Secure and Efficient Cloud Data Deduplication With Randomized Tag

Tao Jiang, Xiaofeng Chen, *Senior Member, IEEE*, Qianhong Wu, *Member, IEEE*, Jianfeng Ma, *Member, IEEE*, Willy Susilo, *Senior Member, IEEE*, and Wenjing Lou, *Fellow, IEEE*

***Abstract*— Cross-client data deduplication has been widely used to eliminate redundant storage overhead in cloud storage system. Recently, Abadi *et al.* introduced the primitive of MLE2 with nice security properties for secure and efficient data deduplication. However, besides the computationally expensive non-interactive zero-knowledge proofs, their fully randomized scheme (R-MLE2) requires the inefficient equality-testing algorithm to identify all duplicate ciphertexts. Thus, an interesting challenging 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 for this challenging problem. We propose two schemes: static scheme and dynamic scheme, where the latter one allows tree adjustment by increasing some computation cost. Our main trick is to use the interactive protocol based on static or dynamic decision trees. The advantage gained from it is, 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 data items in the database. The security analysis and the performance evaluation show that our schemes are Path-PRV-CDA2 secure and achieve several orders of magnitude higher performance for data equality test than R-MLE2 scheme when the number of data items is relatively large.**

***Index Terms*— Deduplication, convergent encryption, message-locked encryption, interactive protocol.**

Manuscript received June 13, 2016; revised August 22, 2016; accepted October 21, 2016. Date of publication October 26, 2016; date of current version December 14, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61572382 and Grant U1405255, in part by the China 111 Project under Grant B16037, in part by the Doctoral Fund of Ministry of Education of China under Grant 20130203110004, in part by the Program for New Century Excellent Talents in University under Grant NCET-13-0946, in part by the Fundamental Research Funds for the Central Universities under Grant BDY151402, in part by the National High Technology Research and Development Program (863 Program) of China under Grant 2015AA016007, and in part by the U.S. National Science Foundation under Grant CNS-1217889 and Grant CNS-1446479. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sheng Zhong. *(Corresponding authors: Xiaofeng Chen and Willy Susilo.)*

T. Jiang is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China (e-mail: jiangt2009@gmail.com).

X. Chen is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710126, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China (e-mail: xfchen@xidian.edu.cn).

Q. Wu is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: qianhong.wu@buaa.edu.cn).

J. Ma is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710126, China (e-mail: jfma@xidian.edu.cn).

W. Susilo is with the Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: wsusilo@uow.edu.au).

W. Lou is with Department of Computer Science, Virginia Polytechnic Institute and State University, Fairfax, VA 22042, USA. (e-mail: wjlou@vt.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2622013

I. Introduction

REMOTE cloud storage have become an indispensable part for various applications in nowadays network, which store a large amount of data and provide the partial data needed. With the rapid growing of cloud storage services, such as cloud storage [2]–[4], encryption becomes an important technique for protecting the confidentiality of data. Although data encryption provides an important guarantee for the security and privacy of clients' data, it limits the manners of the accessibility and availability of the encrypted data. The limitation of schemes with encrypted data is that, when some special processing applications over the data are needed, such as cross-client data deduplication [5]–[7], query [8]–[13], sorting over encrypted data [14]–[18], the schemes usually become inefficient due to the frequent date encryption and decryption operations. Thus, it is important to design efficient schemes to support secure and efficient computation outsourcing [19], [20] and storage outsourcing [21].

Data deduplication enables data storage systems to find and remove duplication within 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.

Since the data from different clients is encrypted with different 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, and efficiently prevent the practical attacks [22]–[24] from poor

deduplication. Douceur et al. [21] 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 [25], [26] are implemented or designed based on convergent encryption. Recently, Bellare et al. [27] defined 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 depending on the public parameter, Abadi et al. [28] 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, concerning the challenging problem described in [28], the R-MLE2 scheme is not efficient in the deduplication process because of the comparison of the randomized tag introduced in their paper. 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 randomly balanced 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 interaction.

Our scheme is much more practical for cloud data deduplication application compared with R-MLE2. Firstly, our schemes inherit the security enhancement of R-MLE2. In other words, the scheme avoids using tags that are derived deterministically from the message. Secondly, since our schemes reduce the linear duplication test from linear time to sublinear time, they are much more efficient in the duplication test, especially when our scheme is applied to cloud storage server that storing a huge number of data items.

A. Related Works

Convergent encryption [21] ensures data privacy in deduplication. It is a *deterministic* scheme in which 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 do not need to coordinate their actions or consider the existence of other clients who hold identical data copy.

Bellare et al. [27] 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. [25] 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 duplication check. Li et al. [26] addressed the key management issue in block-level deduplication by distributing these keys across multiple servers after encrypting the files. Li et al. [29] 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 and Yu [30] proposed a deduplication system in the cloud storage to reduce the storage size of the tags for integrity check. Recently, Bellare and Keelveedhi [31] 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. [28] 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.

B. Our Contributions

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

1) This is the first attempt to solve the challenging problem pointed out by [28] "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, and implement the equality-testing schemes with the existing cryptographic libraries and provide the performance evaluation of our schemes.

