2016

Towards efficient fully randomized message-locked encryption

Tao Jiang
University of Wollongong, tj290@uowmail.edu.au

Xiaofeng Chen
Xidian University, isschxf@mail.sysu.edu.cn

Qianhong Wu
Beihang University, qhw@uow.edu.au

Jianfeng Ma
Xidian University

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

See next page for additional authors

Publication Details

Jiang, T., Chen, X., Wu, Q., Ma, J., Susilo, W. & Lou, W. (2016). Towards efficient fully randomized message-locked encryption. Lecture Notes in Computer Science, 9722 361-375. Melbourne, Australia Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I

Towards efficient fully randomized message-locked encryption

Abstract
Cross-user data deduplication will greatly reduce the storage cost of storage service provider. Motivated by secure data deduplication, Abadi et al. extended the work Message-Locked Encryption (MLE) and introduced the primitive of MLE2 with nice security properties. However, their fully randomized scheme (R-MLE2) requires the inefficient equality-testing algorithm to identify all duplicate ciphertexts. Thus, an interesting open problem is how to reduce the overhead of R-MLE2 and propose an efficient construction for R-MLE2. In this paper, we introduce a new primitive called μR-MLE2, which gives a partial positive answer to this open problem. Our main trick is to use the client-assistant way based on *static* or *dynamic* decision trees. The advantage gained from it is that by interacting with clients, the server will reduce the time complexity of deduplication equality test from linear time to efficient logarithmic time over the whole database items.

Keywords
randomized, message, locked, encryption, towards, efficient, fully

Disciplines
Engineering | Science and Technology Studies

Publication Details
Jiang, T., Chen, X., Wu, Q., Ma, J., Susilo, W. & Lou, W. (2016). Towards efficient fully randomized message-locked encryption. Lecture Notes in Computer Science, 9722 361-375. Melbourne, Australia Information Security and Privacy: 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part I

Authors
Tao Jiang, Xiaofeng Chen, Qianhong Wu, Jianfeng Ma, Willy Susilo, and Wenjing Lou

Towards Efficient Fully Randomized Message-Locked Encryption

Tao Jiang[1,4], Xiaofeng Chen[1], Qianhong Wu[2], Jianfeng Ma[1], Willy Susilo [3], and Wenjing Lou[4]

[1] State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China
[2] School of Electronic and Information Engineering, Beihang University, Beijing, China
[3] Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
[4] Department of Computer Science, Virginia Polytechnic Institute and State University, VA, USA
jiangt2009@gmail.com,xfchen@xidian.edu.cn,qianhong.wu@buaa.edu.cn, jfma@mail.xidian.edu.cn,wsusilo@uow.edu.au,wjlou@vt.edu

Abstract. Cross-user data deduplication will greatly reduce the storage cost of storage service provider. Motivated by secure data deduplication, Abadi et al. extended the work Message-Locked Encryption (MLE) and introduced the primitive of MLE2 with nice security properties. However, their fully randomized scheme (R-MLE2) requires the inefficient equality-testing algorithm to identify all duplicate ciphertexts. Thus, an interesting open problem is how to reduce the overhead of R-MLE2 and propose an efficient construction for R-MLE2. In this paper, we introduce a new primitive called μR-MLE2, which gives a partial positive answer to this open problem. Our main trick is to use the client-assistant way based on *static* or *dynamic* decision trees. The advantage gained from it is that by interacting with clients, the server will reduce the time complexity of deduplication equality test from linear time to efficient logarithmic time over the whole database items.

Keywords: Deduplication, convergent encryption, message-locked encryption, interactive protocol

1 Introduction

With the rapid growing of cloud storage service, such as cloud storage [8, 9, 17], encryption becomes an important technique for protecting the confidentiality of data. Although data encryption provides an important guarantee over the security and privacy of clients' data, it limits the manners of the accessibility and availability of the encrypted data. Thus, it is important to design efficient scheme to support secure and efficient computation outsourcing [5, 6] and storage outsourcing [7]. Data deduplication enables cloud data storage systems to find

and remove duplicate data without compromising its availability. The goal of data deduplication is to store more data in less space by storing and maintaining files (blocks in fine-grained deduplication manner) into a single copy, where the redundant copies of data are replaced by a reference to this copy. It means that data deduplication storage system could reduce the storage size of u clients, who share the same data copy m, from $\mathcal{O}(u \cdot |m|)$ to $\mathcal{O}(u + |m|)$ if some implementation-dependent constants are hidden. Also, clients do not need to upload their data to the cloud storage server when there has been one copy stored, which will not only greatly reduce the communication cost of clients and cloud server, but also save the network bandwidth.

When the data from different clients is encrypted with their private secret keys, it is difficult to conduct ciphertext data deduplication among clients. A secure cross-client deduplication scheme should enable a storage server to detect data deduplication over the data encrypted by different clients, while efficiently prevent the practical attacks [10, 16, 19] from poor deduplication scheme. Douceur et al. [7] proposed the first solution for secure and efficient data deduplication, and they call it *convergent encryption.* This idea promoted many significant applications, where various schemes [3, 12] are implemented or designed based on convergent encryption. Recently, Bellare et al. [4] define a new primitive, Message-Locked Encryption (MLE), which brought rigor to security deduplication, and captured various security aspects of MLE. Also, they constructed several schemes and provided some detailed analysis over them. To strengthen the notions of security by considering plaintext distributions depend on the public parameter, Abadi et al. [1] proposed two approaches (fully random scheme and deterministic scheme) that are secure even for lock-dependent message in realistic. It answered the question: Can message-locked encryption be secure for lock-dependent message? The tag randomization design makes the fully random scheme, R-MLE2 for short, satisfy the standard secure notion of data confidentiality. Also, the overhead in the length of the ciphertext is only additive and independent of the message length.

