Hindawi
Security and Communication Networks
Volume 2020, Article ID 8837904, 13 pages
https://doi.org/10.1155/2020/8837904

Research Article

Order-Revealing Encryption Scheme with Comparison Token for Cloud Computing

Jingjing Guo[1] **and Jiacong Sun**[2]

[1]*State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, Shaanxi 710071, China*
[2]*Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen 518055, China*

Correspondence should be addressed to Jingjing Guo; jennifer.jing.sun@gmail.com

Received 24 September 2020; Revised 5 November 2020; Accepted 30 November 2020; Published 24 December 2020

Academic Editor: Prosanta Gope

Order-preserving encryption (OPE) is a basic paradigm for the outsourced database where the order of plaintexts is kept in ciphertexts. OPE enables efficient order comparison execution while providing privacy protection. Unfortunately, almost all the previous OPE schemes either require numerous rounds of interactions or reveal more information about the encrypted database (e.g., the most significant bit). Order-revealing encryption (ORE) as a generalization is an encryption scheme where the order of plaintexts can be evaluated by running a comparison algorithm. Therefore, it is desirable to design an efficient ORE scheme which addresses above efficiency and security issues. In this paper, we propose a noninteractive ORE scheme from prefix encoding and Bloom filter techniques. The proposed scheme is an encryption scheme where a cloud service provider cannot evaluate the order of plaintexts until a comparison token is provided. The security analysis illustrates that our scheme achieves ideal security with frequency hiding. Furthermore, we illustrate a secure range query scheme through designing an encrypted tree structure named PORE tree from the above ORE scheme. The PORE tree reveals the order between different nodes and leaves encrypted data items in the same node incomparable even after query execution. Finally, the experimental evaluation shows the high efficiency of the proposed ORE scheme and range query scheme.

1. Introduction

The rapid development of cloud computing enables resource-constrained data owners to outsource their data storage and computational tasks to a cloud service provider. The outsourced data could be attacked by outside attackers and the cloud service provider, for example, Verizon cloud leak and Equifax data breach. Thus, sensitive data should be encrypted before outsource, such as CryptDB, CipherCloud, and PRS server. Encryption is a promising approach for protecting sensitive data. However, traditional encryption breaks correlation of plaintexts and brings many difficulties in ordinary services, such as keyword search, numeric operations, and range query. Although some advanced cryptography primitives such as fully homomorphic encryption [1–3] and functional encryption [4–6] are introduced, the existing schemes are somewhat inefficient.

One method to enable efficient query services while meeting privacy requirement is to introduce an encryption scheme that allows only limited services. Searchable encryption [7–9] has been introduced to handle the keyword search in an encrypted database (EDB) [10, 11]. To execute range queries [12, 13], order-preserving encryption (OPE) [14–17] and its generalization order-revealing encryption (ORE) [18–20] have been proposed and developed.

OPE provides a property that the order of ciphertexts retains that of plaintexts. It has many applications in real-world scenarios, such as in SQL queries [16, 21–23], Web applications [24], CRM software [25], and others. The concept of OPE is firstly proposed by Agrawal et al. [14] while without security analysis. As an improvement, Boldyreva et al. [15] introduced a provable OPE scheme called BCLO that provides security models and rigorous analysis from cryptography point. The ideal security of OPE requires

that ciphertexts reveal nothing except the order of plaintexts. The BCLO scheme also points out that ideal security is unachievable if an OPE scheme is immutable and stateless. Therefore, the BCLO scheme is later shown to leak half of plaintext bits. To enhance security, Popa et al. [16] introduced a mutable OPE scheme named MOPE by leveraging binary tree. The OPE scheme achieves ideal security especially indistinguishability under an order chosen-plaintext attack (IND-OCPA). Unfortunately, the ideally secure scheme [16] requires $O(\log n)$-round of interactions.

Different from previous OPE schemes, Boneh et al. [18] proposed a generalization definition of the order-revealing encryption scheme which reveals the order of plaintexts. Due to the heavy computation burden of multilinear maps, the scheme [18] is impractical and remains a theoretical result for most applications. Chenette et al. [26] introduced a practical ORE scheme called CLWW that sacrifices the index of first different bit. Lewi and Wu [27] designed an ORE scheme named Lewi-Wu ORE that makes a trade-off between efficiency and security. But, the scheme [27] scales linearly with the size of domain space and reveals the index of first different block. As security improvement, Cash et al. [19] proposed a parameter-hiding order-revealing encryption by using asymmetric bilinear map to construct property-preserving hash (PPH) as the heart of the scheme. Unfortunately, the parameter-hiding ORE has a heavy computation burden and is vulnerable to frequency attacks.

1.1. Our Contributions. In this work, we focus on the construction of the order-revealing encryption scheme over the encrypted database system in cloud computing. The main contributions are as follows:

(i) We introduce a noninteractive ORE scheme over the encrypted database by leveraging Bloom filter and prefix encoding technologies. The security analysis demonstrates that our ORE scheme achieves ideal security and also hides frequency information of data items.

(ii) The introduced ORE scheme leaves encrypted data items incomparable until a comparison provided. Since the encrypted data items are incomparable, our ORE scheme can resist file-injection attacks [28].

(iii) We further present its application in secure range query on an encrypted database. Specifically, data items are stored in a tree structure (e.g., B-tree) to enhance search efficiency. We apply the introduced ORE scheme to encrypt the data items in the tree node and thus construct an encrypted tree, which is referred as PORE tree.

(iv) The presented PORE tree reveals order relation between different tree nodes and leaves encrypted data items in same node incomparable even after query execution.

This paper is an extended and enhanced version of the conference paper presented at ISPEC 2018 [17]. Compared with the conference version, we design a new ORE scheme which achieves ideal security with frequency hiding as an enhanced security of conference version with leakage function. The detailed description is depicted in Section 4. Furthermore, we modify the stored contents of PORE tree in Section 5. The proposed PORE tree achieves better security and efficiency. Moreover, formal security definition is presented in Section 3.2, and revised analysis according to the improved ORE scheme is introduced in Section 6. Finally, we add the experimental evaluation to evaluate the performance of MOPE [16], Lewi-Wu ORE [27], and our ORE scheme in Section 7.

