \mathsf{AES256}\}$

using OpenSSL version 1.0.0, with AES-NI [30] turned on for both AES128 and AES256. We measured the encryption performance when processing 4KB inputs. (Larger sizes have performance that degrades as expected due to inputs not fitting into the CPU cache.) The tests were performed with a warm cache (prior to the timings, the inputs were accessed several times) on an x86-64 Intel Core i7-970 processor clocking at 3.2 GHz; the test machine had 12GB of memory. The compiler, kernel and operating system on the machine were gcc-4.6, Linux kernel 3.0.0-13 and Ubuntu 11.04 respectively. The programs were compiled with the `-O3 -march=native` flags to produce optimized code and `-msse4` flag to enable support for Streaming SIMD instructions. Processor frequency scaling was turned off while running the experiments. The system was otherwise idle during the tests, and we used the `rdtsc` instruction of the x86 instruction set to measure time. Figure 12 lists the measured performance, in cycles per byte, of the various MLE schemes. In terms of absolute performance, the fastest among the schemes, $\mathsf{RCE}[\mathsf{AES256}]$ can encrypt a 4KB input and generate the tag in just over 8 microseconds. On the other hand, the CE instantiation with AES256 spends about 15 microseconds to generate the ciphertext and tag.



$\underline{\mathcal{P}(1^\lambda)}$	$\underline{\mathcal{E}_P(K, M)}$	$\underline{\mathcal{D}_P(K, (C_1, C_2, T))}$
$P \leftarrow\!\!\$\ \{0,1\}^n$; Ret P	$L \leftarrow\!\!\$\ \{0,1\}^n$	$L \leftarrow C_2 \oplus E(K, P)$
	$C_1 \leftarrow \mathsf{CTR}[E](L, M)$	$M \leftarrow \mathsf{CTR}[E](L, C_1)$
$\underline{\mathcal{K}_P(M)}$	$C_2 \leftarrow L \oplus E(K, P)$	$K' \leftarrow \mathrm{MD}[E](M)$
$K \leftarrow \mathrm{MD}[E](M)$; Ret K	$T \leftarrow E(K, P \oplus 0^{n-1}1)$	If $E(K', P \oplus 0^{n-1}1) = T$ then Ret M
	Ret (C_1, C_2, T)	Else Ret $\perp$

Figure 11: **Top:** Concerted single-pass key generation and encryption of a message of length ℓn using in $\mathsf{RCE}[E]$ for a block cipher E. A notch indicates the key input to the block cipher. The message blocks are $M[1], \ldots, M[n]$, and the ciphertext is $((C_1[1], \ldots, C_1[n]), C_2, T)$. The red shaded region corresponds to the hashing step and the green shaded region corresponds to CTR mode. **Bottom:** The algorithms defining MLE scheme $\mathsf{RCE}[E]$. The tag algorithm $\mathcal{T}_P(C_1, C_2, T)$ simply returns T.

CE variant	Operation	Choice of H			
		SHA256	SHA512	AES128	AES256
CE	$\mathcal{K}$	21.1	12.1	7.5	**5.3**
HCE2		21.1	12.1	7.5	**5.3**
RCE		21.1	12.1	7.5	**5.3**
CE	$\mathcal{E}$	1.2	1.2	1.2	**1.2**
HCE2		1.4	1.3	1.3	1.3
RCE		1.4	1.3	1.3	1.3
CE	$\mathcal{T}$	21.1	12.1	7.5	**5.3**
HCE2		–	–	–	–
RCE		–	–	–	–
CE	$\mathcal{K} + \mathcal{E} + \mathcal{T}$	43.4	25.4	16.3	11.8
HCE2		22.5	13.6	8.9	6.6
RCE		22.3	13.3	8.7	**6.5**
CE	$\mathcal{D}$	1.2	1.2	1.2	**1.2**
HCE2		22.5	13.6	8.9	6.6
RCE		22.3	13.3	8.7	6.5