However, as the open problem described in [1], the R-MLE2 scheme is not efficient in the deduplication process because of the comparison of the randomized tag introduced. It is important to maintain tags for sub-linear deduplication time, since for large data sets linear scans are prohibitive, particularly if they involve a linear number of cryptographic operations. In this paper, we ask whether the R-MLE2 scheme can be much more efficient (with logarithmic or nearly logarithmic deduplication test overhead) in data deduplication for large database while also keep the security properties of the deduplication scheme? We adopt client-server interaction based on random decision tree, mutable tree and self-generation tree to improve the efficiency of our schemes, and design two (static/dynamic) efficient R-MLE2 schemes (μR-MLE2). Both of the designed schemes support efficient data equality test while keeping the security of clients' data by allowing a small number of interactions.

1.1 Related works

Convergent encryption [7] ensures data privacy in deduplication. It is a *deterministic* scheme, where a ciphertext $C = E(k, m)$ is an encryption over message m under a message-dependent key $k = h(m)$, where h is a cryptographic hash function and E is a block cipher. In the deterministic scheme, identical plaintexts will be mapped to one ciphertext. When a client uploads the encrypted plaintext to a server, the server can find the duplicate ciphertext and store only one copy of each data. In this cross-user secure deduplication scheme, the clients need not to coordinate their actions or consider the existence of other clients who hold the same data copy.

Bellare et al. [4] formalized this primitive as message-locked encryption, and explored its application in space-efficient secure outsourced storage. An MLE scheme $\mathcal{MLE} = (\mathsf{P}, \mathsf{K}, \mathsf{E}, \mathsf{D}, \mathsf{T})$ is composed of five polynomial time algorithms. In $\mathcal{MLE}$, the parameter generation algorithm P is used to generate the public parameter. The key generation algorithm K is used to generate the message-derived key. On inputting a key and a message the encryption algorithm E outputs the ciphertext. The decryption algorithm D reverses the process, whose output is used to compute the ciphertext/plaintext, and the tag generation algorithm T is used to generate the tag of the ciphertext. In the scheme, tag generation maps the ciphertext to a tag and identical plaintext result in one equal tag.

To enhance the security of deduplication and protect the data confidentiality, Bellare et al. [3] showed how to protect the data confidentiality by transforming the predictable message into an unpredictable message. In their system, a third party called key server is introduced to generate the file tag for duplicate check. Recently, Liu et al. [14] designed a secure deduplication scheme without additional independent servers. Li et al. [12] addressed the key management issue in block-level deduplication by distributing these keys across multiple servers after encrypting the files. Li et al. [13] considered the hybrid cloud architecture consisting of a public cloud and a private cloud and efficiently solved the problem of deduplication with differential privileges. Yuan et al. [20] proposed a deduplication system in the cloud storage to reduce the storage size of the tags for integrity check. Recently, Bellare and Keelveedhi [2] proposed a new primitive iMLE, which adopted interaction as a new ingredient to provide privacy for messages that are both correlated and dependent on the public system parameters.

Abadi et al. [1] provided stronger security guarantee for secure deduplication. The first approach was to avoid using tags that are derived deterministically from the message. They designed a fully randomized scheme that supported equality test over ciphertext. More precisely, there were three components in the fully randomized scheme, namely a payload, a tag and a proof of consistency. The tag they designed for plaintext m is computed as $\tau = (g^r, g^{rh(m)})$, where g is the generator of a bilinear group, h is a sufficient strong collision-resistant function, and r is a randomly chosen number. Given two tags $\tau_1 = (g_1, h_1)$ and $\tau_2 = (g_2, h_2)$, the equality-testing algorithm verifies $e(g_1, h_2) \stackrel{?}{=} e(g_2, h_1)$. The second approach was a deterministic scheme. It was made secure subject to the condition

where the distributions were efficiently samplable using at most q queries to the random oracle. Thus, the security of the second approach was guaranteed by limiting the computational power of the adversarial message distributions.

1.2 Our contributions

Building on the above insight, we make several contributions, as follows:

1. This is the first attempt to solve the open problem pointed out by [1] "the first scheme (R-MLE2) requires a pairwise application of the equality-testing algorithm to identify all duplicate ciphertexts". We reduce the linear pairing comparison times of the R-MLE2 to nearly logarithmic times.
2. By adopting client-server interaction, we construct two deduplication decision tree structures: *static deduplication decision tree* and *dynamic deduplication decision tree*. The static one is suitable for static data, while the dynamic one, based on the self-generation tree, allows data update such as data insertion, deletion, and modification.
3. We provide the security and theoretical performance analysis for the proposed schemes, which show that our scheme is both secure and efficient.

