Verifiable Message-Locked Encryption

Sébastien Canard[1], Fabien Laguillaumie[2], and Marie Paindavoine[1,2]

[1] Orange Labs, Applied Crypto Group, Caen, France
{sebastien.canard, marie.paindavoine}@orange.com
[2] LIP (U.Lyon, CNRS, ENS Lyon, INRIA, UCBL),
Université Claude Bernard Lyon 1, Villeurbanne, France
{fabien.laguillaumie, marie.paindavoine}@ens-lyon.fr

Abstract. One of today's main challenge related to cloud storage is to maintain the functionalities and the efficiency of customers' and service providers' usual environments, while protecting the confidentiality of sensitive data. Deduplication is one of those functionalities: it enables cloud storage providers to save a lot of memory by storing only once a file uploaded several times. But classical encryption blocks deduplication. One needs to use a "message-locked encryption" (MLE), which allows the detection of duplicates and the storage of only one encrypted file on the server, which can be decrypted by any owner of the file. However, in most existing scheme, a user can bypass this deduplication protocol. In this article, we provide servers verifiability for MLE schemes: the servers can verify that the ciphertexts are well-formed. This property that we formally define forces a customer to prove that she complied to the deduplication protocol, thus preventing her to deviate from *the prescribed functionality* of MLE. We call it *deduplication consistency*. To achieve this deduplication consistency, we provide (i) a generic transformation that applies to any MLE scheme and (ii) an ElGamal-based deduplication-consistent MLE, which is secure in the random oracle model.

1 Introduction

Cloud computing is often promoted towards companies as a way to reduce their costs while increasing accessibility and flexibility. It is common sense to have one large computing infrastructure that companies would share instead of replicating smaller ones. This saves money and is an eco-friendlier way to distribute resources. But cloud platforms are neither cheap nor eco-friendly. The larger amount of data these platforms host, the more expensive they become. Impact on the environment grows as well. One way to address this issue is to delete identical files stored by different users. This method, called *deduplication*, is widely used by cloud providers. However, some of the cloud storage users may want to encrypt their data, distrusting honest-but-curious providers. If they use a classical encryption scheme, deduplication is not possible anymore: two encryptions of the same plaintext under different keys yield indistinguishable ciphertexts. A new kind of encryption is needed, under which it is possible to determine whether two different ciphertexts are *locked* to the same message or not.

Previous Work. The work on the *message-locked encryption* model has been initiated by Douceur et al. [8] with their *convergent encryption* (CE) scheme. The main idea is very simple: everyone that encrypts the same message m will obtain the *same* ciphertext c. The convergent encryption protocol CE given in [8] uses a hash function H (which is modelled as a random oracle for the security proof) and a deterministic symmetric encryption scheme SE: it sets the encryption and decryption key as $K = H(M)$, where M is the message to be encrypted, and the ciphertext C is computed as $\mathsf{SE.Encrypt}(M, K)$. The ciphertext is concatenated to a tag $\tau = H(C)$ which allows the server to easily detect duplicates. When the server receives a new ciphertext, it discards the file if the tag equals one already in its database. Following this protocol, several schemes have been given, focussing mainly on improving efficiency e.g. [10,7,18].

However, in [4], the authors point out the lack of a formal security investigation of this emerging model. They formally introduce the concept of *message-locked encryption* (MLE) and provide a complete security analysis. In particular, they show that a secure MLE does not need to be deterministic to achieve its goal. It is sufficient (and more general) to provide an equality testing procedure that publicly checks if two ciphertexts encrypt the same plaintext, as shown in [1]. The interactive case has recently been studied in [3].

Security. As other kinds of "searchable encryption", MLE stands at the boundary of deterministic and probabilistic encryption worlds. As such, it cannot provide the standard notions of semantic security. Likewise, security can only be achieved for unpredictable data. If one can guess a possible message, one can encrypt it and then easily test ciphertexts for equality. In previous works the following privacy properties (PRV for short) were defined. The first one is $\mathsf{PRV\$\text{-}CDA}$ [4] that states that the semantic security should hold when messages are unpredictable (having high min-entropy), even for an adversary being able to choose the distribution where the messages are drawn (hence the notion of CDA, for Chosen Distribution Attack). In this experiment, the adversary has to distinguish a ciphertext according to a distribution of its choice from a random bit sequence. The second one is $\mathsf{PRV\text{-}CDA2}$ [1] that adds the parameter dependence setting, for which the security should hold even for messages that depend on the public parameters. They are then given to the adversary who chooses a distribution. Abadi et al. [1] have also slightly modified the security experiment, compared to [4], introducing a real-or-random oracle that gives to the encryption algorithm either a set of (unpredictable) messages drawn from the adversary's chosen distribution, or a true randomly chosen set of (unpredictable) messages. The adversary has to distinguish between both cases. Additionally to these indistinguishability-type properties, the authors in [4] introduce the natural requirement of *tag consistency*, whose goal is to make it impossible to undetectably replace a message by a fake one. It states that if two tags are equal, then the underlying messages should be equal.

Our contributions. In this article, we investigate the converse: if two messages are equal, does the server always perform deduplication? Strangely enough, in almost all previous CE and MLE schemes [8,4,3], it is straightforward for a user

to avoid the deduplication process altogether. The main feature for which those schemes were introduced is not achieved. In those schemes, the server does not actually verify that the key has actually been computed as required.

In this context, we have three contributions in this paper. First, we formalize this new security requirement, namely *deduplication consistency*, second we provide a generic transformation of a non-deduplication consistent MLE scheme into a deduplication consistent one and third we give a ElGamal-based construction.

In order to achieve verifiability in MLE, we introduce a new notion of *deduplication consistency*. It states that an equality test run on two valid ciphertexts with the same underlying plaintext will output 1 with overwhelming probability. Verifiability is a classical notion to prevent denial-of-service attacks, but this can be also useful in many scenarios. A court could oblige a cloud service provider to delete *all* copies of a given file, for example a newspaper article (right-to-be-forgotten) or a media file (for copyright infringement). If users are able to escape the deduplication process, the cloud service provider would not be able to prove that he complied to the court decision. A different scenario could be a collaborative database. Some attributes need to have a unique value in each row. If two users want to upload the same information, then the database would want to enforce deduplication.

A natural way to provide the verifiability of a ciphertext in a scheme of e.g. [8], is to provide a NIZK proof that the key $K = H(M)$ is correctly computed from the message M, and the ciphertext $C = \mathsf{SE.Encrypt}(M, K)$ is also consistent w.r.t. the same message M and key K. Based on this, we propose a generic construction to turn any MLE scheme into a deduplication consistent scheme.

