# Message-Locked Encryption and Secure Deduplication

Mihir Bellare[1], Sriram Keelveedhi[1], and Thomas Ristenpart[2]

[1]Department of Computer Science & Engineering, University of California San Diego,
http://cseweb.ucsd.edu/~mihir/, http://cseweb.ucsd.edu/~skeelvee/.
[2]Department of Computer Sciences, University of Wisconsin–Madison,
http://pages.cs.wisc.edu/~rist/.

**Abstract.** We formalize a new cryptographic primitive that we call 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.

## 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, 5, 7, 8, 10, 22, 23, 31, 35, 39, 43], 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) [16]. In the second category we address the challenging question of finding a standard-model MLE scheme, making connections with deterministic public-key encryption [11], correlated-input-secure hash functions [27] and locally-computable extractors [9, 30, 40] to provide schemes exhibiting different trade-offs 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 [6], Dropbox [3] and bitcasa [1] 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 $\mathsf{pw}_A$. Bob would store $C_B$, the encryption of $M$ under his password $\mathsf{pw}_B$. 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 [11, 20, 38] 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) [24] 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 metadata 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 [1,2,4,5,8,31,35,39,43]. 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 Fig. 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 [13,26]. 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 [11], we therefore ask for the best possible privacy, namely semantic security when messages are unpredictable (have high min-entropy). Adapting definitions from [11, 12, 14, 21] we formalize a PRV-CDA notion where encryptions of two unpredictable messages should be indistinguishable. ("cda" stands for "chosen-distribution attack" [12].) 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. [37]).

These basic notions are for non-adaptive adversaries. The corresponding adaptive versions are 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$-CDA-A. (See the right hand side of Fig. 1 for a comprehensive relations summary.) Thus PRV$-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 [39]. We define tag consistency to rule out these types of integrity violations. Notion $\mathsf{TC}$ 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 $\mathsf{STC}$ ("$\mathsf{S}$" 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. $\mathsf{STC}$ is strictly stronger than $\mathsf{TC}$; we define both because, as we will see, some schemes meet only the weaker, but still meaningful, $\mathsf{TC}$ version.

### 1.3 Practical Contributions

The definitional framework outlined above puts us in a position to rigorously assess —a decade after its inception in [24]— 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 $\mathsf{CE}$ and $\mathsf{HCE1}$, 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 $\mathsf{SE}$. $\mathsf{CE}$ with $\mathsf{SE}$ set to a blockcipher, for example, is the scheme of [24] and $\mathsf{HCE1}$ with $\mathsf{SE}$ as a blockcipher in counter mode with fixed IV is used within the Tahoe FileSystem (TahoeFS) [43].

$\mathsf{CE}$ sets $K = H(M)$, $C = \mathsf{SE}(K, M)$ and tag $T = H(C)$, while $\mathsf{HCE1}$ sets $K = H(M)$, $C = \mathsf{SE}(K, M) \| H(K)$ and $T = H(K)$. The rationale for $\mathsf{HCE1}$ 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 $\mathsf{HCE1}$ is vulnerable to duplicate faking attacks, meaning it does not even achieve $\mathsf{TC}$ security. We discuss the implications for the security of TahoeFS in Section 4.

We ask whether performance gains of the type offered by $\mathsf{HCE1}$ over $\mathsf{CE}$ can be obtained without loss in consistency, and offer as answers two new schemes, $\mathsf{HCE2}$ and $\mathsf{RCE}$. The former is as efficient as $\mathsf{HCE1}$. $\mathsf{RCE}$ 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, $\mathsf{HCE2}$ needs two passes, one pass to generate the key and a second for encryption, while $\mathsf{CE}$ needs a third pass for producing the tag. $\mathsf{RCE}$ achieves this via a novel use of randomization (all previous schemes were deterministic). Roughly, 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 [15] show that RCE does indeed outperform the other two.

Fig. 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. Whether this is fundamental or if better schemes exist is an interesting open question.

### 1.4 Theoretical Contributions

Is MLE possible in the standard-model? 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) [11] and Correlated-input-secure Hash Functions (CI-H) [27] 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 Fig. 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 [11]. 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, described in Section 5, is quite different and does not exploit the decryptability 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 [27] 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.

| 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 | ✓ | ✓ | ✓ | ✓ |

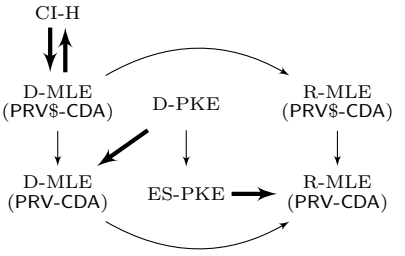


**Fig. 1. Left:** For each MLE scheme that we construct, we indicate whether it is in the RO or standard model; whether it is deterministic or randomized; and which security properties it is proven to possess. The assumptions for XtCIH, XtDPKE and XtESPKE are, respectively, a CI-H function, a D-PKE scheme 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.