2 Preliminaries and Notation

2.1 Notation

The set of binary string of length n is denoted as $\{0,1\}^n$, and the set of all finite binary strings are denoted as $\{0,1\}^*$. We denote the bit length of a given binary string s as $|s|$. Given two binary strings s_1 and s_2, the concatenation is written as $s_1||s_2$. The notation $[1,n]$ denotes the integer set $\{1,...,n\}$ with $n \in N$. We denote the output x of an algorithm $\mathcal{A}$ as $x \leftarrow \mathcal{A}$. Sampling uniformly random from a set X is denoted as $x \xleftarrow{R} X$. Also, $A \leftarrow B$ is used to denote the communication between two entity A and B. Throughout, λ is denoted as the security parameter, and $h(\cdot)$ is modeled as hash function.

2.2 Bilinear pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p, g be a generator of $\mathbb{G}$. A bilinear pairing is a map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- Bilinear: $\hat{e}(u^a, v^b) = \hat{e}(u,v)^{ab}$ for all $u, v \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p^*$
- Non-degenerate: $\hat{e}(g,g) \neq 1$.
- Computable: It is efficient to compute $\hat{e}(u,v)$ for all $u, v \in \mathbb{G}$.

2.3 Decision trees

A decision tree is a decision support tree-like model, where the decision process walks the tree from the root. The tree nodes, correspond to partitioning rules, are used to decide which branch to take until a leaf node is encountered.

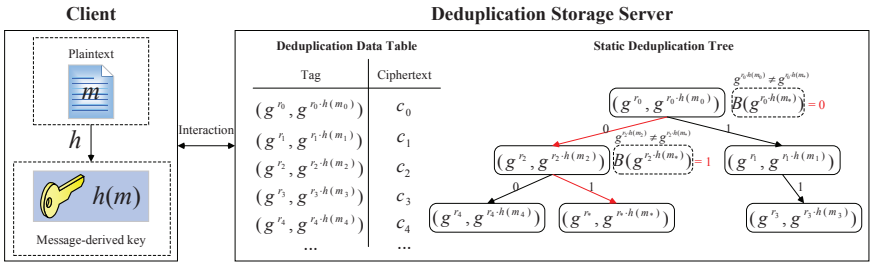

Fig. 1. The General Network Structure of μR-MLE2

2.4 MLE for lock-dependent message

A message-locked encryption for lock-dependent messages MLE2 [1] is a six-tuple $\Pi = (\mathsf{PPGen}, \mathsf{KD}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{EQ}, \mathsf{Valid})$ defined below.

- The parameter generation algorithm PPGen takes as input 1^λ and returns public parameters pp.
- The key derivation function KD takes as input public parameters pp, a message m, and outputs a message-derived key k_m.
- The encryption algorithm Enc takes as input public parameters pp, a message m, and a message-derived key k_m. It outputs a ciphertext c.
- The decryption algorithm Dec takes as input public parameters pp, ciphertext c, and a secret key k and outputs either a message m or $\perp$.
- 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.
- 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.

3 Security Model and Definitions

Our system consists of the clients and a cloud storage server as shown in Fig. 1. The clients (or data owners), will outsource their encrypted data to the untrusted cloud storage server. We consider the following models and basic properties.

Definition 1. *(μR-MLE2) An efficient fully random message-locked encryption scheme with randomized tag is an eight-tuple of polynomial-time algorithms* $\Pi = (\mathsf{PPGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TreeInit}, \mathsf{EQ}, \mathsf{Valid}, \mathsf{Dedup})$ *run by a client and a deduplication server.*

- $pp \leftarrow \mathsf{PPGen}(1^\lambda)$*: The parameter generation algorithm takes* 1^λ *as input and outputs the public parameter pp.*
- $k_m \leftarrow \mathsf{KeyGen}(pp, m)$*: The key generation algorithm takes the public parameters pp and a message m as inputs, and outputs a message-derived key* k_m*.*

- $c \leftarrow \mathsf{Enc}_{pp}(k_m, m)$*: The encryption algorithm takes the public parameters pp and the message derived key* k_m *as inputs, and returns a ciphertext c.*
- $m \leftarrow \mathsf{Dec}_{pp}(k_m, c)$*: The algorithm takes the public parameter pp and the message derived key* k_m *as inputs. If the algorithm runs successfully, it will return the plaintext m. Otherwise, it will return* $\perp$*.*
- $ts \leftarrow \mathsf{TreeInit}(1^\lambda)$*: The tree initialization algorithm takes* 1^κ *as input, and outputs the tree state ts of the current database.*
- $\{0,1\} \leftarrow \mathsf{EQ}_{pp}(\tau_1, \tau_2)$*: The equality-testing algorithm takes the public parameter pp and the tags* τ_1 *and* τ_2 *of two ciphertexts as inputs, and outputs 1 if the tags of the ciphertexts are generated from identical messages.*
- $\{0,1\} \leftarrow \mathsf{Valid}_{pp}(c)$*: The validity-testing algorithm takes public parameters pp and the ciphertext c as input. It outputs 1 if the ciphertext c is a valid input and 0 otherwise.*
- $\{0,1\} \leftarrow \mathsf{Dedup}_{pp}(st, \tau_1, \tau_2)$*: The data deduplication algorithm takes the public parameters pp,* τ_1*, and tag* τ_2 *as inputs. It returns whether a duplicate data copy has been found.*

Intuitively, with a client holding message m' and its corresponding tag τ', the scheme should direct to the identical data copy if a duplicate value is stored in the storage server. We consider that the server stores a sequence of data $\{c_1, ..., c_n\}$ and the corresponding tag values $\{\tau_1, ..., \tau_n\}$. The tree states evolve after each storing (there is no duplication data copy stored in the storage server), from ts_0 to ts_n, where ts_0 is the initial state. We define the following properties.