To instantiate this generic scheme, we rely on an ElGamal-based construction. Indeed, another important issue in cloud storage is efficiency, as people usually expect instant uploading and responses from the cloud storage provider. Moreover, the ciphertexts' expansion should be carefully controlled, as the deduplication main goal is to save space storage. As such, neither generic non-interactive zero-knowledge proof (NIZK) used in [1] nor fully homomorphic encryption used in [3] could be considered as acceptable solutions. Combining an ElGamal encryption wih an algebraic hash funcion makes possible to use efficient NIZK over discrete logarithm relation sets [16] to prove that these computations are all consistent one with each other. Our construction is the first one that provably achieves deduplication consistency. As this is a strong security requirement, our scheme is far less efficient than convergent-like solutions, but it is still more efficient than those of Abadi et al. [1] or those of Bellare and Keelveedhi [3] whose goal is also to strengthen security.

Organization of the paper. The paper is now organized as follows. In the next section, we provide some background that will be useful all along the paper. In Section 3, we give the security model for message-locked encryption. Section 4 introduces our new notion of deduplication consistency and the generic construction and Section 5 describes the ElGamal-based construction. Finally, in Section 6, we provide the security proofs and discuss efficiency.

2 Preliminaries

2.1 Bilinear Groups

Our construction relies on pairings, so we recall the definition of *bilinear groups* that are a set of three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p together with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that (i) for all $X_1 \in \mathbb{G}_1, X_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(X_1^a, X_2^b) = e(X_1, X_2)^{ab}$, (ii) for $X_1 \neq 1_{\mathbb{G}_1}$ and $X_2 \neq 1_{\mathbb{G}_2}$, $e(X_1, X_2) \neq 1_{\mathbb{G}_T}$, and (iii) e is efficiently computable. We use *type 3* pairings for which there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$.

2.2 Computational Assumptions

Our construction security relies on the following computational assumptions.

Assumption 1 (blinded-DDH (bl-DDH)) *Given $(t_1, t_1^u, g_1, g_1^r, g_1^z) \in \mathbb{G}_1^5$ and $(t_2, t_2^{u \cdot k}) \in \mathbb{G}_2^2$ for random $(u, r, k) \in (\mathbb{Z}_p^*)^3$, it is hard to decide whether $z = r \cdot k$ or z is a random element from $\mathbb{Z}_p^*$. We define $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{bl\text{-}DDH}}(\lambda)$ as the advantage of a polynomial-time adversary $\mathcal{A}$ against* bl-DDH.

In the security proof of our scheme, we use the following generalization of the bl-DDH assumption. We stress that $(1,1)$-bl-DDH is the bl-DDH assumption.

Assumption 2 ((T,ℓ)-blinded-DDH ((T,ℓ)-bl-DDH)) *Let T and ℓ be two integers. Let $\left[u_h, \{r_{i,h}\}_{i=1}^{\ell}, k_h\right]_{h=1}^{T}$ be random in $(\mathbb{Z}_p^*)^{T(\ell+2)}$. Given $(t_1, g_1, \ldots, g_\ell)$ in $\mathbb{G}_1^{\ell+1}$, t_2 in $\mathbb{G}_2$, $\left[t_1^{u_h}, \{g_i^{r_{h,i}}\}_{i=1}^{\ell}, \{g_i^{z_{h,i}}\}_{i=1}^{\ell}\right]_{h=1}^{T} \in \mathbb{G}_1^{T(2\ell+1)}$ and $\left[t_2^{u_h \cdot k_h}\right]_{h=1}^{T} \in \mathbb{G}_2^T$, for all $h = 1, \ldots, T$ and for all $i = 1, \ldots, \ell$, it is hard to decide whether $z_{h,i} = r_{h,i} \cdot k_h$ or z is a random element from $\mathbb{Z}_p^*$. We define $\mathsf{Adv}_{\mathcal{A}}^{(T,\ell)\text{-}\mathsf{bl\text{-}DDH}}(\lambda)$ as the advantage of a polynomial-time adversary $\mathcal{A}$ against (T,ℓ)-bl-DDH.*

To assert the strength of our hypothesis, we prove the following reduction to the tripartite decisional Diffie-Hellman assumption [5].

Theorem 1. *The blinded-DDH assumption is polynomially reducible to the tripartite Diffie-Hellman assumption. The (T,ℓ)-blinded-DDH assumption is polynomially reducible to the blinded-DDH assumption.*

2.3 Commitment Schemes

A commitment scheme aims at masking a secret while allowing a later revelation.

Generic Description. Formally, those schemes are made up with three algorithms, namely the Setup which on input a security parameter λ outputs the public parameters pp, the Commit which on input pp and a message M outputs a commit C and a witness r, and the Open which on input a commit C, a message M and a witness r outputs 1 if C is a commitment of M for the witness r, and 0 otherwise. A commitment scheme Γ is considered to be cryptographically

secure if it verifies both the *hiding* and the *binding* properties. We focus on the latter as it is the one that matters the most in our constructions and proofs.

The commitment binding experiment $\mathsf{Exp}^{\mathsf{binding}}_{\Gamma,\mathcal{A}}(\lambda)$ starts by executing the Setup to obtain pp. On input pp, the adversary $\mathcal{A}$ outputs a commitment C, and two pairs (M, r) and (M', r'). The experiment outputs 1 iff $\mathsf{Open}(C, M, r) = \mathsf{Open}(C, M', r') = 1$. We say that the commitment scheme Γ is binding if for all polynomial time adversaries $\mathcal{A}$ the following advantage is negligible for all λ: $\mathsf{Adv}^{\mathsf{binding}}_{\Gamma,\mathcal{A}}(\lambda) = \Pr\left(\mathsf{Exp}^{\mathsf{binding}}_{\Gamma,\mathcal{A}}(\lambda) = 1\right)$.

The Pedersen Commitment. Let $\mathbb{G}$ be a group of prime order p and let g and h be generators of $\mathbb{G}$. The Pedersen Commitment [13] allows a prover to commit on a secret value $m \in \mathbb{Z}_p$. During the Commit execution, one computes $C = g^m h^r$ with $r \in \mathbb{Z}_p$ picked uniformly at random to the verifier. The Open algorithm is straightforward. This commitment scheme is perfectly hiding and is computationally binding under the discrete logarithm assumption.

2.4 Non-interactive Zero-Knowledge (NIZK) Proofs

We use NIZK proofs of membership in NP languages to achieve the notion of deduplication consistency that we introduced.