Figure 12: Performance of CE instantiations in **cycles per byte**. The SHA256, SHA512, AES128, and AES256 columns relay performance of $\mathsf{XXX}[\mathsf{CTR}[\mathsf{AES128}], \mathsf{SHA256}]$, $\mathsf{XXX}[\mathsf{CTR}[\mathsf{AES128}], \mathsf{SHA256}]$, $\mathsf{XXX}[\mathsf{AES128}]$, and $\mathsf{XXX}[\mathsf{AES256}]$, respectively, for $\mathsf{XXX} \in \{\mathsf{CE}, \mathsf{HCE2}, \mathsf{RCE}\}$. The operation $\mathcal{K} + \mathcal{E} + \mathcal{T}$ is the total time for key generation, encryption and tag generation. The – symbol in tag operation performance indicates that a negligible amount of computation is needed to get the tag. The lowest cycles per byte for each of the operations is in bold.

main $G_1(\lambda)$

$S \leftarrow\$ \{0,1\}^{s(\lambda)}$; $K_h \leftarrow\$ \mathcal{HK}(1^\lambda)$; $b \leftarrow\$ \{0,1\}$; $(\mathbf{M}, Z) \leftarrow\$ \mathcal{M}(1^\lambda)$

For $i = 1, \ldots, |\mathbf{M}|$ do

  $M \leftarrow \mathbf{M}[i]$; $K \leftarrow \mathrm{Ext}_\lambda(S, M)$; $\ell \leftarrow |M|$; $M_1 \| \ldots \| M_\ell \leftarrow M$

  For $j = 1$ to ℓ do $C_j^1 \leftarrow \mathcal{H}(K_h, K \| j \| M_j)$; $C_j^0 \leftarrow\$ \{0,1\}^{|C_j^1|}$

  $\mathbf{C}[i] \leftarrow (C_1^b, \ldots, C_\ell^b)$

$b' \leftarrow\$ A(S, K_h, \mathbf{C}, Z)$

Ret $(b = b')$

Adversary $B(K_h, \mathbf{C}, (Z', S))$

Return $A(S, K_h, \mathbf{C}, Z')$

$\mathcal{M}'(1^\lambda)$

$S \leftarrow\$ \{0,1\}^{s(\lambda)}$; $(\mathbf{M}, Z) \leftarrow\$ \mathcal{M}(1^\lambda)$

$i' \leftarrow 0$

For $i = 1, \ldots, |\mathbf{M}|$ do

  $M \leftarrow \mathbf{M}[i]$; $K \leftarrow \mathrm{Ext}_\lambda(S, M)$

  $\ell \leftarrow |M|$; $M_1 \| \ldots \| M_\ell \leftarrow M$

  For $j = 1$ to ℓ do

    $\mathbf{M}'[i'] \leftarrow K \| j \| M_j$

    $i' \leftarrow i' + 1$

Ret $\mathbf{M}', (Z, S)$

Figure 13: Game G_1, adversary A and source $\mathcal{M}'$ for Theorem 5.1.

D Proof of Theorem 5.1

As per the theorem statement, let $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ be a family of hash functions and let $\mathrm{Ext} = \{\mathrm{Ext}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of extractors. Let $\ell(\cdot)$ be the input length and $k(\cdot)$ the output length of Ext. Below we let $\mathrm{XHC} = \mathrm{XHC}[\mathsf{H}, \mathrm{Ext}]$. Below we concentrate on the second part of the theorem (PRV$-CDA); the proof of the first part (PRV-CDA) proceeds in a similar fashion. Sources below are always therefore of arity one.

The following lemma relates the PRV$-CDA advantage of an adversary A against $\mathrm{XHC}[\mathrm{Ext}_\lambda, \mathsf{H}]$ to the PRV$-CDA advantage of a H adversary B.