1.2. Related Work. Order comparison [29] is one of the most popular operation in database-as-a-service (DaaS). In order to protect sensitive data, encryption is introduced as a prevailing consensus method. Numerous researchers have studied the problem of designing a secure and efficient encryption scheme which allows untrusted cloud servers to perform order comparison operation over ciphertexts. Due to heavy computation time and ciphertext size, fully homomorphic encryption and functional encryption are still impractical to handle order comparison [1, 2, 30]. Therefore, practical OPE schemes have proposed in the past few years [14–16, 31].

Agrawal et al. [14] formalized the definition of order-preserving encryption and designed an OPE scheme. It transforms plaintexts into ciphertexts which retain the order and follow a target distribution provided by the client. In [14], the authors gave no security analysis and their scheme only applies for the static system. As an improvement, Boldyreva et al. [15] proposed the ideal security definition and further proved that an OPE scheme with immutable and stateless is impossible to achieve ideal security. The illustrated BCLO scheme in [15] is indistinguishable from a ROPF and later shows that it leaks more than half of plaintext bits [32, 33]. Moving a step forward, Dyer et al. [34] introduced an OPE scheme by using approximate integer common divisor problem. However, Dyer et al.'s scheme achieves window one-wayness security. There exist other OPE schemes [21–23, 33] with weaker security guarantee by making some impractical assumptions.

Popa et al. [16] firstly introduced an ideally secure order-preserving encoding scheme named MOPE by using a balanced search tree and mutable ciphertexts. Mutable ciphertexts mean that encodings for several plaintext change with a deletion and insertion operation. The security on MOPE shows it is, in principle, possible to reveal no additional information besides orders. Kerschbaum [35] indicated each (deterministic) OPE scheme suffers a simple frequency attack from ciphertexts. For this reason, he provided a stronger security definition and indistinguishability under frequency analysis ordered chosen-plaintext attack (IND-FAOCPA). Based on random ciphertexts, the scheme [35] hides frequency information and achieves IND-FAOCPA security. Enlightened by a buffer tree [36], Roche et al. [37] constructed an order-preserving tree and further proposed a partial order-preserving encoding (POPE)

scheme which supports the insert-heavy database and leaks partial orders. Unfortunately, all above schemes limit by multiple round of interactions between clients and cloud servers.

Boneh et al. [18] provided a generalization definition of ORE. Different from traditional OPE, ORE reveals the order of plaintexts. They designed an ORE scheme by using multilinear maps. Because of the heavy computation burden of multilinear maps, the scheme [18] is impractical in most applications. The CLWW scheme [26] is the first practical ORE scheme by using PRFs. For each bit of data items, a keyed PRF takes this bit concatenated with all more significant bits as input. Thus, the ciphertexts include n elements which are the numerically sum of the keyed PRF and the next less significant bit. But, the CLWW scheme leaks the index of first different bit of two plaintexts. In order to balance security and efficiency, Lewi and Wu [27] introduced a generalization of the ORE scheme by using the left/right framework. The scheme of the small domain in [27] scales linearly with the size of message space and large domain reveals the first differing block. Based on bilinear map techniques, Cash et al. [19] proposed a more secure ORE scheme which leaks the equality pattern of the most significant bit.

Several related works about multiclient have been introduced. Xiao et al. [33] gave a method to extend the OPE scheme to multiclient scenario which only supports the OPE system not ORE. As an improvement, two multiclient order-revealing encryption (MC-ORE) schemes [38] have been proposed by using the design principle of the CLWW scheme. Obviously, MC-ORE schemes are proven secure with different leakage functions. Li et al. [20] introduced and designed the delegatable order-revealing encryption (DORE) scheme for the multiclient system. The DORE scheme reveals the index of the first different bit between two plaintexts. Latest, some attack schemes have been introduced, such as inference attack [39], file-injection attack [28], and multicolumn attack [40]. However, most of these attack schemes require auxiliary information which limits the application in practical.

1.3. Organization. The rest of this paper is organized as follows. In Section 2, we introduce some preliminaries. The system model and ORE definition are illustrated in Section 3. Then, in Section 4, we propose a PORE scheme by leveraging prefix encoding and Bloom filter techniques. We demonstrate its application on secure range query in Section 5. The security and efficiency analysis are illustrated in Section 6. In Section 7, we provide a performance evaluation of our scheme. Finally, the conclusions are presented in Section 8.

2. Preliminaries

In this section, we provide some preliminaries and definitions for the design of the ORE scheme and secure range query scheme.

2.1. Bloom Filter. As a space-efficient data structure, Bloom filter [41] is always used to test whether an element a is a member of a set S. The detailed description of Bloom filter is illustrated in the following.

Bloom filter (BF) contains an array of n cells which store 0 or 1 and k different hash functions $H_1, H_2, \ldots, H_k$. The hash function H_i is defined as H_i: $\{0,1\}^* \longrightarrow [1,n]$, $i = 1, 2, \ldots, k$. In the initial phase, we set all cells of the array to 0. When embedding an element w into the Bloom filter BF, we compute its hash functions $H_1(w), H_2(w), \ldots, H_k(w)$ and set all these chosen cells to 1. To test whether an element w belongs to a set S, we compute its hash functions and check whether the indicated cells of set S are all 1. If all these corresponding cells are 1, then w is an element of set S with allowable errors. Otherwise, w does not belong to the set S. The allowable error is called as false positive which satisfies $Pf = (1 - e^{-((km)/n)})^k$. It reaches its minimum value 2^{-k} when $r = \ln 2 * (n/m)$, where n is the size of the Bloom filter, k is the number of hash functions, and m denotes the number of elements in Bloom filter BF. An example of a Bloom filter is shown in Figure 1.

2.2. Prefix Encoding Technique. Prefix encoding technique converts the testing of whether an element d falls into a range $[a,b]$ to the testing of whether two sets have common elements [42]. The details are demonstrated as follows.