Generic Description. Let Π be a proof system in NP languages for a NP-relation rel. Such a system is given by two probabilistic polynomial-time machines $\mathcal{P}$ and $\mathcal{V}$ such that (i) for all $(y, w) \in \mathsf{rel}$ (that is $\mathsf{rel}(y, w) = \mathsf{true}$), $\mathcal{P}$ takes as input (y, w) and $\mathcal{V}$ takes as input y and (ii) at the end of the protocol between $\mathcal{P}$ and $\mathcal{V}$, $\mathcal{V}$ outputs a bit d of acceptance ($d = 1$) or rejection ($d = 0$). We require the following properties: (i) *Completeness* for all $(y, w) \in \mathsf{rel}$, $\mathcal{V}$ returns 1 with probability 1; and (ii) *Soundness*: for all $y \in \{0,1\}^*$, if $\mathcal{V}$ returns 1, then it exists w such that $(y, w) \in \mathsf{rel}$ with overwhelming probability.

We also need this proof system to be *zero-knowledge*. This means that a distinguisher $\mathcal{D}$ cannot distinguish between the proofs produced by a real prover or the ones produced by a simulator. We denote $\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{D}}(\lambda)$ the advantage of an adversary $\mathcal{A}$ in this distinguishing game, and we say that a non-interactive proof system $(\mathcal{P}, \mathcal{V})$ is (statistically) zero-knowledge if there exists a polynomial time simulator sim such that for any polynomial time distinguisher $\mathcal{D}$, the function $\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{D}}(\lambda)$ is negligible.

Double Discrete Logarithms Proofs. For our ElGamal-based scheme, the NIZK proofs we use are conjunctions of classical discrete logarithm relations [16]. They are made non-interactive thanks to the Fiat-Shamir transform [9], proven to be secure in the random oracle model [14]. The main time-consuming part is a double logarithm NIZK proof (with a statistical zero-knowledge property, as shown in [17]), where the statement has the form: $\mathsf{NIZK}\left(s : V_1 = h^{x^s} \wedge V_2 = y^s\right)$, where $h, y \in \mathbb{Z}_p$ and $x \in \mathbb{Z}_p^*$ are public, while $s \in \mathbb{Z}_p^*$ is secret.

2.5 Hashing Block Sources

Message-locked encryption, like deterministic encryption, can only protect messages that are hard to guess. To model this fact, Bellare et al. propose in [2] a definition of privacy, which states that no information about multiple dependent messages is leaked from their encryptions. Though unpredictable, the adversary $\mathcal{A}$ can choose the distribution over the messages, allowing them to be correlated. In order to avoid brute force attacks, the distribution of messages should guarantee a minimal entropy of the messages. The *min-entropy* of a random variable X is defined as $H_\infty(X) = -\log(\max_x(\Pr[X = x]))$. A random variable X such that $H_\infty(X) \geq \mu$ is a *μ-source*. A *(T, μ)-source* is a random variable $\mathbf{X} = (X_1, \ldots, X_T)$ where each X_i is a μ-source. A *(T, μ)-block source* is a random variable $\mathbf{X} = (X_1, \ldots, X_T)$ where each $X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}$ is a μ-source.

One of the crucial point in our construction of MLE is the hashing of such block sources. The Leftover Hash Lemma [12] is a classic tool for extracting random-looking strings from an uncertain source of entropy. Precise and tight results from [6] will help us to prove the privacy of our MLE when the keys are derived from messages produced from a block source. The following theorem from [6] states that if H is a 2-universal hash function applied to some elements of a block source $(X_1, \ldots, X_T)$, the distribution $(H, H(X_1), \ldots, H(X_T))$ is close to the uniform distribution. Let us recall that a family $\mathcal{H}$ of hash functions mapping $\{0,1\}^n$ to $\{0,1\}^\ell$ is said to be 2-*universal* if for all distinct x and y, the probability that $H(x) = H(y)$ equals $\frac{1}{2^\ell}$, when H is drawn at random.

Theorem 2. *[6, Theorem 3.5] Let $H : \{1, \ldots, 2^n\} \to \{1, \ldots, 2^m\}$ be a random hash function from a 2-universal family $\mathcal{H}$. Let $\mathbf{X} = (X_1, \ldots, X_T)$ be a (T, μ)-block source over $\{1, \ldots, 2^n\}^T$. For every $\varepsilon > 0$ such that $\mu > m + \log(T) + 2\log(1/\varepsilon)$, the hashed sequence $(H, \mathbf{Y}) = (H, H(X_1), \ldots, H(X_T))$ is ε-close to uniform in $\mathcal{H} \times \{1, \ldots, 2^m\}^T$.*

3 Message-Locked Encryption: Definition and Security

There are two different definitions for message-locked encryption (MLE) in the literature. The first one is due to Bellare, Keelveedhi and Ristenpart [4] and the second one from Abadi, Boneh, Mironov, Raghunathan and Segev [1]. Our definition, as well as the security model, is based on the ones from [1]. This definition is more general than Bellare et al.'s, but makes the notion of tag less present. In [4], the tag generation is performed only from the ciphertext, and the tag correctness ensures that one message encrypted by two different users will have the same tag, so that a server will be able to remove one of the ciphertexts. Abadi et al.'s definition of MLE (denoted MLE2) introduces a validity test to check the validity of ciphertext, and an equality test to deduplicate redundant files. This allows to handle randomized tags instead of deterministic. Even though there is no tag generation anymore, the security notion of tag consistency is kept, and we will sometimes informally call "tags" the parts of the ciphertext that are involved in the equality test.

3.1 Syntactic Definition

A Message-Locked Encryption (MLE) scheme is defined by the six following algorithms (PPGen, KD, Enc, Dec, EQ, Valid) operating over the plaintext space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and keyspace $\mathcal{K}$, with λ as a security parameter.

- The parameter generation algorithm PPGen takes as input 1^λ and returns the public parameters $\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$.
- The key derivation function KD takes as input the public parameters pp, a message M, and outputs a message-derived key $k_M \leftarrow \mathsf{KD}(\mathsf{pp}, M)$.
- The encryption algorithm Enc takes as input pp, a message-derived key k_M, and a message M. It outputs a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{pp}, k_M, M)$.
- The decryption algorithm Dec takes as input pp, a secret key k_M, a ciphertext c and outputs either a message M or $\perp : \{M, \perp\} \leftarrow \mathsf{Dec}(\mathsf{pp}, k_M, c)$.
- The equality algorithm EQ takes as input public parameters pp, and two ciphertexts c_1 and c_2 and outputs 1 if both ciphertexts are generated from the same underlying message, and 0 otherwise: $\{0, 1\} \leftarrow \mathsf{EQ}(\mathsf{pp}, c_1, c_2)$.
- The validity-test algorithm Valid takes as input public parameters pp and a ciphertext c and outputs 1 if the ciphertext c is a valid ciphertext, and 0 otherwise: $\{0, 1\} \leftarrow \mathsf{Valid}(\mathsf{pp}, c)$.