This is the full version of the paper that has been presented in ACISP 2016 [1]. The main differences from the conference version are as follows: Firstly, introduce some explanations and examples in our paper. Secondly, we add a new section 5 to present the formal security proof. Finally, we use a new section 6 to provide a thorough experimental evaluation of the proposed scheme and compare it with the R-MLE scheme over data equality test.

II. Preliminaries

A. Bilinear Pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p. Let 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:

- Bilinearity: For all $u, v \in \mathbb{G}$, and $a, b \in \mathbb{Z}_p^*$, we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- Non-degeneracy: $\hat{e}(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists a group $\mathbb{G}_T$ and an efficiently computable bilinear map $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$. Also, $\hat{e}(,)$ is symmetric since $\hat{e}(g^a, g^b) = \hat{e}(g^b, g^a)$.

Definition 1 (Bilinear Diffie-Hellman (BDH) Problem): Given a tuple $g, g^a, g^b, g^c \in \mathbb{G}$ as input (for unknown randomly chosen $a, b, c \in \mathbb{Z}_p^*$), compute $e(g, g)^{abc} \in \mathbb{G}_T$.

The BDH problem is hard to solve in $\mathbb{G}$ as the assumption in [32]. In other words, $\mathbb{G}$ and $\mathbb{G}_T$ are chosen so that there is no known algorithm for efficiently solving the Diffie-Hellman problem in either $\mathbb{G}$ or $\mathbb{G}_T$.

B. Decision Trees

A decision tree is a decision support tree-like model, which works by partitioning the data space one attribute at a time. For example, a decision tree consisting of nodes (circles) and branches (lines) is shown in Fig. 1.

The nodes in the tree correspond to partitioning rules (e.g. conditions in Fig. 1), and they are used to decide which branch to take until a leaf node is encountered. A decision tree typically begins with a given first decision called the root node (the black circle). The lines that connect nodes are called branches. Branches that emanate from a decision node are called decision branches with possibly different conditions. A leaf node or a termination node (gray circles) indicates a final outcome for that branch. The decision process walks the tree by starting from the root. Decision trees support three main operations: query, insertion, and deletion.

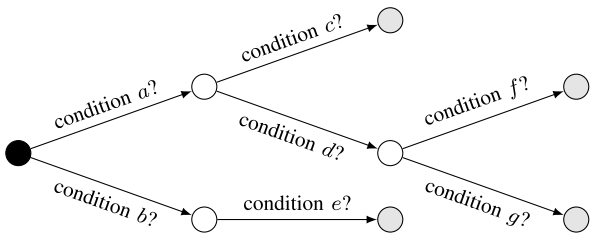

Fig. 1. Example of decision tree.

Query: Find a specific element in the tree. In data deduplication scheme, only one element will be found or no element is found.

Insertion: Insert new elements to a specific position of the decision tree. It is obvious to insert a leaf node, which can also be called node pending. However, we need to consider the relationship of the nodes with the inserted node as a root node, when inserting an intermediate node.

Deletion: Find a specific elements in the tree, and then delete it. Similar to insertion operation, deletion operation needs to consider node relationship if the deleted node is not a leaf node.

C. MLE for Lock-Dependent Message

A message-locked encryption for lock-dependent messages MLE2 [28] is a six-tuple Π = (PPGen, KD, Enc, Dec, EQ, 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_m 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.

III. Notation and Definitions

A. 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

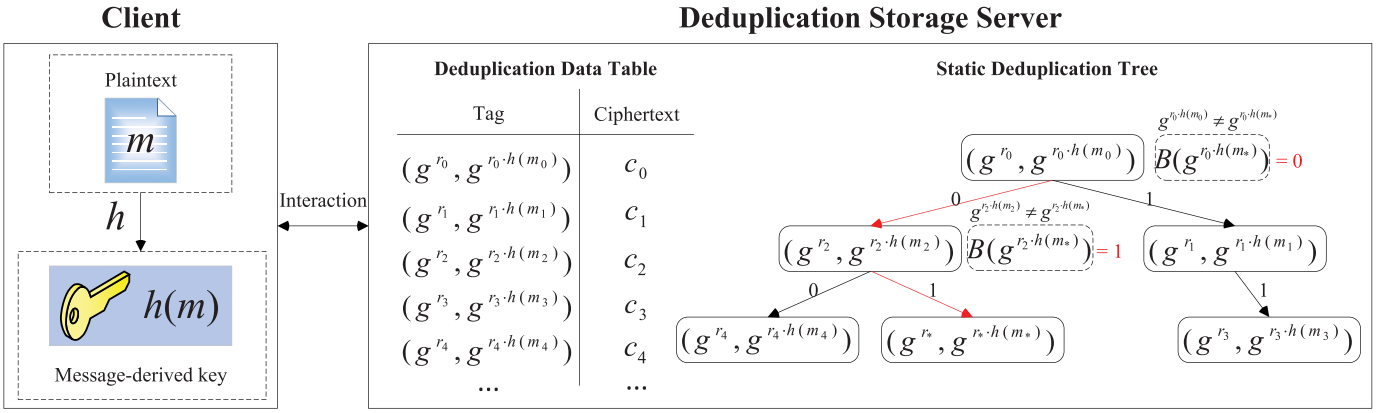

Fig. 2. The General Network Structure of μR-MLE2.