Definition 2. *(Correctness). A* μ*-RMLE2 scheme for the plaintext domain D is correct if for all security parameter* λ*, tag equality test algorithm* $\{0,1\} \leftarrow EQ_{pp}(\tau_i, \tau_j)$*, and deduplication algorithm* $\{0,1\} \leftarrow Dedup_{pp}(st, \tau)$ *for all data sequence* $c_1, ..., c_n$ *and tag sequence* $\tau_1, ..., \tau_n$*, for all tree states* **ts***, we have*

$Dedup_{pp}(st, \tau_i) = Dedup_{pp}(st, \tau_j)$ *for all steps, and finally get* $EQ_{pp}(\tau_i, \tau_j) = 1$ *iff* $m_i = m_j$*.*

We now define the security of our scheme, which intuitively says that the scheme must not leak anything besides the bits for deduplication path choosing in the deduplication test tree. The security definition is the Path-PRV-CDA2. The definition says that an adversary cannot distinguish between two test sequences of values as long as the sequences have the same tree path.

Path-PRV-CDA2 security game. The security game between a client and an adversary *Adv* for security parameter λ proceeds as follows:

The client and the server run μR-MLE2 as constructed, the client and the adversary engage in a number of rounds of interaction (not larger than the height of deduplication decision tree), where the client randomly samples message from real or rand mode.

- At round i, the client will send 1-bit path decision value to the adversary.

- With the additional bit information, the adversary conducts the Path-PRV-CDA2 game, $\mathsf{Exp}^{\mathsf{mode}}_{\Pi,\mathcal{A}}(\lambda)$, as defined in [1] .

- The adversary outputs b.

We provide our secure definition of Path-PRV-CDA2 security based on the definition PRV-CDA2 security presented in [1].

Definition 3 (Path-PRV-CDA2 security). *A μR-MLE2 scheme Π is Path-PRV-CDA2 secure if for any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function* $\mathbf{negl}(\lambda)$ *such that*

$$\mathrm{Adv}^{\mathsf{Path\text{-}PRV\text{-}CAD2}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} |\Pr\left[\mathsf{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(\lambda)=1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{rand}}_{\Pi,\mathcal{A}}(\lambda)=1\right]| \leq \mathsf{negl}(\lambda),$$

where for each $\mathsf{mode} \in \{\mathsf{real}, \mathsf{rand}\}$ *and* λ *the experiment is from* $\mathsf{Exp}^{\mathsf{mode}}_{\Pi,\mathcal{A}}(\lambda)$.

We also define the efficiency and the dynamic properties for our schemes.

Definition 4. *(Efficiency) We say a μR-MLE2 scheme is efficient, if the scheme is of asymptotically optimal over deduplication test, namely sublinear equality-test time.*

Definition 5. *(Dynamic) We say a μR-MLE2 scheme is dynamic, if the scheme can be efficiently added, deleted and changed after the initial outsourcing.* [1]

4 The μR-MLE2 Constructions

4.1 High-Level description

Abadi et al. [1] proposed a construction for building fully randomized message-locked encryption scheme based on entropy-based DDH assumption. In the scheme, the "payload" is used to store the encryption of message using some underlying randomized encryption scheme, and the tag is generated from the message. There is a proof of consistency, which proves that the payload and the tag correspond to the same message. A tag for a message m is computed as $\tau = (g^r, g^{r \cdot h(m)})$. Given two tags $\tau_1 = (g^{r_1}, g^{r_1 \cdot h(m_1)})$ and $\tau_2 = (g^{r_2}, g^{r_2 \cdot h(m_2)})$ the equality algorithm verifies $\hat{e}(g^{r_1}, g^{r_2 \cdot h(m_2)}) \stackrel{?}{=} \hat{e}(g^{r_2}, g^{r_1 \cdot h(m_1)})$.

However, the server needs to conduct data equality test over the whole database, which is inefficient in practical utilization. To solve this problem, we provide an efficient scheme. The main trick is that we adopt an interactive way to construct decision tree structures over the deduplication database, where the client who wants to store data needs to conduct a number of interactions with the server to verify whether the data is a duplicate copy. More precisely, the server maintains a decision tree, which stores the storage states of the current database. A client, who wants to store data, will interact with the server, where the server provides the tree state and the client provides a path decision over the decision tree in each communication round. Trivially, given the private key $h(m)$ and some relevant information, the client computes and sends a 1-bit path decision to the server in each step. When there is no duplicate data stored, the

[1] We assume the specific operation over the outsourced data is step by step.

node pointer of the state tree will move a null child node of a leaf node. Then, the server will store the data and update the decision tree state.

The first decision tree, based on which the scheme conducts data deduplication test without pairing computation, is a static tree. It means that, the deduplication scheme based on the static deduplication decision tree is efficient. However, it does not efficiently support the data insertion and deletion, based on which the deduplication is difficult to support data update. Note that efficient decision tree update is important in practical applications, we design a deduplication decision which supports efficient tree update. Our main trick is to use a self-generation tree constructed from a public seed, where the server verifies whether the data form is identical to the current node in the tree and returns the result to the client then indecently provides the server which node path to take in the next step.