A message-locked encryption is said to be *correct* if for all $\lambda \in \mathbb{N}$, all $\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$, all message $M \in \mathcal{M}$ and all $c \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), M)$,
(i) $M = \mathsf{Dec}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), c)$ and $\mathsf{Valid}(\mathsf{pp}, c) = 1$, and
(ii) $\mathsf{EQ}(\mathsf{pp}, \mathsf{Enc}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), M; \omega), \mathsf{Enc}(\mathsf{pp}, \mathsf{KD}(\mathsf{pp}, M), M; \omega')) = 1$.

This last property is equivalent to tag correctness in [4] (and we explicitely wrote the randomness ω and ω' to recall that encryption is probabilistic). To avoid trivial solutions, we recall that keys kept for decryption must be shorter than the message. As mentioned in [4], there must exist constants $c, d < 1$ such that the function that on input $\lambda \in \mathbb{N}$ returns $\max_{\mathsf{pp}, M} (\Pr[|\mathsf{KD}(\mathsf{pp}, M)| > d|M|^c])$ is negligible, where $\mathsf{pp} \in \mathsf{PPGen}(1^\lambda)$ and $M \in \mathcal{M}$.

3.2 Privacy

The main security requirement for message-locked encryption is privacy of unpredictable messages. No MLE scheme can provide indistinguishability for predictable messages (drawn for a polynomial-size space), because of the equality testing procedure EQ. Our privacy notion is a combination of those existent. We rely on a $\mathsf{PRV\text{-}CDA2}$-like requirement [1], however, like in [4], our scheme does not achieve the privacy property when the message distribution chosen by the adversary depends on the public parameters. Therefore, we call our privacy property $\mathsf{PRV\text{-}piCDA}$, for *Privacy under parameter independent Chosen Distribution Attack*. It captures privacy of messages that are unpredictable but independent of the public parameters. Let us first define the real-or-random encryption oracle.

Definition 1 (Real-or-Random encryption oracle). *The real-or-random encryption oracle takes as input pairs* $(\mathsf{mode}, \mathbb{M})$ *with* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$*, and*

$\mathbb{M}$ *a polynomial size circuit representing a joint distribution over* T *messages from* $\mathcal{M}$. *If* $\mathsf{mode} = \mathsf{real}$ *then the oracle samples* $(M_1, \ldots, M_T) \leftarrow \mathbb{M}$ *and if* $\mathsf{mode} = \mathsf{rand}$ *then the oracle samples* T *uniform and independent messages from* $\mathcal{M}$. *Then the oracle outputs a ciphertexts vector* $\mathbf{C} = (c_1, \ldots, c_T)$ *such that, for each* i *the oracle computes* $k_{M_i} = \mathsf{KD}(pp, M_i)$ *and* $c_i = \mathsf{Enc}(pp, k_{M_i}, M_i)$.

Following [15], we consider adversaries whose restriction on their queries is that they are samplable by a polynomial-size circuit in the random oracle model.

Definition 2 (q-query polynomial-sampling complexity adversary). *We consider* (T, μ)*-block source. Let* $\mathcal{A}(1^\lambda)$ *be a probabilistic polynomial-time algorithm that is given an oracle access to* $\mathsf{RoR}(\mathsf{mode}, pp, \cdot)$ *for some* $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$. *Then,* $\mathcal{A}$ *is a* q*-query* (T, μ)*-source adversary if, for each of* $\mathcal{A}$*'s RoR-queries* $\mathbf{M}$, *it holds that* $\mathbf{M}$ *is a* (T, μ)*-block source that is samplable by a polynomial-size circuit that uses at most* q *queries to the random oracle.*

Informally, PRV-piCDA security notion requires that encryptions of random messages should be indistinguishable from encryptions of messages drawn from a (T, μ)-block source.

Definition 3 (PRV-piCDA Security). *An MLE scheme* Π *is* (T, μ)*-block source* PRV-piCDA *secure if, for any probabilistic polynomial-time* (T, μ)*-block source adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, *there exists a negligible function* $\nu(\lambda)$ *such that:*

$$\mathsf{Adv}^{\mathsf{PRV-piCDA}}_{\Pi,\mathcal{A}}(\lambda) = \left| Pr\left[\mathbf{Exp}^{\mathsf{real}}_{\Pi,A} = 1\right] - Pr\left[\mathbf{Exp}^{\mathsf{rand}}_{\Pi,A} = 1\right]\right| \leq \nu(\lambda),$$

where the game $\mathbf{Exp}^{mode}_{\Pi,\mathcal{A}}(\lambda)$ *is defined Figure 1.*

Experiment $\mathbf{Exp}^{\mathsf{mode}}_{\Pi,\mathcal{A}}(\lambda)$
$\mathbb{M} \leftarrow \mathcal{A}_0(1^\lambda)$;
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$;
$b \leftarrow \mathcal{A}_1^{\mathsf{RoR}(\mathsf{mode},\mathbb{M})}(1^\lambda, \mathsf{pp})$
Return b.

Fig. 1. PRV-piCDA Game: $\mathbf{Exp}^{\mathsf{mode}}_{\Pi,\mathcal{A}}(\lambda)$

Experiment $\mathbf{Exp}^{TC}_{\Pi,\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$;
$(M_0, c_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
If $\mathsf{Valid}(\mathsf{pp}, c_1) = 0$ or $M_0 = \perp$ return 0;
$k_M \leftarrow \mathsf{KD}(\mathsf{pp}, M_0)$;
$c_0 \leftarrow \mathsf{Enc}(\mathsf{pp}, k_M, M_0)$;
$M_1 \leftarrow \mathsf{Dec}(\mathsf{pp}, k_M, c_1)$;
If $(\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 1) \wedge (M_0 \neq M_1) \wedge (M_1 \neq \perp)$ return 1;
Else return 0.

Fig. 2. Tag Consistency Game : $\mathbf{Exp}^{TC}_{\Pi,\mathcal{A}}(\lambda)$

As stated in [1, Theorem 4.6], the case where $\mathcal{A}$ can query the RoR oracle multiple times is equivalent to the case where $\mathcal{A}$ can query this oracle just once.

3.3 Tag Consistency

Tag consistency is a major requirement of any MLE scheme. It ensures that the server will not discard a file if it doesn't have another file encrypting the same plaintext. We use the game defined by Abadi et al in [1].