Given a t-bit number $d = d_1 d_2, \ldots, d_t$, we evaluate its prefix family as $F(d) = \{d_1 d_2, \ldots, d_t, d_1 d_2, \ldots, *, \ldots, d_1 * \cdots *, * * \cdots * \}$. It is easy to see that the size of $F(d)$ is $t + 1$. Given a range $[a,b]$, we compute the minimum cover set of prefixes such that the union of prefixes is $[a,b]$, denoted $S([a,b])$. The number of prefixes in $S([a,b])$ is at most $2t - 2$. In this condition, the testing of whether an element d belongs to a range $[a,b]$ can be translated to the testing $F(x) \cap S([a,b]) = \varnothing$. The element d belongs to a range $[a,b]$ if $F(x) \cap S([a,b]) \neq \varnothing$, otherwise not. As shown in Figure 2, the prefix family of 3 with 5-bit is $F(3) = \{00011, 0001*, 000**, 00***, 0****, *****\}$ and minimum cover set of [0, 6] is $S([0,6]) = \{000**, 0010*, 00110\}$. Owing to that $F(3) \cap S([0,6]) = \{000**\}$, it is easy to draw the conclusion that $3 \in [0,6]$.

3. Problem Formulation

3.1. System Model. As shown in Figure 3, the introduced ORE scheme consists of three main parties, data owner (DO), search user (SU), and cloud service provider (CSP).

(i) Data owner: the data owner is the entity who owns the database, encrypts the database, and delegates the encrypted database to a powerful cloud service provider. During the encryption phase, the data owner encrypts the data items from the symmetric encryption scheme to protect its privacy and generates the encrypted search indexes to facilitate data utilization.

(ii) Search user: search users receive system parameter and secret keys from the data owner and generate search requests according to their requirements. Then, search users send search requests with the

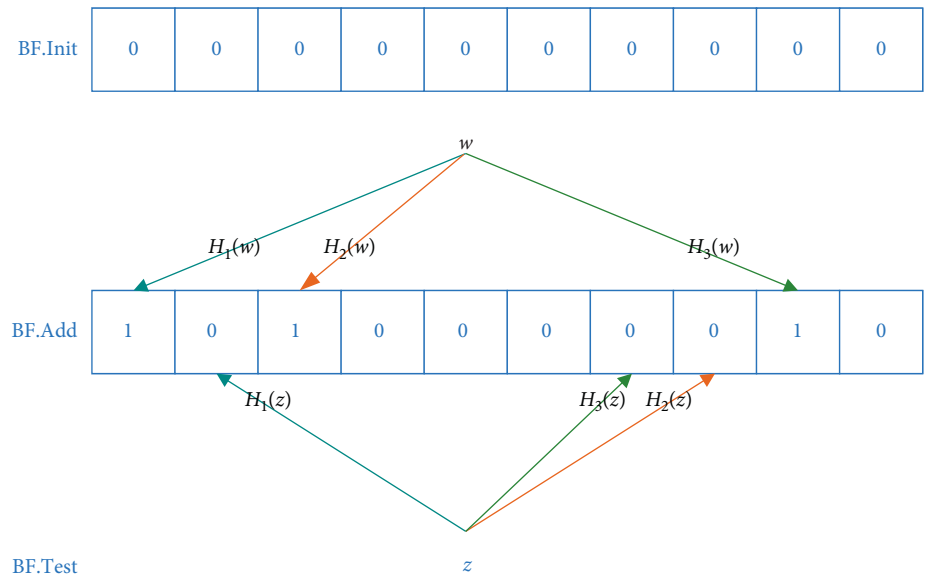

Figure 1: An example of Bloom filter, where w is added in the set and$z$ does not belong to the set.

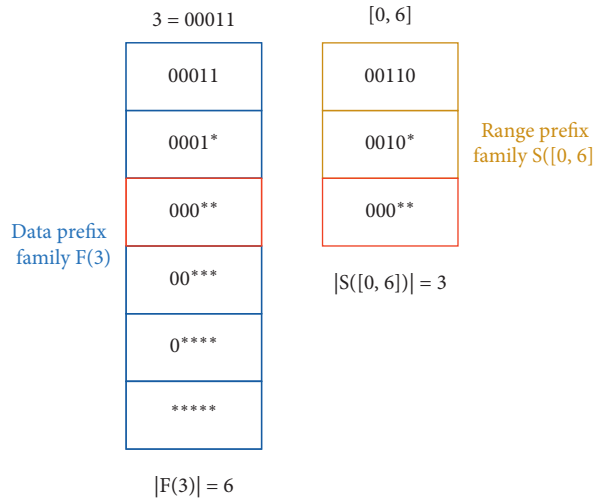

Figure 2: An example of prefix encoding technology.

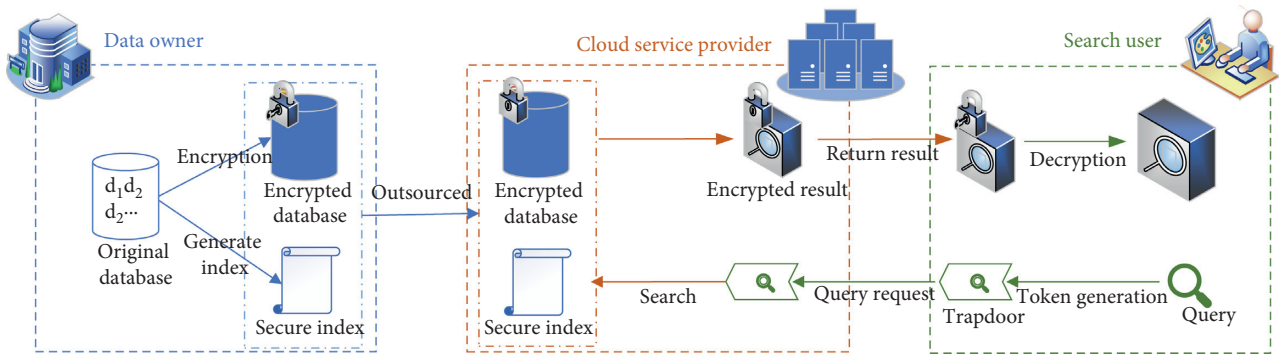

Figure 3: System model of the order-revealing encryption scheme.

corresponding trapdoor to the cloud service provider and receive the search results on ciphertext. Finally, they decrypt the ciphertexts and obtain the results.