4.2 The proposed μR-MLE2 schemes

In this section, we present the detail construction of our efficient randomized MLE2 (μR-MLE2) based on the definition of fully randomized MLE2 [1]. The scheme μR-MLE2 $\Pi = (\mathsf{PPGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{TreeInit}, \mathsf{EQ}, \mathsf{Valid}, \mathsf{Dedup})$ is polynomial in the security parameter. Since our schemes are based on R-MLE2, we omit some construction details of the algorithms in [1], and provide only the related three algorithms as follows:

- Tree initialization algorithm $ts \leftarrow \mathsf{TreeInit}(1^\lambda)$: It initializes server state st with *static/dynamic* deduplication decision tree and returns st.
- Equality-testing algorithm $\{0,1\} \leftarrow \mathsf{EQ}_{pp}(\tau_1, \tau_2)$: On input $\tau_1 = (g_1, h_1) \in \mathbb{G}^2$ and $\tau_2 = (g_2, h_2) \in \mathbb{G}^2$, the algorithm verifies $\hat{e}(g_1, h_2) \stackrel{?}{=} \hat{e}(g_2, h_1)$ and outputs 1 if and only if $\hat{e}(g_1, h_2) = \hat{e}(g_2, h_1)$.
- Data deduplication algorithm $\{0,1\} \leftarrow \mathsf{Dedup}_{pp}(st, \tau_1, \tau_2)$: On input the public parameters pp, τ_1, and tag τ_2, it calls the algorithm EQ at each node in a tree path until the leaf node if $0 \leftarrow \mathsf{EQ}$. It outputs 0 when all the EQ test output 0. Otherwise it output 1.

We design two equality test algorithms based on the static and dynamic deduplication decision tree, respectively.

Static deduplication decision tree. Fig. 1 provides an example about storing m^* based on static deduplication decision tree. A client, with message m_*, wants to conduct secure deduplication. It generates the corresponding tag $\tau = (g^{r_*}, g^{r_* \cdot h(m_*)})$ over message m_*. Also, the storage server stores the deduplication data table and maintains its corresponding deduplication decision tree.

As shown in Algorithm 1, the tag comparison query path is generated according to the data storage sequence of clients, which only allows the server to add data at the leaf nodes. Also, it can be called client-oriented deduplication scheme because the deduplication is conducted at the client side. We need to remark that, the static scheme allows clients to get the detail construction of the deduplication decision tree stored in the storage server, which allows malicious

Algorithm 1: Equality test over *static* deduplication decision tree

1.1 Client$\longleftrightarrow$ Server: The client asks for the deduplication of new data m_*, and the server returns the tag of the current node $(g^{r_i}, g^{r_0 \cdot h(m_i)})$. (Initially, the current node is the root of the tree and its tag is $(g^{r_0}, g^{r_0 \cdot h(m_0)})$.

1.2 Client: The client computes $g^{r_i \cdot h(m_*)}$ and verifies $g^{r_i \cdot h(m_*)} \stackrel{?}{=} g^{r_i \cdot h(m_i)}$.

1.3 Client$\longrightarrow$Server: If $g^{r_i \cdot h(m_*)} = g^{r_i \cdot h(m_i)}$, the client sends "duplication find" to the server. Otherwise, it computes $b = B(g^{r_i \cdot h(m_*)}) \in \{0, 1\}$ and sends b to the server.

1.4 Server: The server moves the current pointer of the tree according to b. If $b = 0$, the server moves the pointer to its left child. Otherwise, it moves the current pointer to its right child. Then, return to step 1.1. The algorithm stops, when the server receives "duplication find", or when the server needs to move the pointer to an empty node.

attackers to monitor the data deduplication activity of the storage server, such as the exactly time when a new data is uploaded to the storage server.

Dynamic deduplication decision tree. Fig. 2 provides an example about storing m^* based on dynamic deduplication decision tree. The dynamic deduplication decision tree is a self-generation tree, where the seed s_0 of the tree is a public parameter generated by the server. The left(right) child of node with $s_{0b_1b_2\ldots b_i}$ is $s_{0b_1b_2\ldots b_i} = h(s_{0b_1b_2\ldots b_{i-1}}||0)(s_{0b_1b_2\ldots b_i} = h(s_{0b_1b_2\ldots b_{i-1}}||1))$. The scheme, as shown in Algorithm 2, can be called server-oriented deduplication scheme, where the deduplication test is conducted by the storage server. It hides the deduplication decision tree structure stored in the storage server, which will defeat the security problem in static deduplication scheme.

Algorithm 2: Equality test over *dynamic* deduplication decision tree

2.1 Client$\longrightarrow$ Server: The client asks for the deduplication of new data m_* and sends the server $\tau = (g^{r_*}, g^{r_* \cdot h(m_*)})$ and b_i. (Initially, $b = -1$, which means that the current node is the root of the tree and the corresponding tag is $\tau = (g^{r_0}, g^{r_0 \cdot h(m_0)})$.

2.2 Server: The server verifies whether $e(g^{r_*}, g^{r_i \cdot h(m_i)}) = e(g^{r_i}, g^{r_* \cdot h(m_*)})$.

2.3 Server $\rightarrow$ Client: The server returns 1 when the equation holds, and 0 otherwise.

2.4 Client: When the client receives 0 from the server, the client computes $s_{0b_1\cdots b_i} = h(s_{0b_1b_2\ldots b_{i-1}}||b_i)$. Then it computes $b_i = B(h(m)||s_{0b_1\cdots b_{i-1}})$. (The initial seed is s_0.)