Definition 4 (Tag consistency). *An MLE scheme Π is* tag consistent *if for any probabilistic polynomial-time $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that:*

$$\mathsf{Adv}^{TC}_{\Pi,\mathcal{A}}(\lambda) = Pr\left[\mathbf{Exp}^{TC}_{\Pi,\mathcal{A}} = 1\right] \leq \nu(\lambda),$$

where the random experiment $\mathbf{Exp}^{TC}_{\Pi,\mathcal{A}}(\lambda)$ is described in Figure 2.

4 Deduplication Consistency

In precedent works, the main security requirement, besides privacy, was tag consistency, meaning that if the equality test $\mathsf{EQ}(c_1, c_2)$ outputs 1 on two ciphertexts, then the underlying plaintexts are the same. As sketched in the introduction, we tackle here the converse case: if two ciphertexts c_1 and c_2 are meant to encrypt the same message, we require that $\mathsf{EQ}(c_1, c_2)$ outputs 1 with overwhelming probability. To capture such a security issue, we introduce in the following a new security notion for message-locked encryption, called *deduplication consistency*. This notion ensures that a MLE scheme *provably* provides deduplication.

4.1 Overview

The main point of deduplication consistency is to make a MLE scheme *verifiable*. In fact, if a server makes use of an MLE scheme for which it cannot be convinced that deduplication is actually enforced, he will loose the benefit he has expected from deduplication. In most existing schemes indeed (see below), only users are responsible for a smooth deduplication process. Then these schemes can easily be "deviate[d] from [their] prescribed functionality"[3].

In addition to save space storage, verifiable deduplication is a functionality that can have an interest of its own. Today, a really hot topic is the right-to-be-forgotten. An important question related to this topic is how a server can prove that he really deleted some given files. The problem is even more difficult if the files are encrypted on the server: the right to privacy of a user cannot prevail over the right to privacy of other users. It can happen however that a court asks a cloud service provider to remove a defamatory newspaper article or video from its storage space. Then the server's manager could encrypt this specific file with a verifiable MLE scheme and match it against the other files in the server. If the equality test returns one, deleting the corresponding file will be sufficient to prove that no user can now access to this file, as no user can bypass the deduplication procedure.

[3] Oded Goldreich, The Foundations of Cryptography, Preface.

4.2 Formal Definition

We define the deduplication experiment $\mathbf{Exp}^{DC}_{\Pi,\mathcal{A}}(1^\lambda)$ described on Figure 3.

Definition 5 (Deduplication consistency). *An MLE scheme Π is* deduplication consistent *if for any probabilistic polynomial-time $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that:*

$$\mathsf{Adv}^{DC}_{\Pi,\mathcal{A}}(\lambda) = Pr\left[\mathbf{Exp}^{DC}_{\Pi,\mathcal{A}} = 1\right] \leq \nu(\lambda),$$

where the random experiment $\mathbf{Exp}^{DC}_{\Pi,\mathcal{A}}(1^\lambda)$ is described in Figure 3.

Experiment $\mathbf{Exp}^{DC}_{\Pi,\mathcal{A}}(\lambda)$
$\mathsf{pp} \leftarrow \mathsf{PPGen}(1^\lambda)$;
$(M, c_0, c_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$;
If $(\mathsf{Valid}(\mathsf{pp}, c_0) = 0) \vee (\mathsf{Valid}(\mathsf{pp}, c_1) = 0)$ then return 0;
If $\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 1$ then return 0;
$k_M \leftarrow \mathsf{KD}(\mathsf{pp}, M)$;
$M_0 \leftarrow \mathsf{Dec}(\mathsf{pp}, k_M, c_0)$; $M_1 \leftarrow \mathsf{Dec}(\mathsf{pp}, k_M, c_1)$;
If $M \neq M_0 \vee M \neq M_1$ then return 0;
Return 1;

Fig. 3. Deduplication Security Game : $\mathbf{Exp}^{DC}_{\Pi,\mathcal{A}}(\lambda)$

As for previous schemes, none of them formalizes this notion. Moreover, it is obvious that the different solutions given by Bellare et al. [4] do not achieve deduplication consistency. We have the intuition that the randomized scheme proposed by Abadi et al. [1] is deduplication consistent due to the presence of the NIZK, but a formal proof remains an open problem.

4.3 A Generic Construction

We first describe a generic construction permitting to transform any private and tag consistent MLE scheme Λ into a MLE scheme Λ' additionally achieving the deduplication consistency.

Our construction. For this purpose, we need a secure commitment scheme Γ and a NIZK proof system Π (see Section 2). Our scheme is described as follows.

- PPGen. This step executes (i) $\Lambda.\mathsf{PPGen}$ which outputs $\Lambda.\mathsf{pp}$, (ii) $\Gamma.\mathsf{Setup}$ which gives $\Gamma.\mathsf{pp}$ and (iii) the generation of a reference string R for the NIZK proof. Then, $\Lambda'.\mathsf{pp} = (\Lambda.\mathsf{pp}, \Gamma.\mathsf{pp}, R)$.
- KD. On input $\Lambda'.\mathsf{pp}$ and a message M, this algorithm simply corresponds to the execution of $\Lambda.\mathsf{KD}$, which outputs $k_M = \Lambda.\mathsf{KD}(\Lambda.\mathsf{pp}, M)$.

– Enc. There are three steps during the encryption algorithm. At first, we execute the $\Lambda.\mathsf{Enc}(\Lambda.\mathsf{pp}, k_M, M)$ which outputs c. Then, we compute a commitment on M, as $(C, r) = \Gamma.\mathsf{Commit}(\Gamma.\mathsf{pp}, M)$. Finally, we provide the following NIZK proof:

$$\pi = \mathsf{NIZK}\Big(M, r : c = \Lambda.\mathsf{Enc}(\Lambda.\mathsf{pp}, \Lambda.\mathsf{KD}(\Lambda.\mathsf{pp}, M), M) \\ \wedge (C, r) = \Gamma.\mathsf{Commit}(\Gamma.\mathsf{pp}, M)\Big)$$

  and the output ciphertext is $c' = (c, C, \pi)$.
– Valid. On input $c' = (c, C, \pi)$, this algorithm executes $\Lambda.\mathsf{Valid}(\Lambda.\mathsf{pp}, c)$ and verifies that the NIZK π is correct.
– Dec. On input $c' = (c, C, \pi)$ and k_M, this algorithm first executes Valid, and, if it returned 1, it executes $\Lambda.\mathsf{Dec}(\Lambda.\mathsf{pp}, k_M, c)$ to obtain M.
– EQ. On input $c'_1 = (c_1, C_1, \pi_1)$ and $c'_2 = (c_2, C_2, \pi_2)$, this algorithm first executes Valid on both ciphertexts, and then, if both returned 1, it executes $\Lambda.\mathsf{EQ}(\Lambda.\mathsf{pp}, c_1, c_2)$.