(iii) Cloud service provider: the cloud service provider owns the powerful storage resources and unlimited computation resources. Thus, the cloud service provider always provides the storage, computation, and query services for the data owner and search user.

To protect the private sensitive information, the data owner encrypts the database, generates the indexes, and sends them to the cloud service provider. Search users compute the search trapdoor according to their request and send search trapdoor to the cloud service provider. The cloud service provider stores the encrypted database and then searches the encrypted database for the search users. In our paper, we assume the data owner and search users are fully trusted and the cloud service provider is "honest-but-curious." It means the cloud service provider follows protocol and returns answers faithfully but intends to learn additional information about the encrypted database. The assumption is very common in other work, such as [12, 13, 43].

3.2. Order-Revealing Encryption Scheme. An ORE scheme is defined by four probabilistic polynomial-time (PPT) algorithms ORE.KeyGen, ORE.Enc, ORE.TokenGen, and ORE.OrderComp. These algorithms are defined as follows:

(i) ORE.KeyGen $(\lambda) \longrightarrow$SK: DO runs this algorithm to output a secret key SK. The secret key will be secretly stored by the data owner. Furthermore, the data owner will send the secret key to search users through the access control technique.

(ii) ORE.Enc(SK, d) $\longrightarrow C_d$: DO runs the data encryption algorithm to encrypt data item d. The data encryption algorithm takes as input a secret key SK and a data item d and outputs the ORE ciphertext as C_d.

(iii) ORE.TokenGen (SK, a) $\longrightarrow \mathrm{TK}_a$: SU runs this algorithm to generate the comparison token. It takes as input a secret key SK and a data item a and outputs the comparison token TK_a.

(iv) ORE.OrderComp(C_d, TK_a) $\longrightarrow$ ans: CSP runs the order comparison algorithm to output the order relation. The algorithm takes as input the ciphertext C_d of data item d and comparison token TK_a for another data item. The order comparison algorithm outputs the order relation for data item d and a. If ans= 1, it means $a \geq d$; otherwise, ans= 0 means $a < d$.

In the following, the correctness and security definitions were introduced in the ORE scheme [27].

Definition 1 (Correctness). An ORE scheme over a well-domain D is correct if for SK$\leftarrow$ ORE.KeyGen(λ) such that, for any $a, b \in D$, if $a < b$, $C_a \leftarrow$ORE.Enc(SK, a), $\mathrm{TK}_b \leftarrow$ ORE.TokenGen(SK, b) then $1 \leftarrow$ ORE.OrderComp(C_a, TK_b).

Consider an experiment defined for the ORE scheme as follows. In the experiment, we introduce the random variable $win^{\mathscr{A},k}$ as the success of adversary $\mathscr{A}$.

The IND-OCPA indistinguishability experiment ExpA,OPE is as follows:

(i) The client runs the ORE.KeyGen(λ) algorithm to produce the secret key SK and chooses a random bit b;

(ii) The client and adversary $\mathscr{A}$ engage in polynomial rounds of interaction. At round i,

The adversary $\mathscr{A}$ chooses $\{v_i^0\}$ and $\{v_i^1\}$ in adaptive and sends both of them to the client;
After receiving $\{v_i^0\}$ and $\{v_i^1\}$, the client runs the encryption algorithm ORE.Enc(SK,) and returns $C_{v_i^b}$ to the adversary $\mathscr{A}$;

(iii) The adversary $\mathscr{A}$ outputs a bit b', its guess for b. The output is defined as 1 if $b' = b$ and 0 otherwise.

Definition 2 (Security). An ORE scheme is IND-OCPA secure, if for all PPT adversaries $\mathscr{A}$ and sequences $\{v_i^0\}_i$ and $\{v_i^1\}_i$ with the same order relations, there is a negligible function negl$(\cdot)$ satisfying

$$\Pr\left[\mathrm{win}^{\mathscr{A},k}\right] \leq \frac{1}{2} + \mathrm{negl}(k). \tag{1}$$

4. The Proposed ORE Scheme

4.1. High Description. Before describing the details of our ORE scheme, we introduce the construction principle in brief. Different from the existing works, we partition a ciphertext C_d into two parts E_d and I_d, where $E_d = \mathrm{DET.Enc}(sk_1, d)$ refers to the ciphertext which is produced by the symmetric encryption scheme DET with the strongest security (e.g., AES a DES) and I_d is an index which indicates the order of plaintexts. Comparison is only proceeded between the index I_d of one plaintext and the search token TK_a of another data item a. Thus, CSP proceeds the query request until a search token provided by SU.

In the following, we focus on the construction of index I_d. The order comparison between value d and a is equal to check whether a value d falls into a range $[0, a]$. By using prefix encoding technique which is proposed in [42], we compute the data prefix $F(d)$ and range prefix $S([0, a])$. Thus, the checking of whether a value d falls into a range $[0, a]$ converts to the checking of whether two sets $F(d)$ and $S([0, a])$ have common elements. The basic method is to check whether an element of one set falls into another set. Trivially, the checking is done by leveraging Bloom filter technique in [41].

Since frequency information leads some simple attacks which were pointed out by Naveed et al. [39], DO should also hide the frequency (i.e., hide equality) in the ciphertext E_d and the index I_d. Owing to DET is an IND-CPA secure

encryption scheme, the frequency is hidden in the ciphertext E_d. To hide the frequency in index, DO randomizes a data item d as $d' = d\|01\|r_1$ to break the equality before constructing the index.

4.2. The Main Construction. In this section, we introduce the ORE scheme from four processes: key generation, data encryption, token generation, and order comparison.