Lemma D.1. *Let $\mathcal{M}$ be an* XHC*-valid source with message number $m(\cdot)$ and message lengths $\ell(\cdot)$. Let A be an adversary. Then there exists a* H*-valid source $\mathcal{M}'$ and adversary B such that for all $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{\text{prv\$-cda}}_{\mathrm{XHC},\mathcal{M},A}(\lambda) \leq \mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},\mathcal{M}',B}(\lambda) \ .$$

Moreover, $\mathcal{M}'$ is such that $\mathbf{GP}_{\mathcal{M}'}(\lambda) \leq 2^{-k(\lambda)} + \sqrt{2^{k(\lambda)} \cdot \mathbf{GP}_{\mathcal{M}}(\lambda)}$; its message number is $m'(\lambda) = m(\lambda) \cdot \ell(\lambda)$; and the length of each message it outputs is $k(\lambda) + 1 + \lceil \log_2(\ell(\lambda)) \rceil$. The running time of adversary B is $\mathbf{T}_B = \mathcal{O}(\mathbf{T}_A + m(\cdot)\ell(\cdot)t_{\mathcal{H}}(\cdot))$, and the running time of $\mathcal{M}'$ is $\mathbf{T}_{\mathcal{M}'} = \mathcal{O}(\mathbf{T}_{\mathcal{M}} + m(\cdot)t_{\mathrm{Ext}_\lambda(\cdot)})$ where $t_{\mathcal{H}}$ and t_{Ext_λ} are bounds on running times of of $\mathcal{H}$ and Ext_λ. □

Proof. Consider game G_1, source $\mathcal{M}'$ and adversary B of Figure 13. Game G_1 is the PRV$-CDA$_{\mathrm{XHC}}$ game and so it follows that

$$\mathbf{Adv}^{\text{prv\$-cda}}_{\mathrm{XHC},\mathcal{M},A}(\lambda) = 2 \cdot \Pr\left[\, G_1^A(\lambda) \,\right] - 1.$$

Adversary B plays the PRV$-CDA$_{\mathsf{H},\mathcal{M}'}$ game by running A. The source $\mathcal{M}'$ first picks a random extractor key S, then runs $\mathcal{M}$ to get $\mathbf{M}$ and then runs each component of $\mathbf{M}$ through Ext_λ with S to get the MLE keys $\mathbf{K}$. In the next step, it splits each $\mathbf{M}$ component into individual bits and prepends them with the appropriate MLE key and index to get $m(\lambda)\ell(\lambda)$ messages of the form $\mathbf{K}[i] \| \langle j \rangle \| \mathbf{M}[i][j]$ for $j \in [\ell(\lambda)]$ for $i \in [m(\lambda)]$. Then, $\mathcal{M}'$ outputs these messages along with auxiliary information Z of $\mathcal{M}$ and S. As B plays the PRV-CDA game, depending on the bit b in the game, either the hashes corresponding to $\mathbf{M}$, or random strings are provided to B, and this corresponds to getting XHC ciphertexts for $\mathbf{M}$, or random bits. Thus, B simulates G_1 for A with the bit of its PRV$-CDA game playing the role of the bit in G_1. Finally, when A finishes and outputs a bit, B also exits, echoing that bit. We have $\Pr[G_1^A(\lambda)] = \Pr[\text{PRV\$-CDA}^B_{\mathsf{H},\mathcal{M}'}(\lambda)]$ and hence

$$\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},\mathcal{M}',B}(\lambda) = 2 \cdot \Pr\left[\, G_1^A \,\right] - 1 = \mathbf{Adv}^{\text{prv\$-cda}}_{\mathrm{XHC},\mathcal{M},A}(\lambda).$$

Now, we relate $\mathcal{M}'$ and $\mathcal{M}$, by observing that S is picked at random, it follows from the properties of

main $G_1(\lambda)$
$S \leftarrow\$ \{0,1\}^{k(\lambda)}$; $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$
$g \leftarrow\$ m(\lambda)$; $b \leftarrow\$ \{0,1\}$; $\mathbf{M}, Z \leftarrow\$ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, \lvert\mathbf{M}\rvert$ do
$M_1 \leftarrow \mathsf{Proj}(\mathbf{M}[i], U)$; $M_2 \leftarrow \mathsf{Proj}(\mathbf{M}[i], [n(\lambda)] \setminus U)$
$K \leftarrow \mathrm{Ext}_\lambda(S, M_1)$
If $i < g + b$ then $\mathbf{C}[i] \leftarrow \mathcal{SE}' K, M_2)$
If $i = g$ and $b = 1$ then $\mathbf{C}[i] \leftarrow \mathcal{SE}(K, M_2)$
If $i = g$ and $b = 0$ then $\mathbf{C}[i] \leftarrow\$ \{0,1\}^{\lvert\mathcal{SE}(K,M_2)\rvert}$
If $i > g$ then $C' \leftarrow \mathcal{SE}(K, M_2)$; $\mathbf{C}[i] \leftarrow\$ \{0,1\}^{C'}$
$b' \leftarrow\$ A(S \lvert\lvert U, \mathbf{C}, Z)$; Ret $(b = b')$