Security. Regarding the security of the above construction, it verifies the privacy and the tag consistency properties. This is mainly due to the fact that the addition of the commitment and the NIZK proof does not affect the security proofs related to both privacy and tag consistency, for obvious reasons.

More precisely, regarding the privacy property, the NIZK proof can be simulated during the experiment (by the real-or-random oracle), using the zero-knowledge property. We have a slight loss in security, corresponding to the advantage of the adversary against the hiding property of the used commitment scheme Γ. Regarding the tag consistency property, this is similarly obvious.

The deduplication consistency is given by the following theorem.

Theorem 3. *The scheme Λ' given above is deduplication-consistent, assuming that Λ is tag consistent, Γ is binding, and π is a sound zero-knowledge proof.*

Proof. Our aim in this proof is to reduce the deduplication consistency of our construction to the binding property of the commitment scheme. We consider the commitment binding experiment given in Section 2 and play the role of an adversary $\mathcal{A}$ against it. $\mathcal{A}$ has the parameters $\Gamma.\mathsf{pp}$ of the scheme Γ. His aim is to output a commitment C and two pairs (M, r) and (M', r') such that $\mathsf{Open}(C, M, r) = \mathsf{Open}(C, M', r') = 1$. We assume that $\mathcal{A}$ has access to an adversary $\mathcal{B}$ which has a non-negligible advantage against the deduplication consistency experiment of our scheme.

Parameter generation. We first generate the parameter of the MLE system, executing $\Lambda.\mathsf{PPGen}$, generating a reference string R for the NIZK proof, and adding the commitment parameter $\Gamma.\mathsf{pp}$ obtained above. We then give $(\Lambda.\mathsf{pp}, \Gamma.\mathsf{pp}, R)$ to the adversary $\mathcal{B}$.

Adversary's answer. At any time of the experiment, using its advantage against deduplication consistency, $\mathcal{B}$ outputs (M, c'_0, c'_1), with $c'_0 = (c_0, C_0, \pi_0)$ and $c'_1 = (c_1, C_1, \pi_1)$, such that all the conditions related to the deduplication

consistency experiment are verified. In particular, we have $\Lambda'.\mathsf{Valid}(\mathsf{pp}, c'_0) = 1$, $\Lambda'.\mathsf{Valid}(\mathsf{pp}, c'_1) = 1$ and $\Lambda.\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 0$.

Answer to the commitment challenge. Using the soundness property of the NIZK, $\mathcal{A}$ is then able to extract, from π_0 and π_1, the underlying secret messages M'_0 and M'_1, related to c_0 and c_1 respectively. As we have $\Lambda.\mathsf{EQ}(\mathsf{pp}, c_0, c_1) = 0$, and since Λ is tag consistent, we necessarily have $M'_0 \neq M'_1$.

As $\mathcal{B}$ is successful in the DC experiment, and using c_0 and c_1 respectively, $\mathcal{A}$ computes $k_M = \Lambda.\mathsf{KD}(\mathsf{pp}, M)$, and then $M_0 = \Lambda.\mathsf{Dec}(\mathsf{pp}, k_M, c_0)$ and $M_1 = \Lambda.\mathsf{Dec}(\mathsf{pp}, k_M, c_1)$, with $M = M_0 = M_1$.

It means that it exists $i \in \{0, 1\}$ such that $M_i \neq M'_i$. As both ciphertexts are valid, for both M_i and M'_i, there is a sound zero-knowledge proof provided by $\mathcal{B}$ from which $\mathcal{A}$ is able to extract the corresponding witnesses r_i and r'_i. $\mathcal{A}$ sends $C_i, (M_i, r_i), (M'_i, r'_i)$ to the binding challenger. It is a winning output for the binding experiment of Γ, with an advantage at least equal to the one of $\mathcal{B}$ against deduplication consistency, which concludes the proof.

5 A Concrete Message Locked Encryption with Deduplication Consistency

In this section, we describe our construction of a deduplication consistent MLE. Compared to the fully randomized message-locked encryption from [1], the main difference is that the secret key is derived from the message using a hash function which has algebraic properties. Thus, we avoid generic NIZK [11], gaining efficiency. More precisely, the message M is cut into small blocks $(m_1 \| \ldots \| m_\ell)$ of ρ bits, and the key is computed as $k_M = \prod_{i=1}^{\ell} a_i^{m_i} \mod p$ for publicly known a_i's. By using Theorem 2, we prove that if the messages come from a source with high enough min-entropy, the key k_M is indistinguishable from a uniform key.

These blocks m_i are chosen small enough to be efficiently decrypted, as we encrypt them using the ElGamal encryption with messages in the exponent, and the key k_M: $T_{1,i} = g_i^{r_i}$ and $T_{2,i} = h^{m_i} \cdot g_i^{r_i \cdot k_M}$. In order to achieve DC, those equations should be included in the NIZK proof.

It remains to create a suitable tag, which is done by using the same technique as in [1]. More precisely, we provide a pair $(\tau_1 = t_1^u, \tau_2 = t_2^{u \cdot k_M})$, which will make it possible to detect a duplication using a pairing computation. We add the following relations to the NIZK proof: $\tau_1 = t_1^u$ and $e(\tau_1, t_2)^{k_M} = e(t_1, \tau_2)$.

We finally provide a Pedersen commitment C of the m_i's using a generator x of $\mathbb{Z}_p$ and a random s: $C = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s = k_M \cdot x^s \mod p$.

The main point regarding our NIZK is that we need to prove that the secret k_M (as an exponent for the groups $\mathbb{G}_1$ and $\mathbb{G}_T$) involved in the tag, the commitment and the ciphertexts is the same secret k_M as the one (as an element of the group $\mathbb{Z}_p^*$ of order p) computed from the message. Regarding the tag and the ElGamal ciphertext equations, the key k_M is seen as an exponent, and we can thus use standard and efficient ZK proofs *à la* Schnorr [16], making them non-interactive using the Fiat-Shamir heuristic [9].

The correctness of the commitment is easily proven. It remains to make the link between the message and the key. Equation $e(\tau_1, t_2)^{k_M} = e(t_1, \tau_2)$ can be rewritten as: $e(\tau_1, t_2)^C = e(t_1, \tau_2)^{x^s}$. Proving that this last equation is true involves the use of a double discrete logarithm. We use the techniques from [17] described in Section 2, which alter the efficiency of our construction.