For the sake of clarity, we assume that all data items are w-bit and greater than 0, and DET = (DET.Key, DET.Enc, DET.Dec) is an IND-CPA secure encryption scheme. The details of our ORE scheme are illustrated as follows:

(i) ORE.KeyGen(λ): on input a security parameter λ, DO computes $sk_1 \leftarrow$ DET.Key(λ) as a secret key. It also chooses random numbers $k_1, k_2, \ldots, k_s$ as secret keys to compute hash $H(\cdot)$ for a Bloom filter BF. Thus, DO has the secret key $\mathrm{SK} = \{sk_1, sk_2\}$.

(ii) ORE.Enc(SK, d): on input a secret key $\mathrm{SK} = \{sk_1, k_1, k_2, \ldots, k_s\}$ and a data item d, DO firstly leverages privacy-preserving techniques to protect the privacy of data items. To keep the functionality of order comparison, DO illustrates an index for each data item. The following processes need to be performed:

(1) Data encryption: DO encrypts data items in the database as $E_d = \mathrm{DET.Enc}(sk_1, d)$.

(2) Index construction: to facilitate look-ups, DO constructs a secure Bloom filter BF_d as index structure I_d for each data item d. The secure Bloom filter BF algorithm is depicted in Algorithm 1.

(iii) ORE.TokenGen(SK, a): on input a secret key SK and a data item a, SU wants to compute the comparison token for it.

(1) Data randomization: for a data item a, SU uniformly samples a w-bit random number r and computes its randomization as $a' = a\|11\|r$.

(2) Token computation: the search user first computes range prefix $S([0, a||11||r]) = \{P_{a1}, P_{a2}, \ldots, P_{al}\}$ with l prefixes. Next, SU computes hashes for each prefix $P \in S([0, a||11||r])$ as $H(k_1 \| P), H(k_2 \| P), \ldots, H(k_s \| P)$.

The comparison token Tk_a for the data item a is a hash matrix:

$$\begin{aligned}&\{H(k_1 \| P_{a1}), H(k_2 \| P_{a1}), \ldots, H(k_s \| P_{a1}), \ldots,\\ &H(k_1 \| P_{al}), H(k_2 \| P_{al}), \ldots, H(k_s \| P_{al})\}.\end{aligned} \tag{2}$$

(iv) ORE.OrderComp(C_d, TK_a): after receiving the ciphertext C_d and comparison token TK_a, the cloud service provider checks whether $\mathrm{BF}_d(\mathrm{TK}_a) = 1$. The checking $\mathrm{BF}_d(\mathrm{TK}_a) = 1$ is done by checking whether there exists a row i in TK_a such that all the positions $H(v \cdot r \| H(k_1 \| P_{ai})), H(v \cdot r \| H(k_2 \| P_{ai})), \ldots, H(v \cdot r \| H(k_s \| P_{ai}))$ of Bloom filter BF_d are 1. That is, there exists prefix P in $S([0, a'])$ such that $P \in \mathrm{BF}_{d'}$. If $\mathrm{BF}_d(\mathrm{TK}_a) = 1$, set ans = 1; otherwise, ans = 0.

The ciphertext of data item d is $C_d = (E_d, I_d) = \mathbf{ORE.Enc}(\mathrm{SK}, d)$.

5. Encrypted Range Queries

In this section, we describe the application of the introduced ORE in range queries over the encrypted database. In our model, the resource-constraint data owner outsources the storage and computation tasks to the cloud service provider. Subsequently, search users submit the range query requests and obtain the query results. In outsourcing scenario, the cloud service provider needs to learn the order information for the range query. In this work, the above ORE scheme is introduced to help it in the operation of order comparison. To enhance the efficiency, the data owner stores data items in tree structure, such as B-tree. We describe the secure range query scheme with the proposed ORE scheme in the following algorithms

(i) RQ.Setup(λ): takes as input a security parameter, it outputs the secret key SK = {sk1, sk2}, where secret key SK is the same with that in the ORE scheme.

(ii) RQ.TreeBuild(D): takes as input a database D, the data owner stores the data items D in a B-tree data structure and obtains tree:

$$\Gamma = \{d_1, d_2, \ldots, d_n, P\}, \tag{3}$$

where $d_1, d_2, \ldots, d_n$ are data items in database D and P is a set of pointers to cover the parent-child relations of Γ tree.

(iii) RQ.TreeEnc(SK, Γ): takes as input the secret key SK and the tree Γ, the data owner runs the ORE.Enc algorithm in the ORE scheme to encrypt Γ tree as

$$\Gamma^* = \{C_{d_1}, C_{d_2}, \ldots, C_{d_n}, P\}, \tag{4}$$

where $C_{d_i} = \{E_{d_i}, I_{d_i}\} = \mathbf{ORE.Enc}(\mathrm{SK}d_i)$. Since Γ^* tree reveals order of data items in different nodes and leaves data items in same node incomparable, the encrypted Γ^* tree is named as PORE tree.

(iv) RQ.RangeQuery(SK, Γ^*, $[a, b]$): the range query algorithm consists of two phases. In the first phase, search users compute and send the encrypted ranges to CSP. In the second phase, CSP proceeds the search process over the encrypted tree structure Γ^*.