2.5 Client $\rightarrow$Server: The client sends b_{i+1} to the server.

2.6 Server: The server moves the current pointer over the tree according to b_{i+1}. If $b_{i+1} = 0$, the server moves the pointer to its left child. Otherwise, it moves the pointer to its right child. Then, go to step 2.3. The algorithm 2 stops as described in algorithm 1.

The tree construction relies on the self-generated hash value $s_{0b_1...b_i}$ in self-generation tree, and clients can generate the tree themselves. More precisely, the path decision of dynamic deduplication scheme is decided according to $b = B(h(m)||s_{0b_1...b_i})$, which is independent of data storage sequence. The complement and deletion operations are obvious. To insert a value at a certain position, firstly conduct the complement operation and move the existing data to a leaf node. Then, insert the specified data to the current position. To delete a node, firstly delete the current node. Then, choose an appropriate leaf node of the deleted node and insert it to the position of the deleted node. With dynamic deduplication tree, we are able to improve the efficiency of our scheme by conducting tree balancing as illustrated in appendix B. To further reduce the communication round, the client could compute and send multiple bits to the server. Also, the client could reduce some computational overhead of the hash value generation by storing some hash elements of the self-generation tree. The security analysis of our proposed scheme will be given in the full version of this paper.

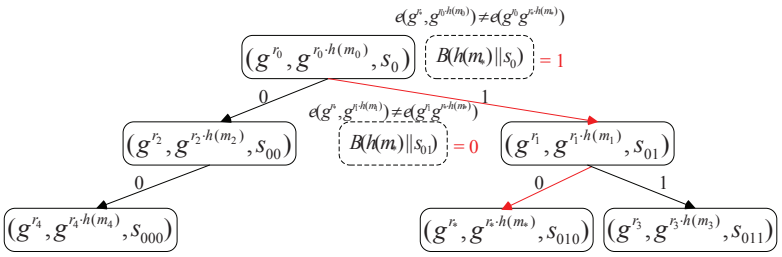

Fig. 2. Dynamic Deduplication Tree

5 Performance Analysis

We provide theoretical efficiency analysis and comparison of the proposed schemes. In appendix A, we discuss the theoretical results over our decision tree. Also. appendix B provide the tree optimization from deduplication tree balancing. Based on these analyzes, Table 1 and Table 2 illustrate the comparison among the three schemes in terms of both asymptotic computation and communication complexity and actual execution time. In our computation analysis, **Hash** denotes hash operation mapping a bit-string to a designed length value, **Mul** denotes a multiplication operation, **Exp** denotes the exponentiation operation in $\mathbb{G}$, and **Pair** denotes the pairing operation. In our communication analysis, we consider the bit-length of the content that needed to transfer between the client and the server. In our analysis, **SND** and **RCV** denote the overhead of sending and receiving a message with a certain length respectively.

Table 1. Computation Overhead of Data Deduplication

We omit lightweight string comparison overhead and some common computational overhead among the three scheme such as key generation, data encryption and validity testing.

Scheme	Client	Server
R-MLE2	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash}$	$2\mathbf{Pair} \cdot O(n)$
μR-MLE2 (Static)	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash} + (\mathbf{Exp} + \mathbf{Hash}) \cdot O(h)$	$\varnothing$
μR-MLE2 (Dynamic)	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash} + (2\mathbf{Hash}) \cdot O(h)$	$2\mathbf{Pair} \cdot O(h)$

As shown in Table 1 and Table 2, the client needs to conduct one-time tag generation and transmission. Then, the server will use pairing computation over the whole database to realize the equality test. More precisely, the computation and communication overhead of the server are $O(1)$ and $O(n)$, respectively.

The scheme based on static deduplication decision tree needs to fetch the tags in the path of deduplication decision tree and verify whether there is a duplicate copy of data stored in the server. Then, it sends a 1-bit path decision information to the server for choosing the path over the deduplication decision tree. The server just provides the tag values for the client according to the 1-bit path decision information each time. Since the static scheme is client side equality test, it does not need the expensive pairing computation. Instead, compared with the dynamic scheme, the scheme based on the static one needs much more communication overhead in each communication round.

The scheme based on dynamic deduplication decision tree, will greatly reduce both the communication and computation over of the client. More precisely, the client will only compute the path decision bit with the seeds of self-generation tree and send it to the server each time. The server will conduct the expensive equality test based on bilinear pairing. The maximum communication rounds of our schemes are decided by the deduplication decision tree height h, while not the whole data items n stored in the server. Actually, with the self-generation tree, the client could send multiple bits to help the server to conduct path decision each time, which will further reduce the communication rounds of the scheme. Compared with the scheme based on static deduplication decision tree, the client conducts lightweight hash operations and leaves the expensive pairing computation to the server.

Remark 1. The maximum equality test times of our schemes are the height of the deduplication decision tree. According to the theoretical analysis of the deduplication decision tree height in the appendix A, our scheme is efficient. As the tree updating and the tree balancing discussion in the appendix B, it is obvious that the scheme based on the dynamic deduplication decision tree is a dynamic scheme. For data deletion, the server could directly delete the relation between the deduplication decision tree and the data item. For data insertion, if a node is empty, the server will just insert the data. Otherwise, the data owner need help the server to move the data to a leaf node of the tree according to the deduplication policy and then insert the data element to the designed node.