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. $B(x)$ denotes the bitwise exclusive or of the digest of x, and the digest of x can be calculated with a secure hash function. Throughout, λ is denoted as the security parameter, and $h(\cdot)$ is modeled as hash function.

B. Security Model and Definitions

Our system consists of the clients and a cloud storage server as shown in Fig. 2. The clients, who own the data, will outsource and store their encrypted data to the untrusted cloud storage server. We consider the following models and basic properties of our scheme.

Definition 2 (μR-MLE2): An efficient fully random message-locked encryption scheme with randomized tag is an eight-tuple of polynomial-time algorithms Π = (PPGen, KeyGen, Enc, Dec, TreeInit, EQ, Valid, 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, our scheme conducts data deduplication item by item among clients. A client does not need to upload its encrypted data item to the storage server when there is a duplicate copy stored. Otherwise, the client needs to upload its data. We consider that the server stores a sequence of data $\{c_1, ..., c_n\}$ and the corresponding tag values $\{\tau_1, ..., \tau_n\}$ at some point. When a client, holding message m' and its corresponding tag τ', wants to conduct data deduplication, the scheme should efficiently direct to the identical data copy if a duplicate value is stored in the storage server. The decision tree state evolves 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 3 (Correctness): A μ-RMLE2 scheme for the plaintext domain D is correct if for all security parameter λ, tag equality test algorithm $\{0, 1\} \leftarrow \mathsf{EQ}_{pp}(\tau_i, \tau_j)$, and deduplication algorithm $\{0, 1\} \leftarrow \mathsf{Dedup}_{pp}(st, \tau)$ for all data sequence $c_1, ..., c_n$ and tag sequence $\tau_1, ..., \tau_n$, for all tree states **ts**, we have

$\mathsf{Dedup}_{pp}(st, \tau_i) = \mathsf{Dedup}_{pp}(st, \tau_j)$ for all steps, and finally get $\mathsf{EQ}_{pp}(\tau_i, \tau_j) = 1$ if $m_i = m_j$.

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 [28] .

- The adversary outputs b.

We provide our secure definition of Path-PRV-CDA2 security based on the definition PRV-CDA2 security presented in [28].

Definition 4 (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 $\mathsf{negl}(\lambda)$ 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 following two properties for the efficiency analysis of our schemes.

Definition 5 (Efficiency): We say a μR-MLE2 scheme is efficient, if the equality-test time of the deduplication scheme is sublinear.

Definition 6 (Dynamic): We say a μR-MLE2 scheme is dynamic, if the scheme can be efficiently added, deleted and changed after the initial outsourcing.

IV. The μR-MLE2 Constructions

In this section, we present the general description of our constructions and provide the μR-MLE2 scheme based on static deduplication decision tree and dynamic deduplication decision tree. Also, we discuss some detail properties about our schemes.

A. High Description

Abadi et al. [28] 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 for 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 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.

B. 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 [28]. The scheme μR-MLE2 Π = (PPGen, KeyGen, Enc, Dec, $\mathsf{TreeInit}$, EQ, Valid, Dedup) is polynomial in the security parameter. Since our schemes are based on R-MLE2, we omit some construction details of the algorithms in [28], 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 outputs 1.

We design two equality test algorithms based on the static and dynamic deduplication decision tree, respectively. The static one is much more efficient, which does not need to conduct expensive pairing computation. The dynamic one is efficient in deduplication decision tree operations, which allows server side data insertion and deletion. Also, the dynamic one will further reduce the communication rounds between clients and the server.

Static Deduplication Decision Tree: Fig. 2 provides an example of storing m^* based on static deduplication decision tree. A client, with message m_*, wants to conduct secure deduplication. It generates the corresponding tag

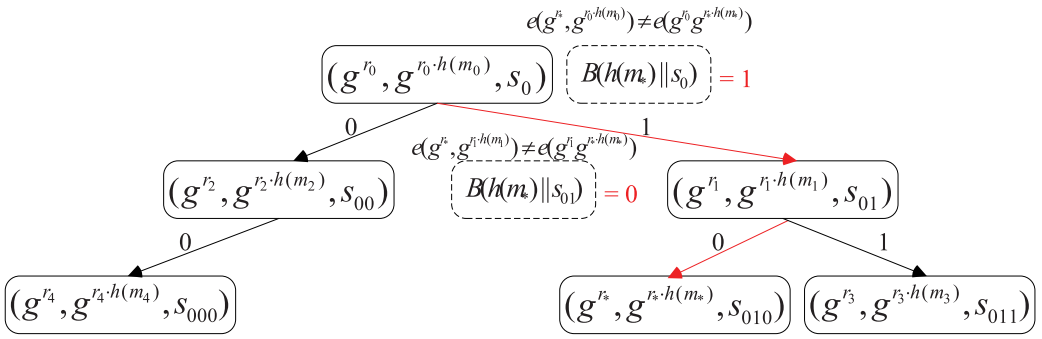

Fig. 3. Dynamic deduplication tree.