Description

In this section, we formally describe our verifiable message-locked encryption Λ.

- PPGen. Let λ be the security parameter, and ℓ, ρ be integers. The parameter generation consists in generating a bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where p is a λ-bit prime, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative groups of same order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear asymmetric pairing. Let $t_1, \{g_i\}_{i=1,\ldots,\ell}, h$ be generators of $\mathbb{G}_1$ and t_2 be a generator of $\mathbb{G}_2$. We finally need $\ell + 1$ public elements $x, a_1, \ldots, a_\ell$ that generates $\mathbb{Z}_p^*$.
  $\mathsf{pp} = \{p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, t_1, \{g_i\}_{i=1,\ldots,\ell}, h, t_2, x, \{a_i\}_{i=1,\ldots,\ell}\}$.
- KD. On input public parameters pp and a message $M = (m_1 \| \ldots \| m_\ell)$ divided into ℓ blocks of ρ bits, it computes the key $k_M = \prod_{i=1}^{\ell} a_i^{m_i} \mod p$.
- Enc. On input public parameters pp, a message $M = (m_1 \| \ldots \| m_\ell)$ and a key k_M, the ciphertext is computed as follows:
  1. uniformly pick $u \in \mathbb{Z}_p^*$, compute $\tau = (\tau_1, \tau_2) = (t_1^u, t_2^{u \cdot k_M})$;
  2. uniformly pick $s \in \mathbb{Z}_p^*$ and compute a Pedersen commitment over the m_i's: $C = k_M \cdot x^s = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s \mod p$;
  3. for all $1 \leq i \leq \ell$, pick uniform and independent $r_i \in \mathbb{Z}_p^*$ and compute $T_{1,i} = g_i^{r_i}$ and $T_{2,i} = h^{m_i} \cdot g_i^{r_i \cdot k_M}$;
  4. compute the following non-interactive zero knowledge proof

$$\pi = \mathsf{NIZK}\Big(u, \{r_i\}_{i=1,\ldots,\ell}, M, k_M, s : \tau_1 = t_1^u \wedge e(\tau_1, t_2)^{k_M} = e(t_1, \tau_2)$$
$$\wedge\, T_{1,1} = g_1^r \wedge \cdots \wedge T_{1,\ell} = g_\ell^r \wedge T_{2,1} = T_{1,1}^{k_M} g^{m_1} \wedge \cdots \wedge T_{2,\ell} = T_{1,l}^{k_M} g^{m_\ell}$$
$$\wedge\, C = \prod_{i=1}^{\ell} a_i^{m_i} \cdot x^s \wedge e(\tau_1, t_2)^C = e(t_1, \tau_2)^{x^s}\Big).$$

  Finally output $c = (\tau, \{T_{1,i}, T_{2,i}\}_{i=1,\ldots,\ell}, C, \pi)$.
- Valid. On input a ciphertext $c = (\tau, \{T_{1,i}, T_{2,i}\}_i, C, \pi)$, this algorithm outputs 1 iff π is correct.
- Dec. On input pp, a key k_M and a valid ciphertext c, the procedure is:
  1. for all $i \in \{1, \ldots, \ell\}$ compute $h^{m_i} = T_{2,i} / T_{1,i}^{k_M}$ as in a standard ElGamal decryption procedure;
  2. for all $i \in \{1, \ldots, \ell\}$, retrieve the m_i with a discrete logarithm computation (this step is made possible by the choice of a small ρ);
  3. output $M = (m_1 \| \ldots \| m_\ell)$.

– EQ. On input pp and two valid ciphertexts $c = (\tau, \{T_{1,i}, T_{2,i}\}_i, C, \pi)$ and $\tilde{c} = (\tilde{\tau}, \{\tilde{T}_{1,i}, \tilde{T}_{2,i}\}, \tilde{C}, \tilde{\pi})$, parse τ as $\tau_1 = t_1^u$ and $\tau_2 = t_2^{u \cdot k}$ and $\tilde{\tau}$ as $\tilde{\tau}_1 = t_1^{\tilde{u}}$ and $\tilde{\tau}_2 = t_2^{\tilde{u} \cdot k}$. This algorithm outputs 1 iff $e(\tau_1, \tilde{\tau}_2) = e(\tilde{\tau}_1, \tau_2)$.

Correctness. Correctness is directly derived from the correctness of the ElGamal encryption scheme and properties of bilinear maps.

6 Security and Efficiency Arguments

6.1 Privacy

Theorem 4. *Let ε and μ be two non-zero positive reals, p a prime number and T,ℓ be integers such that $\mu > \log p + \log(T) + 2\log(1/\varepsilon)$. Our scheme Λ is* $\mathsf{PRV\text{-}piCDA}$ *secure for (T, μ)-block sources under the (T, ℓ)-*$\mathsf{bl\text{-}DDH}$ *assumption in the random oracle model.*

Sketch of Proof. As the inner product $\langle \cdot, \cdot \rangle : \left(\mathbb{Z}_p^\ell\right)^T \to \mathbb{Z}_p$ is a 2-universal hash function, we can apply Theorem 2: the keys extracted from the adversarially chosen (T, μ)-block source random variable $\mathbf{M} = (M_1, \ldots, M_T)$, will be indistinguishable from uniform. More precisely, if $\mu \geq \log(p) + \log(T) + 2\log(1/\varepsilon)$, the distribution of the keys is at distance ε from the uniform distribution in $\mathbb{Z}_p^\ell \times \mathbb{Z}_p^T$.

We construct a simulator $\mathcal{S}$ of the real-or-random encryption oracle against which q_H-query (T, μ)-block source polynomial-sampling complexity adversary $\mathcal{A}$ for the $\mathsf{PRV\text{-}piCDA}$ game has advantage exactly $\frac{1}{2}$, using a sequence of games.

Game G_0. This is the original game. We consider an adversary $\mathcal{A}$ able to break the $\mathsf{PRV\text{-}piCDA}$ security. In this game, $\mathcal{A}$ chooses a distribution $\mathbb{M}$ of the messages. She then queries the real-or-random oracle. Only after the query to this oracle, the public parameters of the scheme are generated.