Table 2. Data Deduplication Communication Overhead

We omit the ciphertext uploading overhead among the three scheme. We do not provide the communication overhead of the server, because the communication content between the server and client is asymmetric. The server-side communication overhead is **SND**(**RCV**), when the client-side communication overhead is **RCV**(**SND**).
* $x = |g^r| + |g^{r \cdot h(m)}| + O(h)$ and $y = (|g^r| + |g^{r \cdot h(m)}|)O(h)$ and $1 \leq x \leq h$.

Scheme	Communication bits (Client)	Communication rounds
R-MLE2	$\mathbf{SND}(\|g^r\| + \|g^{r \cdot h(m)}\|)$	$O(1)$
μR-MLE2 (Static)	$\mathbf{SND}(x) + \mathbf{RCV}(y)$ *	$O(h)$
μR-MLE2 (Dynamic)	$\mathbf{SND}(x) + \mathbf{RCV}(O(h))^\star$	$O(h)$

6 Conclusion

We explore intractive avenues to extentd the efficiency of fully randomized secure deduplication scheme. We construct two interactive schemes based on static and dynamic deduplication decision tree structures, respectively. The scheme based on the static deduplication decision tree does not allow the tree to update, while the scheme the dynamic deduplication decision tree is constructed based on the designed self-generation tree, which allows the server to conduct tree update and some other optimization such as tree balancing based on deduplication access frequency of the user. We also provide the security and performance analysis of our scheme, which show that our scheme is Path-PRV-CDA2 secure and it achieves several orders of magnitude higher performance than the state-of-the-art scheme in practical data deduplication.

7 Acknowledgement

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (No. 61272455), China 111 Project (No. B08038), Doctoral Fund of Ministry of Education of China (No.20130203110004), Program for New Century Excellent Talents in University (No. NCET-13-0946), and the Fundamental Research Funds for the Central Universities (No. BDY151402); Additionally, this work is also supported by US National Science Foundation under grant (CNS-1217889 and CNS-1446479).

References

[1] Abadi, M., Boneh, D., Mironov, I., Raghunathan, A., Segev, G.: Message-locked encryption for lock-dependent messages. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Computer Science, vol. 8042 of LNCS, pp. 374–391. Springer (2013)

[2] Bellare, M., Keelveedhi, S.: Interactive message-locked encryption and secure deduplication. In: Katz, J. (ed.) PKC 2015, Computer Science, vol. 9020 of LNCS, pp. 516–538. Springer (2015)

[3] Bellare, M., Keelveedhi, S., Ristenpart, T.: Dupless: Server-aided encryption for deduplicated storage. In: Proc. of the USENIX Security Symposium. pp. 179–194. DC, USA (Aug 2013)

[4] Bellare, M., Keelveedhi, S., Ristenpart, T.: Message-locked encryption and secure deduplication. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013, Computer Science, vol. 7881 of LNCS, pp. 296–312. Springer (2013)

[5] Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. IEEE Transactions on Parallel and Distributed Systems 25(9), 2386–2396 (Jul 2014)

[6] Chen, X., Li, J., Weng, J., Ma, J., Lou, W.: Verifiable computation over large database with incremental updates. In: Kutyowski, M., Vaidya, J. (eds.) ESORICS 2014, Computer Science, vol. 8712 of LNCS, pp. 148–162. Springer-Verlag (2014)

[7] Douceur, J., Adya, A., Bolosky, W., Simon, D., Theimer, M.: Reclaiming space from duplicate files in a serverless distributed file system. In: Proc. of IEEE International Conference on Distributed Computing Systems. pp. 617–624. Macau, China (Jun 2002)

[8] Dropbox: Dropbox. `https://www.dropbox.com/`, your stuff, anywhere

[9] Google: Google drive. `http://drive.google.com`, all your files, ready where you are.

[10] Harnik, D., Pinkas, B., Shulman-Peleg, A.: Side channels in cloud services: Deduplication in cloud storage. In: Proc. of IEEE Symposium on Security and Privacy. pp. 40–47. CA, USA (Jan 2010)

[11] Knuth, D.E.: Optimum binary search trees. Journal of Acta Informatica 1, 14–25 (May 1971)

[12] Li, J., Chen, X., Li, M., Li, J., Lee, P., Lou, W.: Secure deduplication with efficient and reliable convergent key management. IEEE Transactions on Parallel and Distributed Systems 25, 1615–1625 (Nov 2013)

[13] Li, J., Chen, X., Li, M., Li, J., Lee, P.P.C., Lou, W.: A hybrid cloud approach for secure authorized deduplication. IEEE Transactions on Parallel and Distributed Systems PP, 1–12 (Apr 2014)

[14] Liu, J., Asokan, N., Pinkas, B.: Secure deduplication of encrypted data without additional independent servers. In: Proc. of the ACM Conference on Computer and Communications Security. pp. 874–885. CO, USA (Oct 2015)

[15] Mehlhorn, K.: Nearly optimal binary search trees. Journal of Acta Informatica 5, 287–295 (May 1975)

[16] Mulazzani, M., Schrittwieser, S., Leithner, M., Huber, M., Weippl, E.R.: Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In: Proc. of USENIX Security Symposium. pp. 65–76. CA, USA (Aug 2011)

[17] NetApp: Netapp. `http://www.netapp.com/us/products/platform-os/dedupe.aspx`, universal Storage System