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 it needs to move the pointer to an empty node.

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 $s_{0b_1 \cdots b_i} = h(s_{0b_1 b_2 \ldots b_{i-1}} || b_i)$. 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 node to its right child. Then, go to step 2.3. The algorithm stops as described in algorithm 1.

$\tau = (g^{r_*}, g^{r_* \cdot h(m_*)})$ 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 scheme based on the static deduplication decision tree supports efficient query, leaf insertion, and leaf deletion operations. In the scheme, the tag comparison query path is generated according to the data storage sequence of clients. Since the path to each node is decided by each node in the path, the static deduplication decision tree only supports query, inserting/delete data at the leaf nodes. When conducting an intermediate node insertion/deletion, the server needs to reinsert all the nodes with the inserted/deleted node as a root node. Since these nodes may come from different clients, it is very difficult for the server to conduct intermediate node insertion/deletion.

Dynamic Deduplication Decision Tree: Fig. 3 provides an example of 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 $s_{0b_1 b_2 \ldots b_i}$ is $s_{0b_1 b_2 \ldots b_i} = h(s_{0b_1 b_2 \ldots b_{i-1}} || 0)(s_{0b_1 b_2 \ldots b_i} = h(s_{0b_1 b_2 \ldots b_{i-1}} || 1))$. All the $s_0, s_{0b_1} \ldots$ are used for computing a random decision bit, which is foundation of the random decision tree. 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.

In the scheme based on dynamic deduplication decision tree, the seed of self-generation tree is a public parameter. The tree construction relies on the self-generated hash value $s_{0b_1 \ldots b_i}$ in self-generation tree, and clients can generate the tree paths themselves of the owned data items. More precisely, the path decision of dynamic deduplication scheme is decided according to $b = B(h(m)||s_{0b_1 \ldots b_i})$, which is independent of data storage sequence. The insertion/deletion operations of leaf nodes are obvious. However, the server needs the assistant from the client who own the data items. For example, to insert an intermediate node at a certain position of the data tree path, the server first conducts the complement operation and moves the existing data to a leaf node with the help of the data owner. Then, it inserts the specified data to the empty position that storing the previous data item. To delete a node,

TABLE I
COMPUTATION OVERHEAD OF DATA DEDUPLICATION

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)	$2\mathbf{Exp} + \mathbf{Mul} + \mathbf{Hash} + (2\mathbf{Hash}) \cdot O(h)$	$2\mathbf{Pair} \cdot O(h)$

TABLE II
DATA DEDUPLICATION COMMUNICATION OVERHEAD

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

firstly delete the current node. Then, choose an appropriate leaf node with the deleted node in it path and insert it to the position of the deleted node. It is obvious that, in this scheme, only the intermediate node insertion operation needs the help from an additional client. Thus, it is efficient compared with the scheme based on static deduplication decision tree over decision tree operation. 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 each time. Also, the client could reduce some computational overhead of the hash value generation by storing some hash elements of the self-generation tree.

V. SECURITY ANALYSIS

Theorem 1: Our deduplication schemes are correct in data equality testing.

Proof: (Correctness) We assume that the storage server is honest but curious, and it strictly follows the protocol and constructs the deduplication decision tree. The correctness property required of the scheme is straightforward: equality should finally return only the right tags that exist at the server and the equality test algorithm will finally output the correct result.

Firstly, we need to prove that the client will get the right tree path. In scheme based on the static deduplication decision tree, if two messages are m and m', we get path decision bit $b = B(g^{r_i \cdot h(m)})$ and $b' = B(g^{r_i \cdot h(m')})$ for m and m', respectively. If $m = m'$, we get $b = b'$. Since the entrance of each tree path is the root of the decision tree, we will get the same tree path if the tree is deterministic. We need to remark that the scheme based on the static deduplication decision tree does not allow the server to change the tree construction, except adding leaf nodes. Thus, it is difficult to conduct any optimization the equality test over the deduplication database. In scheme based on the dynamic deduplication decision tree, since the tree is self-generated, it is obvious that $s_\alpha = s_\beta$ if $\alpha = 0b_1...b_i$, $\beta = 0b'_1...b'_i$, and $\alpha = \beta$. Then, if we assume $b = B(h(m)||s_\alpha)$ and $b' = B(h(m')||s_\beta)$, we will get $b = b'$ if $m = m'$.

Secondly, in each step of the two schemes, the client also needs to verify $e(g^{r_i}, g^{r_i \cdot h(m)}) \stackrel{?}{=} e(g^{r_i}, g^{r_i \cdot h(m')})$. It is obvious that the equation holds only when $m = m'$. The interaction of the client and server will reach a final deduplication test result as defined in correctness definition. ■

Theorem 2: Our μR-MLE2 scheme is Path-PRV-CDA2 secure.