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 [41] 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 [11], 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 [14, 19, 21, 25]. CI-H functions have been built for messages given by polynomials evaluated at the same random point [27]. 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 [9,30,40] and the sample-then-extract paradigm [33, 40]. 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

There are folklore suggestions along the lines of CE predating [24]. See [34].

Recall that we have 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 [12]. We suggest that progress can be made by defining and then targeting indistinguishability-from-random style definitions.

Mironov, Pandey, Reingold and Segev [32] suggest deduplication as a potential application of their incremental deterministic public-key encryption scheme. But this will only work with a single client. It won't allow deduplication across clients, since they would all 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 [29]. MLE is compatible with either client- or server-side deduplication (the latter prevents such side-channels). We note that one of our new schemes, RCE, gives rise to such a side-channel (see Section 4). MLE targets a different class of threats than proofs of ownership [28], which were proposed for deduplication systems in order to mitigate abuse of services for surreptitious content distribution.

In independent and concurrent work, Xu, Chang and Zhou [44] consider leakage resilience in the deduplication setting. They provide a randomized construction similar to RCE.

## 2 Preliminaries

Notations and Conventions. The empty string is denoted by $\varepsilon$. 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 $1^\lambda$ its unary representation.

"PT" stands for "polynomial-time." Algorithms are randomized unless otherwise indicated. 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.

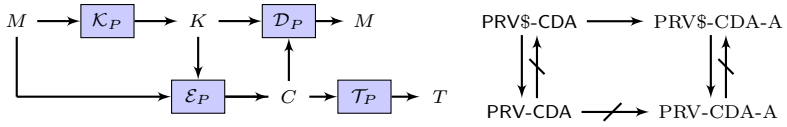


**Fig. 2. Left:** Depiction of syntax of MLE scheme $\mathsf{MLE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{T})$. The parameter generation algorithm is not shown. **Right:** Relations between notions of privacy for MLE schemes. An arrow from A to B means that any A-secure MLE scheme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

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 $Y$. For our security definitions and proofs we use the code-based game playing framework of [17], though adopting some of the syntax and semantics of [36].

## 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 Fig. 2. On input $1^\lambda$ 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 $\mathsf{Cl}$ 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 $\mathsf{MLE}$ is deterministic if $\mathcal{K}$ and $\mathcal{E}$ are deterministic. We observe that if $\mathsf{MLE}$ 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 $\mathsf{Da}$, 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 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 [11]. These requirements could be relaxed to define MLE variants where search was allowed linear time (cf. [20]) or search ability was not even provided. MLE does not appear easy to achieve even in the last case.

PRIVACY. A *source* is a PT algorithm $\mathcal{M}$ that on input $1^\lambda$ 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 $\mathsf{len}$, 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

| main PRV-CDA$^{A}_{\mathsf{MLE},\mathcal{M}}(\lambda)$ | main PRV\$-CDA$^{A}_{\mathsf{MLE},\mathcal{M}}(\lambda)$ | main $\boxed{\mathrm{TC}^{A}_{\mathsf{MLE}}(\lambda)}$ $\mathrm{STC}^{A}_{\mathsf{MLE}}(\lambda)$ |
|---|---|---|
| $P \leftarrow\$ \mathcal{P}(1^\lambda)$ | $P \leftarrow\$ \mathcal{P}(1^\lambda)$ | $P \leftarrow\$ \mathcal{P}(1^\lambda)$; $(M, C') \leftarrow\$ A(P)$ |
| $b \leftarrow\$ \{0,1\}$ | $b \leftarrow\$ \{0,1\}$ | If $(M = \bot)$ or $(C' = \bot)$ then |
| $(\mathbf{M}_0, \mathbf{M}_1, Z) \leftarrow\$ \mathcal{M}(1^\lambda)$ | $(\mathbf{M}, Z) \leftarrow\$ \mathcal{M}(1^\lambda)$ | Ret false |
| For $i = 1, \ldots, \lvert\mathbf{M}_b\rvert$ do | For $i = 1, \ldots, \lvert\mathbf{M}\rvert$ do | $T \leftarrow\$ \mathcal{T}_P(\mathcal{E}_P(\mathcal{K}_P(M), M))$ |
| $\quad \mathbf{K}[i] \leftarrow\$ \mathcal{K}_P(\mathbf{M}_b[i])$ | $\quad \mathbf{K}[i] \leftarrow\$ \mathcal{K}_P(\mathbf{M}[i])$ | $T' \leftarrow\$ \mathcal{T}_P(C')$ |
| $\quad \mathbf{C}[i] \leftarrow\$ \mathcal{E}_P(\mathbf{K}[i], \mathbf{M}_b[i])$ | $\quad \mathbf{C}_1[i] \leftarrow\$ \mathcal{E}_P(\mathbf{K}[i], \mathbf{M}[i])$ | $M' \leftarrow\$ \mathcal{D}_P(\mathcal{K}_P(M), C')$ |
| $b' \leftarrow\$ A(P, \mathbf{C}, Z)$ | $\quad \mathbf{C}_0[i] \leftarrow\$ \{0,1\}^{\lvert\mathbf{C}_1[i]\rvert}$ | If $(M = M')$ then Ret false |
| Ret $(b = b')$ | $b' \leftarrow\$ A(P, \mathbf{C}_b, Z)$ | If $(T \neq T')$ then Ret false |
| | Ret $(b = b')$ | $\boxed{\text{If } (M' = \bot) \text{ then Ret false}}$ |
| | | Ret true |