main G_2
$S \leftarrow\$ \{0,1\}^{k(\lambda)}$; $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$
$g \leftarrow\$ m(\lambda)$; $b \leftarrow\$ \{0,1\}$; $\mathbf{M}, Z \leftarrow\$ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, \lvert\mathbf{M}\rvert$ do
$M_1 \leftarrow \mathsf{Proj}(\mathbf{M}[i], U)$; $M_2 \leftarrow \mathsf{Proj}(\mathbf{M}[i], [n(\lambda)] \setminus U)$
$K \leftarrow \mathrm{Ext}_\lambda(S, M_1)$
If $i < g + b$ then $\mathbf{C}[i] \leftarrow \mathcal{SE}(K, M_2)$
If $i = g$ and $b = 1$ then $\mathbf{C}[i] \leftarrow \mathcal{SE}(K, M_2)$
If $i = g$ and $b = 0$ then $K' \leftarrow\$ \{0,1\}^{k_s}$; $\mathbf{C}[i] \leftarrow \mathcal{SE}(K', M_2)$
If $i > g$ then $\mathbf{C}[i] \leftarrow\$ \{0,1\}^{\lvert\mathcal{SE}(K,M_2)\rvert}$
$b' \leftarrow\$ A(S \lvert\lvert U, \mathbf{C}, Z)$; Ret $(b = b')$

Figure 14: Games for Theorem 5.2.

Ext_λ that for a randomly picked $K \leftarrow\$ \{0,1\}^{k(\lambda)}$

$$\begin{aligned}\mathbf{GP}(\mathrm{Ext}_\lambda(S, \mathbf{M}[i]) \mid\mid j \mid\mid \mathbf{M}[i][j] | S, Z) &\leq \mathbf{GP}(K|S,Z) + \mathsf{SD}((S, \mathrm{Ext}_\lambda(S, \mathbf{M}[i]), Z); (S, K, Z)) \\ &\leq \frac{1}{2^{k(\lambda)}} + \sqrt{2^{k(\lambda)} \mathbf{GP}(\mathbf{M}[i]|Z)} \leq \frac{1}{2^{k(\lambda)}} + \sqrt{2^{k(\lambda)} \mathbf{GP}_{\mathcal{M}}(\lambda)}\end{aligned}$$

for all $i \in [\ell(\lambda)]$, for all $i \in [m(\lambda)]$. This completes the proof of the lemma. □

To finish the proof of the theorem in the PRV$-CDA case, we note that $\mathbf{GP}_{\mathcal{M}'}(\cdot)$ is negligible when $2^{k(\cdot)} \cdot \mathbf{GP}_{\mathcal{M}}(\cdot)$ is negligible. In turn, $\mathbf{Adv}^{\text{prv\$-cda}}_{\mathsf{H},\mathcal{M}',B}(\cdot)$ is negligible for all PT B and hence $\mathbf{Adv}^{\text{prv\$-cda}}_{\text{XHC},\mathcal{M},A}(\cdot)$ is negligible for all PT A.

E Proof of Theorem 5.2

Proof. Game G_1 of Figure 14 is a hybrid between the PRV$-CDA with $b = 0$ and $b = 1$, with the code of SXE and $\mathcal{M}$. We have

$$\Pr[G_1^A(\lambda)] \geq \frac{1}{m(\lambda)} \mathbf{Adv}^{\text{PRV\$-CDA}}_{\mathsf{SXE}[\mathrm{Ext},\mathsf{SE},p],\mathcal{M},A}(\lambda).$$

Game G_2 is like G_1, but makes a modification in how the ciphertext corresponding to the switching value g is treated. Specifically, it picks a random $g \leftarrow\$ [m(\lambda)]$, creates the first $g - 1$ ciphertexts and later $m(\lambda) - g$ ciphertexts as in G_1, but the g-th ciphertext is set to an encryption of the g-th plaintext under a random key, if $b = 0$, as opposed to a random string as in G_1.