Proof: Consider any adversary $\mathcal{A}$ and any two sequences of values $\mathcal{A}$ distinguishes $\mathbf{v} = (v_1, ..., v_n)$ and $\mathbf{v}' = (v'_1, ..., v'_n)$. The view of $\mathcal{A}$ consists of the information the server receives in the security experiment. The first is to use the security of R-MLE2 which is PRV-CDA2 secure. Then, we examine the information the adversary learns in case when the client uploads $\mathbf{v}$ and $\mathbf{v}'$, and show that this information is information-theoretically the same when the uploaded message is of the same path.

For this goal, we proceed inductively in the number of values to be uploaded. The base case is when no value is uploaded and we can see that $\mathcal{A}$ starts off with the same information. Now considering that after i steps, we show that the information after $i + 1$ steps also remains the same.

We have two possibilities. The first possibility is that v_i is in the deduplication table. Then the encoding of v'_i is also in the deduplication table because $\mathbf{v}$ and $\mathbf{v}'$ have the same tree path relation. In this case, the client does not give any more information to $\mathcal{A}$. The second possibility is that the encoding of v_i is not in the deduplication table. Since $\mathbf{v}$ and $\mathbf{v}'$ have the same order relation, the path down in the tree taken by client and $\mathcal{A}$ must be the same. Also, the only information the client gives to the server is which path to take in deduplication decision tree, which is also the same for both cases. Therefore, $\mathcal{A}$ receives the same information in both cases, and hence is Path-PRV-CDA2 secure, which leaks bits no more than the tree height of the deduplication decision tree. ■

VI. PERFORMANCE ANALYSIS

In this section, we provide theoretical and piratical efficiency analysis of the proposed schemes and give a comparison with scheme [28].

A. Theoretical Analysis

In Appendix A, we discuss the theoretical results over our decision tree. Also, Appendix B provides the tree optimization from deduplication tree balancing. Based on these analysis, Table I and Table II 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 is 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.

As shown in Table I and Table II, 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, the scheme does not need the expensive pairing computation. Instead, compared with the scheme based on dynamic deduplication decision tree, 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 overhead 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: Since our schemes are correct, 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 discussed in the appendix, 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 item. Otherwise, the server needs the owner of the data to move the data to a leaf node of the tree according to the deduplication policy and then it will insert the data element to an appropriate position of the decision tree.

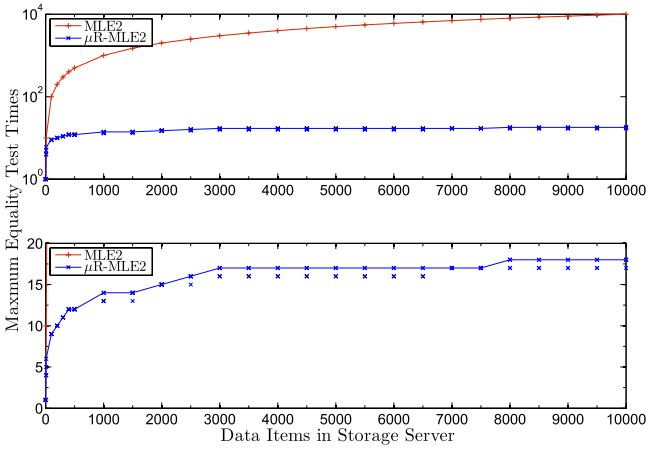

Fig. 4. The deduplication tree height.

B. Implementation

We implement the equality-test algorithms of R-MLE2 and μR-MLE2 schemes based on the Pairing Based Cryptography library (PBC) [33]. In our implementations, we instantiate the bilinear map with type-A pairing ($\lambda = 512$), which offers a level of security that is equivalent to 1024-bit DLOG [33]. For all the schemes, we instantiate the hash function with SHA-1 provided by the OpenSSL cryptographic library [34]. In our scheme, we construct the tree structure with C language. We provide the efficiency analysis of the schemes based on some random values. For comparison, the algorithms run by the storage server and the clients are both executed on a machine with Linux OS, 2.70GHz Intel(R) Core(TM) i7-4600U CPU and 8GB RAM.

Since the deduplication decision tree height (the maximum path length) is closely related to equality test time of deduplication, Fig. 4 shows the height of deduplication decision tree generated from random values. Since the equality-test times of R-MLE2 scheme are linear with the data item, we use a tree where each node has only one child. In our μR-MLE2 schemes, the height of our randomly generated deduplication decision tree is nearly logarithmic to the data items as shown in Fig. 4. We also get the result based on our simulation that the tree height is around 20 for 100,000 data item, which is not provided in Fig. 4.

We provide the time to generate the deduplication decision tree in Fig. 5. It is obvious that, to generate a tree with different data, our scheme achieves 1-2 orders of magnitude higher performance than the state-of-the-art R-MLE2 scheme even when there are less than 1000 data items in the tree. Also, our schemes will be much more efficient when applied over large data set deduplication.