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

$i \in [m(\lambda)]$ 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 Fig. 3, "CDA" stands for "Chosen-Distribution Attack," referring to the distribution on messages imposed by the MLE-valid source $\mathcal{M}$, which in game PRV-CDA has arity 2 and in game PRV$-CDA has arity 1. 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 [11,14,21] and hedged [12] 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 [37].

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 [11], 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 the full version [15], we give corresponding adaptive definitions PRV-CDA-A and PRV$-CDA-A, and prove the relations summarized in Fig. 2. The one we highlight is that non-adaptive PRV$-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 Fig. 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 duplicate faking attacks that lead to erasures: a client can detect corruption but no longer be able to recover their message. STC (strong tag consistency) aims to additionally provide security against such 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 [39], 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 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 her message. If not, she complains. This however is not optimal, being expensive and complex and leading to deduplication side-channels (cf. [29]).

If an MLE scheme is deterministic, letting the tag equal the ciphertext will result in a scheme that is STC secure. 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. An RO [16] 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 $\mathtt{H}[x,k]$. We will omit the length $k$ when it is clear from context. In the ROM, both scheme algorithms and adversary algorithms will have access to H.

In lifting the privacy definitions to the ROM, 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 with access to H under an extension of the definition of unpredictability to this setting in which, following [12,36], the unpredictability of the source is independent of the coins underlying H.

## 4 The Security of Fast MLE Schemes

We investigate four MLE schemes, two that correspond to in-use schemes and two new 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 of algorithms $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$: key generation $\mathcal{SK}$, on input $1^\lambda$, 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 $\mathsf{SE}$ 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 (KR) and one-time real-or-random security (ROR) [37]. Formal definitions are recalled in [15].

The four schemes. Let $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ be a symmetric encryption scheme and $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ be a hash function family. All schemes inherit their message space from $\mathsf{SE}$ (typically $\{0,1\}^*$), use as parameter generation $\mathcal{HK}$, and share a common key generation algorithm $\mathcal{K}$ which derives keys as $K \leftarrow \mathcal{H}(P, M)$.

The first scheme, that we simply call convergent encryption ($\mathsf{CE}$), generalizes the original scheme of DABST [24]. $\mathsf{CE}$ encrypts the message as $C \leftarrow \mathcal{SE}(K, M)$. Tags are computed as $T \leftarrow \mathcal{H}(P, C)$. (One could alternatively use the ciphertext itself as the tag, but this is typically not practical.) Decryption returns $M \leftarrow \mathcal{SD}(K, C)$ on input $K, C$. The second scheme, $\mathsf{HCE1}$ (Hash-and-CE 1), is a popular variant of the CE scheme used in a number of systems [4, 22, 23, 43]. Compared to $\mathsf{CE}$, $\mathsf{HCE1}$ computes tags during encryption by hashing the per-message key ($T \leftarrow \mathcal{H}(P, K)$) 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 passes needed to encrypt and generate a tag from three to two.

$\mathsf{HCE1}$ is vulnerable to attacks that break $\mathsf{TC}$ security, as first discussed in [39]. 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 any arbitrary message. In TahoeFS's use of $\mathsf{HCE1}$, the client additionally stores a message authentication code (MAC) computed over the message, and checks this MAC during decryption. This means

that the TC attack against HCE1 would be detected. TahoeFS is, however, still vulnerable to erasure attacks. We have reported this to the developers, and are discussing possible fixes with them.

We suggest a new scheme, HCE2, that modifies HCE1 to directly include a mechanism, called guarded decryption, that helps it to achieve TC security. The decryption routine now additionally checks the tag embedded in the ciphertext by recomputing the tag using the just-decrypted message. If the check fails, then $\perp$ is returned.

For performance 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 passes to perform encryption and tag generation, while HCE1 and HCE2 require two. Using at least two passes here is fundamental: deterministic MLE schemes that output bits of ciphertext before processing most of the message will not achieve PRV-CDA security.