(1) Token generation: SU partitions the range $[a, b]$ into $[0, a]$ and $[0, b]$ two parts and uniformly samples two random numbers r_1 and r_2 to randomize the ranges $[0, a]$ and $[0, b]$ as

Input:
hash function: $\mathbf{H}(\cdot)$
secret keys: $\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_s$
data item: $\mathbf{d}$
Output:
–Bloom filter: $\mathbf{BF_d}$
(1) In order to hide the frequency, DO adds a random fractional part $\mathbf{r}$ for the data item $\mathbf{d}$ as $d' = \mathbf{d}\|01\|\mathbf{r}$, where $\mathbf{r}$ has the same number of bits with the data item $\mathbf{d}$ and d' is a $(2\mathbf{w}+2)$-bit number;
(2) DO computes prefix family of d' as $\mathbf{F}(d') = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{2\mathbf{w}+3}\}$;
(3) **for** $\mathbf{i}\leftarrow 1$ to $2\mathbf{w}+3$ **do**
(4) To remove the correlation among different nodes, DO chooses a random number$\mathbf{v}\cdot\mathbf{r}$ which has the same size with secret keys $\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_s$;
(5) **for** $j\leftarrow 1$ to s **do**
(6) DO first computes hash value $\mathbf{H}(\mathbf{k}_j \| \mathbf{P}_i)$ and then sets 1 on the position $\mathbf{H}(\mathbf{v}\cdot\mathbf{r} \| \mathbf{H}(\mathbf{k}_j \| \mathbf{P}_i))$;
(7) **end for**
(8) **end for**
(9) After inserting the prefix family, we have a secure Bloom filter $\mathbf{BF_d}$.

Algorithm 1: Secure BF.

$[0, a\|00\|r_1]$ and $[0, b\|11\|r_2]$. Next, search users proceed the token computation algorithm in **ORE.TokenGen** to generate search token $\mathrm{TK}_{[0,a]}$ and $\mathrm{TK}_{[0,b]}$. Finally, search users send the search token $\mathrm{TK}_{[0,a]}$ and $\mathrm{TK}_{[0,b]}$ to CSP.

(2) After receiving search token, CSP searches the tree from root to the leaf and finds the leftmost and the rightmost leaf node intersecting with the range. The details of the search algorithm are illustrated in Algorithm 2.

(v) RQ.Update(SK,Γ^*, d): to update the outsourced database, the data owner generates the ciphertext E_d, index I_d, and token TK_d and submits (E_d, I_d, TK_d) to the cloud service provider CSP. After receiving the update request, CSP updates the database by leveraging (E_d, I_d, TK_d). In the following, we introduce the insertion, deletion, and modification algorithms in detail:

(1) Suppose that DO wants to insert a data d into the encrypted database. DO firstly computes and sends the ciphertext E_d, index I_d, and comparison token TK_d to CSP. Upon receiving the insertion token, CSP takes this new data as a split point and inserts it into the parent node $\overline{v}$. For a data item (E_{d_i}, I_{d_i}) in the leaf node v, CSP checks whether data item d_i satisfying $\mathrm{BF}_{d_i}(\mathrm{TK}_{[0,d]}) = 1$. CSP stores ($E_{d_i}$, I_{d_i}) into the left node v_1 if $\mathrm{BF}_{d_i}(\mathrm{TK}_{[0,d]}) = 1$. Otherwise, CSP stores it into the right node v_2.

(2) Suppose that DO wants to delete a data d from the encrypted database. The deletion algorithm finds data items in the range $[d, d]$, which is similar to the range query algorithm and deletes these data items.

(3) Suppose that DO wants to modify a data item d to $\overline{d}$ in the encrypted database. The modification is converted to deletion and insertion operation. Thus, it can be done using above two steps.

(vi) RQ.Dec(SK, Res*): on input the secret key SK and encrypted results Res$*$, DO firstly runs DET.Dec to decrypt Res* and then removes random numbers to obtain final results.

6. Analysis

In this section, we demonstrate the security analysis and efficiency analysis.

6.1. Security Analysis

Theorem 1. *The proposed ORE scheme over a well-domain D is correct.*

Proof. Suppose that r_1 and r_2 are two w-bit random numbers, C_d is the ciphertext for data item$d$, and TK_a is the comparison token for data item a. We prove the correctness through $d \le a$ if and only if ans = 1 without considering the false positive of Bloom filter.

First, we prove if $d \le a$, then ans = 1. If $d \le a$, we have the relation$d' < a'$. Owing to the fact that a data item d falls into a range $[a, b]$ if and only if $F(d) \cap S([a, b]) \neq \phi$, we can compute and draw the conclusion that $F(d') \cap S([0, a']) \neq \phi$. That means, $\mathrm{BF}_d(\mathrm{TK}_a) = 1$, namely, ans = 1.

Moreover, if we do not take false positive of Bloom filter into consideration, we can prove the correctness. That is, if ans = 1, then $\mathrm{BF}_d(\mathrm{TK}_a) = 1$. Since $\mathrm{BF}_d(\mathrm{TK}_a) = 1$, we have the conclusion that there exists a row$i$ in TK_a such that all the corresponding positions of Bloom filter BF_d are equal to 1. Namely, $F(d') \cap S([0, a']) \neq \phi$ that means $d' \le a'$. Because $d' = d||01||r_1$ and$a' = a\|11\|r_2$, we have the conclusion that $d \le a$ (if $d > a$, then $d\prime > a\prime$). □

Theorem 2. *The proposed secure range query scheme over a well-domain D is correct.*

(i) **Input:**
(ii) encrypted PORE tree: Γ^*
(iii) search token: $\mathbf{TK}_{[a,b]} = (\mathbf{TK}_{[0,\mathbf{a}]}\mathbf{TK}_{[0,\mathbf{b}]})$
(iv) **Output:**
(v) encrypted results: $\mathbf{Res}^*$
(1) CSP proceeds the search algorithm twice, once for $\mathbf{TK}_{[0,\mathbf{a}]}$ to find the leftmost leaf node and once for $\mathbf{TK}_{[0,\mathbf{b}]}$ to find the rightmost leaf node;
(2) **for v←root to leaf no de do**
(3) CSP checks whether $\mathbf{BF_d}(\mathbf{TK}_{[0,\mathbf{a}]}) = 1$ by checking whether there exists a row $\mathbf{i}$ in matrix $\mathbf{TK}_{[0,\mathbf{a}]}$ such that the positions $\mathbf{H}(\mathbf{v}\cdot\mathbf{r} \parallel \mathbf{H}(\mathbf{k}_1 \parallel \mathbf{P_{ai}}), \mathbf{H}(\mathbf{v}\cdot\mathbf{r} \parallel \mathbf{H}(\mathbf{k}_2 \parallel \mathbf{P_{ai}}), \ldots, \mathbf{H}(\mathbf{v}\cdot\mathbf{r} \parallel \mathbf{H}(\mathbf{k}_s \parallel \mathbf{P_{ai}})$ of $\mathbf{BF_d}$ are equal to 1;
(4) Afterwards, CSP finds the largest value $\mathbf{d}$ satisfying $\mathbf{BF_d}(\mathbf{TK}_{[0,\mathbf{a}]}) = 1$ and updates its corresponding child as the new value for $\mathbf{v}$;
(5) **end for**
(6) CSP obtains the leftmost leaf node and the rightmost leaf node;
(7) Finally, CSP returns results $\mathbf{Res}^*$ which include data items in leaf nodes between the two leaf nodes, in the leftmost leaf node satisfying $\mathbf{BF_d}(\mathbf{TK}_{[0,\mathbf{a}]}) = 0$ and in the rightmost leaf node satisfying $\mathbf{BF_d}(\mathbf{TK}_{[0,\mathbf{a}]}) = 1$.