Since the data items and the deduplication decision tree are randomly generated, the storage time of each data item is usually not the same each time. Fig. 6 and Fig. 7 show some detail about deduplication time of our static and dynamic schemes. It is obvious that, the deduplication time for the

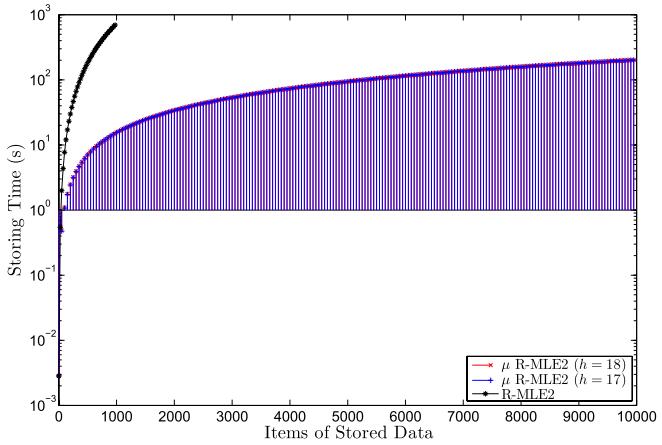

Fig. 5. Tree construction time.

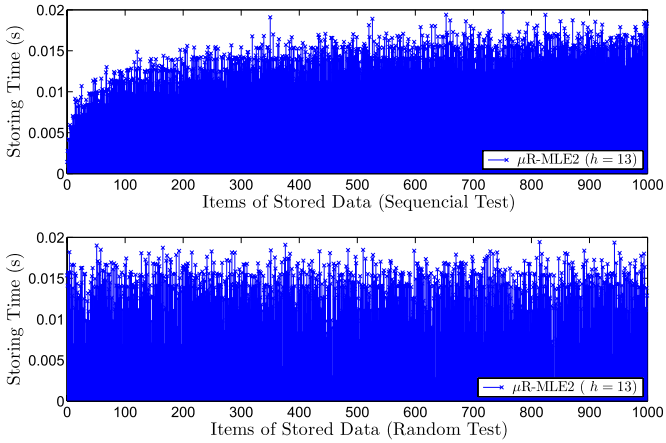

Fig. 6. Static μR-MLE2 storing time.

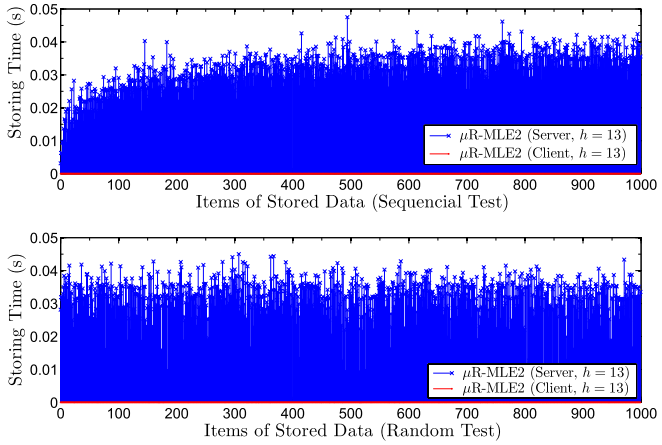

Fig. 7. Dynamic μR-MLE2 storing time.

sequential test is nearly logarithmic to the number of data items, and it is nearly linear for the random test. Since the client only needs to calculate the hash values, the client time cost in dynamic μR-MLE2 is not obvious compared to the expensive pairing computation at the server.

Fig. 8 shows the deduplication time comparison of our μR-MLE2 schemes. The static μR-MLE2 scheme needs less deduplication testing time than the dynamic one, since the client does not need to conduct pairing operations each time. The advantage of the dynamic scheme is that, the client only needs to conduct the lightweight hash function and leave the expensive pairing computation to the storage server.

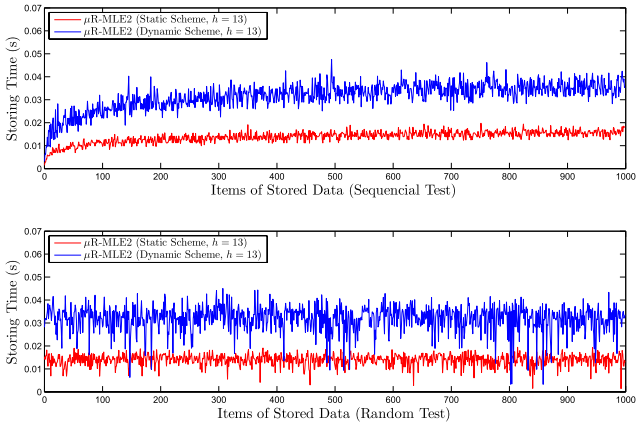

Fig. 8. Deduplication time comparison of the μR-MLE2 schemes.