The fourth scheme, Randomized Convergent Encryption (RCE), 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 $L$ 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.

We note that RCE does admit a side-channel attack similar to those arising in client-side deduplication systems [29]. A user can infer whether she is the first to upload a file, by first storing its encryption under RCE and then immediately downloading to check if the recovered ciphertext is the one just uploaded.

For all the schemes, it is easy to verify decryption correctness. Tag correctness follows as the tags are all deterministic.

Privacy. We prove the following theorem, which establishes the PRV$-CDA security of the four schemes when modeling H as a RO, in the full version [15]. The key-recovery (KR) and one-time real-or-random (ROR) security notions referred to below are recalled in [15].

**Theorem 1.** *Let* H *be a RO and let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme with key length* $k(\cdot)$*. 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. 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. Here CR refers to standard collision resistance, the definition being recalled in [15].

**Theorem 2.** *Let* $\mathsf{SE} = (\mathcal{SK}, \mathcal{SE}, \mathcal{SD})$ *be a one-time symmetric encryption scheme and let* $\mathsf{H} = (\mathcal{HK}, \mathcal{H})$ *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 in theory achieve STC security using non-interactive zero-knowledge proofs [18], but this would obviate the speedups offered by the schemes compared to CE. We conclude that 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. In [15] we explore various instantiations of the MLE schemes, including ones that are entirely built from AES. We also report on performance there.

## 5 Constructions without ROs

We overview Extract-Hash-Check (which yields standard model MLE from D-PKE or CI-H hash functions) and Sample-Extract-Encrypt (which yields the same from weaker assumptions but for particular classes of sources). Refer to [15] for full construction descriptions and security proofs.

Extract-Hash-Check. 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. Details may be found in [15].

Sample-Extract-Encrypt. This MLE scheme relies only on a standard and weak assumption, namely a one-time symmetric encryption scheme, which can

be built from any one-way function. The tradeoff is that it is only secure 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 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 [19]. 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.

Sample-Extract-Encrypt (SXE) builds an MLE scheme for 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. Refer to [15] for details.

## Acknowledgments

We thank James Lentini for pointing out the STC attack against RCE, Ananth Raghunathan for early discussions about convergent encryption, the TahoeFS developers for discussions regarding duplicate faking attacks in TahoeFS, and the anonymous reviewers for their helpful comments. Bellare and Keelveedhi were supported in part by NSF grants CNS-0904380, CCF-0915675 and CNS-1116800. Ristenpart was supported in part by NSF grant CNS-1065134 and gifts from RSA Labs and Microsoft.

## References

1. Bitcasa, inifinite storage. `http://www.bitcasa.com/`.
2. Ciphertite data backup. `http://www.ciphertite.com/`.
3. Dropbox, a file-storage and sharing service. `http://www.dropbox.com/`.
4. The Flud backup system. `http://flud.org/wiki/Architecture`.
5. GNUnet, a framework for secure peer-to-peer networking. `https://gnunet.org/`.
6. Google Drive. `http://drive.google.com/`.
7. 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.
8. P. Anderson and L. Zhang. Fast and secure laptop backups with encrypted deduplication. In *Proc. of USENIX LISA*, 2010.
9. 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.

10. C. Batten, K. Barr, A. Saraf, and S. Trepetin. pStore: A secure peer-to-peer backup system. *Unpublished report, MIT Laboratory for Computer Science*, 2001.
11. 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.
12. 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.
13. 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.
14. 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.
15. M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. Cryptology ePrint Archive, Report 2012/631, 2012. `http://eprint.iacr.org/`.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. J. Cooley, C. Taylor, and A. Peacock. ABS: the apportioned backup system. *MIT Laboratory for Computer Science*, 2004.
23. 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.
24. J. Douceur, A. Adya, W. Bolosky, D. 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.
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.
26. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

27. 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.
28. 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.
29. 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.
30. 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.
31. 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.
32. 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.
33. N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.
34. J. Pettitt. Content based hashing. `http://cypherpunks.venona.com/date/1996/02/msg02013.html`.
35. 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.
36. 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.
37. 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.
38. 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.
39. 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.
40. 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.
41. D. Wichs. Barriers in cryptography with weak, correlated and leaky sources. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 111–126, New York, NY, USA, 2013. ACM.
42. Z. Wilcox-O'Hearn. Convergent encryption reconsidered, 2011. `http://www.mail-archive.com/cryptography@metzdowd.com/msg08949.html`.
43. 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.
44. J. Xu, E.-C. Chang, and J. Zhou. Leakage-resilient client-side deduplication of encrypted data in cloud storage. Cryptology ePrint Archive, Report 2011/538, 2011. `http://eprint.iacr.org/`.