Algorithm 2: Tree search.

Proof. Suppose that a data item $d \in D$, a range $[a,b]$, and their randomization $d' = d\|01\|r_1$, $a' = a\|00\|r_2$ and $b' = b\|11\|r_3$, where r_1, r_2, and r_3 are three w-bit random numbers. We proof the correctness through $d \in [a,b]$ if and only if $d \in \mathrm{Res}$.

First, we prove if $d \in [a,b]$, then $d \in \mathrm{Res}$. If $d \in [a,b]$, we have the relation $a' < d' < b'$. Owing to the fact that for a data item d and range $[a,b]$, $x \in [a,b]$ if and only if $F(d) \cap S([a,b]) \neq \phi$, we can compute $\mathrm{BF}_{d'}(M[0,a']) = 0, \mathrm{BF}_{d'}(M[0,b']) = 1$. That means, $C_d \in \mathrm{Res}^*$, namely, $d \in \mathrm{Res}$.

Moreover, if we do not take false positive of Bloom filter into consideration, we can prove the correctness. That is, if $d \in \mathrm{Res}$, then $d \in [a,b]$. If $d \in \mathrm{Dec}(sk, \mathrm{Res}^*)$, we can draw the conclusion that

$$\begin{aligned} C_d &\in \mathrm{Res}^*, \\ \mathrm{BF}_{d'}(M[0,a']) &= 0, \\ \mathrm{BF}_{d'}(M[0,b']) &= 1. \end{aligned} \tag{5}$$

That is, $d' \notin [0,a']$ and $d' \in [0,b']$, namely,

$$\begin{aligned} d||01||r_1 &\notin [0,a||00||r_2], \\ d\|01\|r_1 &\in [0,b\|11\|r_3]. \end{aligned} \tag{6}$$

After derandomization, we can draw the conclusion that

$$d \notin [0,a), d \in [0,b]. \tag{7}$$

That means $d \in [a,b]$. □

Theorem 3. *The proposed ORE scheme is IND-OCPA secure in the random oracle model.*

Proof. For the convenience of understanding, we first introduce the concept of computationally indistinguishable. Let $X = \{X_k\}_k$ and $Y = \{Y_k\}_k$ be ensembles of distributions. X and Y are said to be computationally indistinguishable (written $X^C \approx Y$), if for all PPT adversaries $\mathscr{A}$,

$$\Pr[(X_k) = 1] - \Pr[(Y_k) = 1] = \mathrm{negl}(k). \tag{8}$$

Next, we will prove the theorem in the following hybrid games:

(i) Hybrid 1: The IND-OCPA game for an adversary $\mathscr{A}$ and the proposed scheme. Let $H_1^{\mathscr{A}}(\lambda)$ be the random variable indicating the output of $\mathscr{A}$ in the IND-OCPA game, namely, $\mathscr{A}$'s guess bit.
(ii) Hybrid 2: The IND-OCPA game for an adversary $\mathscr{A}$, where the proposed scheme is revised to instead the DET with a random oracle O (a random oracle refers to an oracle which outputs a random value r when given a value v for the first time, and the same value r returns when v is given again.). Let $H_2^{\mathscr{A}}(\lambda)$ be the random variable indicating the output of $\mathscr{A}$ in the IND-OCPA game, namely, $\mathscr{A}$'s guess bit.
(iii) Hybrid 3: The IND-OCPA game for adversary $\mathscr{A}$, where the scheme in Hybrid 2 is revised to instead hash function with a random oracle O. Let $H_3^{\mathscr{A}}(\lambda)$ be the random variable indicating the output of $\mathscr{A}$, namely, $\mathscr{A}$'s guess bit.

Because DET is a pseudo-random function, it is obvious that $H_1^{\mathscr{A}}(\lambda)^C \approx H_2^{\mathscr{A}}(\lambda)$. We give a proof by contradiction for this conclusion. If these two hybrids are distinguishable, then we construct a simulator $\mathscr{D}_1$ which distinguishes pseudo-random numbers from random numbers.

The distinguisher $\mathscr{D}_1(\{r_k\})$:

(1) Adversary $\mathscr{A}$ is given a security parameter and outputs ($\{v_k^0\}$ and $\{v_k^1\}$) with the same order relation;
(2) A uniform bit $b \leftarrow \{0,1\}$ is chosen, and then a ciphertext $\{C_k^b\} = \{r_k\}$ is computed and given to the adversary $\mathscr{A}$;
(3) The adversary $\mathscr{A}$ outputs a bit b', its guess for b. The distinguisher $\mathscr{D}_1$ outputs 1 if $b' = b$ (meaning pseudo-random sequence) and 0 (meaning random sequence) otherwise.