VII. Conclusion

Interactive avenues are explored to improve the efficiency of fully randomized secure deduplication scheme R-MLE2. To achieve secure and efficient data deduplication, we construct two interactive schemes based on static and dynamic deduplication decision tree structures, respectively. The static deduplication decision tree is constructed based on the random elements from the client, which does not allow the tree to update. However, 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. The security, theoretical and practical performance analysis show that our scheme is Path-PRV-CDA2 secure and it achieves several orders of magnitude higher performance than the state-of-the-art schemes in practical data deduplication.

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 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. $H = h : U \rightarrow [m]$, 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 any value in the self-generation tree as random value. Finally, we model $b = B(s||h(m))$ as random bit.

The two deduplication decision trees constructed in our scheme are similar to the random binary trees widely studied to provide information useful in evaluating algorithms based

on this storage structure. In our constructions, we model the hash value as the random input and adopt the compressed binary b to make path decision.

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 exist 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 $Var(H_n) = O(1)$ [35]. Here, ln is the natural logarithm and log is the base 2 logarithm. The expected height of the two deduplication decision trees is of logarithmic equality-testing time, which means that our schemes are efficient.

APPENDIX B
DEDUPLICATION DECISION TREE BALANCING

In computer science, 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, since 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, as we will never know how the current storing sequence and frequency affect the structure of deduplication decision tree in the future. Also, if the server wants to change the position of a node in static deduplication for performance optimization, all the first data owners relevant to the node's children need to take part into this procedure. Thus, we just consider tree balancing over the scheme based on dynamic deduplication decision tree, 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 tree, 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 [36]. In the dynamic deduplication decision tree with n interior nodes and $n + 1$ leaves, as defined in [37], 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 $1 \le i \le n$ and $\alpha_j = 0$ for $1 \le j \le n$. In a balanced tree for the above frequency distribution, as shown in [37], its weighted path length is

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

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. The weighted path length sum of the two node is P. If 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

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

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.

ACKNOWLEDGMENT

This work is an extension of conference paper published in ACISP 2016 [1].

REFERENCES

[1] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, "Towards efficient fully randomized message-locked encryption," in *Proc. 21st Austral. Conf. Inf. Secur. Privacy (ACISP)*, Melbourne, VIC, Australia, Jul. 2016, pp. 361–375.

[2] *Dropbox*, accessed on Feb. 15, 2016. [Online]. Available: https://www.dropbox.com/

[3] Google. *Google Drive*, accessed on Mar. 6, 2016. http://drive.google.com

[4] NetApp. *Universal Storage System*, accessed on Mar. 6, 2016. [Online]. Available: http://www.netapp.com/us/products/platform-os/dedupe.aspx

[5] C. Batten, K. Barr, A. Saraf, and S. Trepetin, "pStore: A secure peer-to-peer backup system," MIT Lab. Comput. Sci., Cambridge, MA, USA, Progr. Rep. 6.824, 2001.

[6] M. W. Storer, K. Greenan, D. D. E. Long, and E. L. Miller, "Secure data deduplication," in *Proc. 4th ACM Int. Workshop Storage Secur. Survivability*, New York, NY, USA, Oct. 2008, pp. 1–10.

[7] L. Marques and C. J. Costa, "Secure deduplication on mobile devices," in *Proc. Workshop Open Source Design Commun.*, Lisbon, Portugal, Jul. 2011, pp. 19–26.

[8] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2000, pp. 44–55.

[9] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, Oct. 2006, pp. 79–88.

[10] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for Boolean queries," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8042. Berlin, Germany: Springer, Aug. 2013, pp. 353–373.

[11] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, Raleigh, NC, USA, Oct. 2012, pp. 965–976.

[12] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Financial Cryptogr.*, Okinawa, Japan, Apr. 2013, pp. 258–274.

[13] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2014, pp. 639–654.

[14] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. ACM SIGMOD*, Paris, France, Jun. 2004, pp. 563–574.