The adversary $\mathcal{A}$ has access to a vector of T ciphertexts and she must return the value b' (real or random), matching how the plaintexts were generated by the real-or-random oracle. Let S_i be the event that $b = b'$ in game G_i. We have:

$$\mathsf{Adv}^{\mathsf{PRV-piCDA}}_{\Lambda,\mathcal{A}}(\lambda) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{real}}_{\Lambda,\mathcal{A}} = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{rand}}_{\Lambda,\mathcal{A}} = 1\right] \right| = 2 \left| \Pr(S_0) - \frac{1}{2} \right|.$$

Game G_1. In this game, $\mathcal{S}$ simulates the T non-interactive zero-knowledge proofs, using the random oracle, rather than computing them. The advantage of $\mathcal{A}$ against the zero-knowledge property of the NIZK proof is bounded by $\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}}(\lambda)$. Moreover, this simulation is computationally indistinguishable for $\mathcal{A}$ if there is no collision in the requests of the hash oracle. Let q_H the number of queries $\mathcal{A}$ makes to the random hash oracle.

$$|\Pr(S_0) - \Pr(S_1)| \leq \frac{q_H}{2^{\rho \ell T} p^{T(2+\ell)}} + \mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{A}}(\lambda).$$

Game G_2. We address the key generation. Instead of computing the keys with the KD procedure, $\mathcal{S}$ draws keys uniformly from $\mathbb{Z}_p$. From now on, the encryption key does not depend on the messages. From Theorem 2, $(k_{M_1}, \ldots, k_{M_T})$

is at distance ε from the uniform distribution in $\mathbb{Z}_p^T$, independently on how the messages were generated: $|\Pr(S_1) - \Pr(S_2)| \leq \varepsilon$.

Game G_3. With those simulated keys, the view of the adversary is exactly a (T, ℓ)-bl-DDH instance. Let $\mathcal{B}$ be an adversary against (T, ℓ)-bl-DDH, then for all $\mathcal{B}$, we have: $|\Pr(S_2) - \Pr(S_3)| \leq \mathsf{Adv}_{\mathcal{B}}^{(T,\ell)\text{-bl-DDH}}(\lambda)$.

Game G_4. In this game, $\mathcal{S}$ behaves as the real-or-random oracle, computing a Pedersen commitment over the m_i's. Thus we have $\Pr(S_3) = \Pr(S_4)$.

Moreover, $\mathcal{A}$'s advantage for breaking the indistinguishability of the Pedersen commitment is exactly $\frac{1}{2}$, as it is perfectly hiding. Then we have:

$$\left|\Pr(S_0) - \frac{1}{2}\right| \leq \mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{zk}}(\lambda) + \frac{q_H}{2^{\rho\ell T} p^{T(2+\ell)}} + \varepsilon + \mathsf{Adv}_{\mathcal{B}}^{(T,\ell)\text{-bl-DDH}}(\lambda),$$

and the probability for $\mathcal{A}$ to win the PRV-piCDA game is negligible. □

6.2 Tag Consistency and Deduplication Consistency.

Theorem 5. *Our scheme Λ is tag consistent as that the key derivation function is collision-free (the inner product is a 2-universal hash-function).*

The proof derives from the EQ procedure : the bilinearity property of the pairing implies that if two ciphertexts are considered duplicate, then the keys used to genrate them must be equal. Which means that $\mathcal{A}$ is able to find collisions for the key-derivation function with non-negligible probability.

Theorem 6. *As a Pedersen commitment is computationally binding, our scheme Λ is deduplication-consistent in the random oracle model.*

As our construction is an instantiation of the generic construction given Section 4, the proof of this theorem directly follows from the proof of Theorem 3.

6.3 Efficiency

As [1] and [3], we improve upon security of convergent encryption, resulting in a loss in efficiency. We are however obviously more efficient than [1] as it uses generic NIZK (for a hash function represented as a circuit) and than [3] as it uses several times a fully homomorphic encryption. But a complete comparison is difficult as the three schemes achieve completely different security properties.

The most time and space consuming steps of our construction are the NIZK proof computation (especially the double logarithm), and the decryption which requires ℓ small discrete logarithm computation.

Acknowledgment

This work is supported by the European Union SUPERCLOUD Project (H2020 Research and Innovation Program grant 643964 and Swiss Secretariat for Education, Research and Innovation contract 15.0091) and by ERC Starting Grant ERC-2013-StG-335086-LATTAC. The authors want to thank Benoit Libert, Olivier Sanders, Jacques Traoré and Damien Vergnaud for helpful discussions.

References

1. M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev. Message-locked encryption for lock-dependent messages. In *Advances in Cryptology - CRYPTO 2013*. LNCS, vol. 8042 pp. 374–391, 2013.
2. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology - CRYPTO 2007*. LNCS, vol.4622, pp. 535–552, 2007.
3. M. Bellare and S. Keelveedhi. Interactive message-locked encryption and secure deduplication. In *Public-Key Cryptography - PKC 2015*. LNCS, vol.9020, pp. 516–538, 2015.
4. M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In *Advances in Cryptology - EUROCRYPT 2013*. LNCS, vol.7881, pp. 296–312, 2013.
5. D. Boneh, A. Sahai, and B. Waters. Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys. In *Advances in Cryptology - EUROCRYPT 2006*. LNCS vol. 4004, pp. 573–592, 2006.
6. K. Chung and S. P. Vadhan. Tight bounds for hashing block sources. In *APPROX 2008 RANDOM 2008*. LNCS, vol.5171, pp. 357–370, 2008.
7. L. P. Cox, C. D. Murray, and B. D. Noble. Pastiche: Making backup cheap and easy. In *Proceedings of the 5th Symposium on Operating Systems Design and implementation*, OSDI '02. ACM, pp. 285–298, 2002
8. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pp. 617–624, 2002.
9. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO'86*. LNCS, vol.263, pp. 186–194, 1986.
10. The Flud backup system. http://flud.org.
11. J. Groth. Short non-interactive zero-knowledge proofs. In *Advances in Cryptology – ASIACRYPT 2010*. LNCS, vol.6477, pp. 341–358, 2010.
12. R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89. ACM, pp. 12–24, 1989.
13. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91*. LNCS, vol.576, pp. 129–140, 1991.
14. David Pointcheval, Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. In J. Cryptology 13(3), pages 361-396, 2000.
15. A. Raghunathan, G. Segev, and S. P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In *Advances in Cryptology - EUROCRYPT 2013*. LNCS, vol.7881, pp. 93–110, 2013.
16. C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology – CRYPTO'89*. LNCS, vol.435, pp. 239–252, 1989.
17. M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology – EUROCRYPT'96*. LNCS, vol.1070, pp. 190–199, 1996 .
18. Z. Wilcox-O'Hearn and B. Warner. Tahoe: The least-authority filesystem. In *4th ACM Workshop* StorageSS '08. ACM, pp. 21–26, 2008
19. G. Yang, C. H. Tan, Q. Huang, and D. S. Wong. Probabilistic public key encryption with equality test. In *Topics in Cryptology – CT-RSA 2010*. LNCS, vol.5985, pp. 119–131, 2010.