[18] Reed, B.: The height of a random binary search tree. Journal of the ACM 50, 306–332 (May 2003)

[19] Stanek, J., Sorniotti, A., Androulaki, E., Kencl, L.: A secure data deduplication scheme for cloud storage. In: Proc. of Financial Cryptography. pp. 99–118. CA, USA (Mar 2014)
[20] Yuan, J., Yu, S.: Secure and constant cost public cloud storage auditing with deduplication. In: Proc. of IEEE Conference on Communications and Network Security. pp. 145–153. MD, USA (Oct 2013)

Appendix A Tree Height

Most operations on a deduplication decision tree take time directly proportional to the height of the tree, so it is desirable to keep the height small. A binary tree with height h can contain at most $2^{h+1}-1$ nodes. It follows that for a tree with n nodes and height h where $h \geq \lfloor log_2^n \rfloor$.

We model the hash function as random oracle in this paper. Since the input $g^{r \cdot h(m)}$ each time is random in our first deduplication decision tree construction, we consider $g^{r \cdot h(m)}$ as a random bit-string. Then, we also consider $b = B(g^{r \cdot h(m)})$ as random bit.

In the self-generation binary tree, we consider the adopted hash function from a family of hash functions which maps the value of each key from some universe U into m. $H = h : U \to [m]$, where $\forall x, y \in U, x \neq y : \Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{m}$. Then, we can model the values in the self-generation tree as random values. Finally, we model $b = B(s||h(m))$ as random bit.

The two deduplication decision trees constructed are similar to the random binary trees widely studied to provide information useful in evaluating algorithms based on this storage structure. We model the hash value as the random input and adopt the compressed binary b to make path decision.

If we construct a binary search tree from a sequence of n different numbers by inserting them in the random order into an initially empty tree as shown in our two constructions. Let H_n be the height of the constructed tree on n nodes. As n approaches ∞, there exists constants $\alpha = 4.311...$ and $\beta = 1.953...$, such that the expected value $E(H_n) = \alpha lnn - \beta lnlnn + O(1)$, and the variety of H_n is $Var(H_n) = O(1)$ [18]. Here, ln is the natural logarithm and log is the base 2 logarithm. The expect height of the two deduplication decision trees are of logarithmic equality-testing time, which means that our schemes are efficient.

Appendix B Deduplication Decision Tree Balancing

An optimal binary search tree (BST) is usually used to provide the smallest possible search time for a given sequence of accesses. We consider the server side optimization over our dynamic deduplication decision tree, because it could gradually collect and record the deduplication access frequency of all the elements stored in the database. Actually, it is not practical for us to conduct tree balancing over the static deduplication decision tree, because we can not predict

the storing sequence and frequency affect the structure of deduplication decision tree. Also, all the first data owners relevant to the node's children need to take part into the tree balancing procedure in static scheme. Thus, we just consider tree balancing of dynamic scheme, where the tree can be modified at any time, typically by permitting tree rotations.

By generalizing the problem, we consider not only the frequencies with which a successful search is completed, but also the frequencies where unsuccessful searches occur. In our deduplication three, we consider there are n elements $B_1, ..., B_n$ and $2n + 1$ frequencies $\beta_1, ..., \beta_n$, $\alpha_1, ..., \alpha_n$ with $\sum \beta_i + \sum \alpha_j = 1$, where β_i is the frequency of encountering element B_i, and α_j is the frequency of encountering an element which lies between B_j and B_{j+1} as defined in [11]. In the dynamic deduplication decision tree with n interior nodes and $n + 1$ leaves, as defined in [15], the weighted path length of the tree is

$$P = \sum_{i=1}^{n} \beta_i (b_i + 1) + \sum_{j=0}^{n} \alpha_j a_j. \tag{1}$$

Let $n = 2^k - 1, \beta_i = 2^{-k} + \varepsilon_i$, with $\sum_{i=1}^{n} \varepsilon_i = 2^{-k}$ and $\varepsilon_1 > \varepsilon_2 > \cdots > \varepsilon_n > 0$ for $1 \leq i \leq n$ and $\alpha_j = 0$ for $1 \leq j \leq n$. In a balanced tree for the above frequency distribution, as shown in [15], its weighted path length is

$$P \leq 2^{-(k-1)} \sum_{i=1}^{n} (b_i + 1) \leq 2^{-(k-1)} \sum_{i=1}^{n} 2^{(l-1)} \cdot l \leq 2 \cdot logn. \tag{2}$$

If we get $\beta_i > \beta_j$ where β_i is the frequency of encountering element B_i and β_j is the frequency of B_i's child node B_j. We get the weighted path length sum of the two node is P. We exchange the position of element B_i and B_j, the sum of the weighted path length of the two node is P'. Since the distance of node B_i from the root is always smaller than that of its children, we have $b_i < b_j$. Then, we get

$$\begin{aligned} P - P' &= \beta_i (b_i + 1) + \beta_j (b_j + 1) - \beta_i (b_j + 1) - \beta_j (b_i + 1) \\ &= (\beta_i - \beta_j)(b_i - b_j) < 0 \end{aligned} \tag{3}$$

Equation 3 shows that we will get smaller weighted path length, if we move the element with larger frequency closer to the root. Thus, the server will be able to optimize the tree structure by moving element closer to the root in our scheme based on dynamic deduplication decision tree.