[15] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD*, Madison, WI, USA, Jun. 2002, pp. 216–227.
[16] H. Kadhem, T. Amagasa, and H. Kitagawa, "A secure and efficient order preserving encryption scheme for relational databases," in *Proc. Int. Conf. Knowl. Manage. Inf. Sharing*, Valencia, Spain, Oct. 2010, pp. 25–35.
[17] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. ACM Symp. Oper. Syst. Principles*, Cascais, Portugal, Oct. 2011, pp. 85–100.
[18] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, May 2013, pp. 463–477.
[19] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.
[20] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," in *Computer Security—ESORICS* (Lecture Notes in Computer Science), vol. 8712. Chennai, India: Springer-Verlag, 2014, pp. 148–162.
[21] J. D. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Macau, China, Jun. 2002, pp. 617–624.
[22] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, Jan. 2010, pp. 40–47.
[23] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. R. Weippl, "Dark clouds on the horizon: Using cloud storage as attack vector and online slack space," in *Proc. USENIX Secur. Symp.*, Berkeley, CA, USA, Aug. 2011, pp. 65–76.
[24] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *Proc. Financial Cryptogr.*, Barbados, CA, USA, Mar. 2014, pp. 99–118.
[25] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. USENIX Secur. Symp.*, Washington, DC, USA, Aug. 2013, pp. 179–194.
[26] J. Li, X. Chen, M. Li, J. Li, P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 6, pp. 1615–1625, Nov. 2013.
[27] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 7881, T. Johansson and P. Q. Nguyen, Eds. Chennai, India: Springer, 2013, pp. 296–312.
[28] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8042, R. Canetti and J. A. Garay, Eds. Chennai, India: Springer, 2013, pp. 374–391.
[29] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, "A hybrid cloud approach for secure authorized deduplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1206–1216, May 2015.
[30] J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in *Proc. IEEE Conf. Commun. Netw. Secur.*, National Harbor, MD, USA, Oct. 2013, pp. 145–153.
[31] M. Bellare and S. Keelveedhi, "Interactive message-locked encryption and secure deduplication," in *Public-Key Cryptography* (Lecture Notes in Computer Science), vol. 9020, J. Katz, Ed. Berlin, Germany: Springer, 2015, pp. 516–538.
[32] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 2139. Berlin, Germany: Springer-Verlag, 2001, pp. 213–229.
[33] B. Lynn. *The Pairing-Based Cryptography Library*, accessed on Mar. 6, 2016. [Online]. Available: http://crypto.stanford.edu/pbc/
[34] *OpenSSL Cryptography and SSL/TLS Toolkit*, accessed on Mar. 6, 2016. [Online]. Available: https://www.openssl.org/
[35] B. Reed, "The height of a random binary search tree," *J. ACM*, vol. 50, pp. 306–332, May 2003.
[36] D. E. Knuth, "Optimum binary search trees," *J. Acta Inf.*, vol. 1, pp. 14–25, May 1971.
[37] K. Mehlhorn, "Nearly optimal binary search trees," *J. Acta Inf.*, vol. 5, pp. 287–295, May 1975.

Tao Jiang received the B.S. degree from the School of Computer Science and Technology, Shandong Jianzhu University, in 2009, and the M.S. degree from the School of Computer Science and Communication Engineering, Jiangsu University, in 2012. He is currently pursuing the Ph.D. degree with the School of Telecommunications Engineering, Xidian University. He is also a Visiting Student with the Complex Networks and Security Research Laboratory, Virginia Tech. His research interests include cryptography, network and data security, and cloud computing security.

Xiaofeng Chen (SM'15) received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 2003. He is currently a Professor with Xidian University. He has authored over 100 research papers in refereed international conferences and journals. His work has been cited over 4800 times at Google Scholar. His research interests include applied cryptography and cloud computing security. He is on the Editorial Board of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING SECURITY, the *Communication Networks*, and the *Telecommunication Systems*. He has served as the Program/General Chair or a Program Committee Member of over 30 international conferences.

Qianhong Wu (M'10) received the M.Sc. degree in applied mathematics from Sichuan University, Sichuan, China, in 2001, and the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2004. Since 2004, he has been an Associate Research Fellow with the University of Wollongong, Wollongong, Australia, an Associate Professor with Wuhan University, Wuhan, China, and a Senior Researcher with Universitat Rovira i Virgili, Tarragona, Catalonia. He has authored over 60 publications. His research interests include cryptography, information security and privacy, and ad hoc network security. He has been a main researcher or project holder/co-holder for more than ten Chinese-, Australian-, and Spanish-funded projects. He is a member of the International Association for Cryptologic Research. He has served on the program committees of several international conferences on information security and privacy.

Jianfeng Ma (M'12) received the B.Sc. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 1985, and the M.Sc. and Ph.D. degrees in computer software and communications engineering from Xidian University, Xi'an, in 1988 and 1995, respectively. From 1999 to 2001, he was a Research Fellow with the Nanyang Technological University of Singapore. He is currently a Professor and a Ph.D. Supervisor with the School of Computer Science and Technology, Xidian University. He is also the Director of the Shaanxi Key Laboratory of Network and System Security. His current research interests include information and network security, wireless and mobile computing systems, and computer networks. He has authored over 200 refereed articles in these areas and coauthored over ten books. He is a Senior Member of the Chinese Institute of Electronics and the China Computer Federation.

Willy Susilo (SM'01) is currently a Professor and Head of the School of Computing and Information Technology, University of Wollongong, where he is also the Director of the Centre for Computer and Information Security Research. He previously held the prestigious ARC Future Fellow awarded by the Australian Research Council (ARC). He has authored numerous publications in the area of digital signature schemes and encryption schemes. His main research interests include cyber security, cryptography, and information security. He has served as a Program Committee Member in dozens of international conferences.

Wenjing Lou (F'14) received the Ph.D. degree in electrical and computer engineering from the University of Florida, in 2003. From 2003 to 2011, she was a Faculty Member at the Worcester Polytechnic Institute. She has been a Professor with Virginia Tech since 2011. Since 2014, she has been with the U.S. National Science Foundation as the Program Director, where she is involved in the Networking Technology and Systems program and the Secure and Trustworthy Cyberspace program. Her current research interests focus on privacy protection techniques in networked information systems and cross-layer security enhancement in wireless networks, by exploiting intrinsic wireless networking and communication properties.