The difference between G_1 and G_2 can be bounded by a simple single key ROR adversary B which runs the hybrid, and handles its switching ciphertext as follows. It flips a random bit b, and if $b = 1$, it runs the extractor on the g-th input to get the key and encrypts the g-th message under this key. If $b = 0$, then it forwards the g-th message to its Enc oracle. Depending on whether the bit in the ROR game is 0 or 1, adversary A is simulated either in G_1 or G_2. Moreover, B never makes more than one query to its Enc oracle. We have, $(\lvert \Pr[G_1^A(\lambda)] - \Pr[G_2^A(\lambda)] \rvert)/2 = \mathbf{Adv}^{\mathsf{1\text{-}ror}}_{\mathsf{SE},B}(\lambda)$ for all $\lambda \in \mathbb{N}$.

In game G_2, the difference between the settings of $b = 0$ and $b = 1$ is in how the key for g-th plaintext is derived. If $b = 0$, the extractor output is used, and if $b = 1$ then a random key is chosen. Importantly, this is the last plaintext for which the game replies with a real encryption; subsequent components of $\mathbf{C}$

are simply random strings. We have

$$\begin{aligned}\Pr[G_2^A(\lambda)] &\leq \mathsf{SD}((\mathcal{SE}(K, \mathsf{Proj}(M_g, [n(\lambda)] \setminus U)), M_1, \ldots, M_{g-1}, Z, S), \\ &\qquad (\mathcal{SE}(\mathrm{Ext}_\lambda(S, \mathsf{Proj}(M_g, U)), \mathsf{Proj}(M_g, [n(\lambda)] \setminus U)), M_1, \ldots, M_{g-1}, Z, S)) \\ &\leq \mathsf{SD}(K, \mathsf{Proj}(M_g, [n(\lambda)] \setminus U), M_1, \ldots, M_{g-1}, Z, S), \\ &\qquad (\mathrm{Ext}_\lambda(S, \mathsf{Proj}(M_g, U))), \mathsf{Proj}(M_g, [n(\lambda)] \setminus U), M_1, \ldots, M_{g-1}, Z, S)).\end{aligned}$$

In the above probabilities, $U \leftarrow_{p(\lambda)} \{0,1\}^{n(\lambda)}$. The source $\mathcal{M}$ is such that

$$\mathbf{GP}_{\mathcal{M},p}(\lambda) = \max_i \mathbf{GP}(\mathsf{Proj}(\mathbf{M}[i], U) \mid (\mathbf{M}[1], \ldots, \mathbf{M}[i-1], \mathsf{Proj}(\mathbf{M}[i], [n(\lambda)] \setminus U), Z)) = 2^{-k(\lambda)}\nu(\lambda),$$

where ν is some negligible function. Applying that Ext is an extractor, the above statistical distance is bounded by $\sqrt{2^{k(\lambda)}\mathbf{GP}_{\mathcal{M},p}(\lambda)} = \sqrt{\nu(\lambda)}$. Putting these together, we have

$$\mathbf{Adv}^{\mathsf{PRV\$\text{-}CDA}}_{\mathsf{SXE}[\mathrm{Ext},\mathsf{SE},p],\mathcal{M},A}(\lambda) \leq m(\lambda)(\sqrt{\nu(\lambda)} + \mathbf{Adv}^{\mathsf{1\text{-}ror}}_{\mathsf{SE},B}(\lambda)).$$

Since both the quantities on the right are negligible and A and $\mathcal{M}$ are PT algorithms, the PRV$-CDA advantage of A is negligible and $\mathsf{SXE}[\mathrm{Ext}, \mathsf{SE}, p]$ is PRV$-CDA secure for these types of sources.

□