If adversary $\mathscr{A}$ can distinguish these two hybrids, namely,

$$\left|\Pr\left[\mathscr{A}\left(\mathrm{DET.Enc}\left(sk_1, v_k^b\right)\right)=1\right]-\Pr\left[\mathscr{A}\left(r_k\right)=1\right]\right|>\mathrm{negl}(k), \tag{9}$$

then we conclude that

$$\begin{aligned}&\Pr\left[\mathscr{D}_1=1\right]\\&=\frac{1}{2}\Pr\left[\mathscr{D}_1=1|b=0\right]+\frac{1}{2}\Pr\left[\mathscr{D}_1=1|b=1\right]\\&=\frac{1}{2}+\frac{1}{2}\cdot\left(\Pr\left[\mathscr{A}\left(\mathrm{DET.Enc}\left(sk_1, v_k^b\right)\right)=1\right]-\Pr\left[\mathscr{A}\left(r_k\right)=1\right]\right)\\&\geq\frac{1}{2}+\frac{1}{2}\cdot\mathrm{negl}(k).\end{aligned} \tag{10}$$

Thus, we can draw the conclusion that $H_1^{\mathscr{A}}(\lambda)^C \approx H_2^{\mathscr{A}}(\lambda)$. Since $H(\cdot)$ is a pseudo-random function, it draws the same conclusion that$H_2^{\mathscr{A}}(\lambda)^C \approx H_3^{\mathscr{A}}(\lambda)$. Following, we prove the probability adversary $\mathscr{A}$ guesses correctly in Hybrid 3 is 1/2. That is, the winning advantage of $\mathscr{A}$ can be at most $1/(2+\mathrm{negl}(k))$. The data owner stores $C_d = (E_d, I_d)$ in the cloud service provider for a data item d. Because E_d is replaced with a random oracle, the adversary $\mathscr{A}$ cannot distinguish the ciphertexts of v_i^0 and v_i^1, namely, $E_{v_i^0}{}^C \approx E_{v_i^1}$.

Moreover, recall that every value d is assigned a different random number $v.r$, and random oracle O uses $v.r$ to compute hash functions for the secure Bloom filter BF_d. Thus, given two different Bloom filters BF_i and BF_j, it is infeasible for any PPT adversary $\mathscr{A}$ to distinguish whether a same string or two different strings are mapped to them. From this, it is easy to draw the conclusion that

$$BF_{v_i^0}{}^C \approx BF_{v_i^1}.\text{Namely}, I_{v_i^0}{}^C \approx I_{v_i^1}. \tag{11}$$

As a result, the adversary $\mathscr{A}$ cannot distinguish with nonnegligible probability. That is,

$$\Pr\left[\mathrm{win}^{\mathscr{A},k}\text{in Hybrid 3}\right]\leq\frac{1}{2}+\mathrm{negl}(k). \tag{12}$$

We complete the proof of Theorem 3. □

6.2. Efficiency Analysis. For the convenience of discussion, some marks are introduced in this section. We denote by E/D an encryption/decryption algorithm of the DET scheme with IND-CPA secure, F is a PRF, H is a hash function, r is the number of hashes in a Bloom filter, w is the bits of our data item, n is the size of the database, N is the size of message space, and t is the size of results. We omit other operations such as comparison of plaintexts and uniformly random permutation in Lewi-Wu ORE.

We now describe the efficiency analysis among MOPE [16], Lewi-Wu ORE [27], and the proposed range query scheme with PORE tree, shorted for the PORE scheme. The security in Table 1 demonstrates that all schemes achieve IND-OCPA security which reveals nothing besides the order relation. Furthermore, the PORE scheme hides data frequency and data items in leaf nodes of our PORE scheme which remain incomparable. Stored information in MOPE [16] only consists of the ciphertexts, that is, produced by an IND-CPA secure DET scheme. Therefore, the order comparison is achieved by the interaction with a DO/SU. In this condition, there is about $O(\log n)$-round communication between DO/SU and CSP during the range query and update operation. Ciphertexts in Lewi-Wu ORE [27] consist of the left ciphertexts and the right ciphertexts. It is possible to proceed the comparison algorithm with the left of one ciphertext and the right of another ciphertext. To enhance the storage efficiency, CSP only stores the left ciphertexts (about $\log n$ bits long) in the sorted order instead of the complete ciphertexts (about $n\log 3+\log n$ bits long). When proceeding a range query or an update operation, DO generates the right ciphertexts and sends them to the CSP.

The basic idea of our PORE is inspired by Lewi-Wu ORE. We split stored information into a ciphertext produced by DET and an index I_d. The index I_d is a Bloom filter storing$F(d)$ with the size of $w+1$. Compared with the MOPE and Lewi-Wu ORE scheme, the cost of encryption in our PORE scheme is a little higher. While encryption is one-time in the initialization phase, the cost is acceptable. In range query and update phases, CSP obtains order comparison by communicating with the DO and SU in MOPE. SU decrypts ciphertexts and returns the order information. The cost in Lewi-Wu ORE [27] is linear with the message space for the DO/SU and logarithm with the database for CSP. MOPE and Lewi-Wu ORE have a better efficiency for CSP in both phases. Both schemes do not use the tremendous compute power of CSP violating the definition of cloud computing. Finally, we observe from decryption cost that all these schemes have a high efficiency.

7. Performance Evaluation

In this section, we introduce a formal experimental simulation among MOPE [16], Lewi-Wu ORE [27], and the proposed PORE scheme in order to test the practical utility. Specifically, the code is run with Python3 language on a machine with an Intel Xeon (R) CPU i7-8565U processor running at 16 GHz and 1 T memory. We instantiate the hash function with SHA-256 in the Lewi-Wu ORE scheme. The symmetric cipher used is AES-ECB-128 as provided by the PyCrypto library, and the scalable Bloom filer is provided in pybloom live library. Furthermore, we set error rate of Bloom filter as 0.001 and storage limitation $L = 4$ of B-tree to store ciphertexts.

7.1. Database Size. In the following experiment, we use the Gowalla database which lists 6,442,890 check-in locations collected from 196,591 users. We sample 1,000,000 data items of 48 bit in the database to test the performance of the MOPE and PORE scheme. Figure 4 demonstrates that Lewi-Wu ORE [27] is only suitable in the small domain. In this condition, we cut 12 bit from data items in the Gowalla database to simulate the performance of search and update.

Table 1: Efficiency comparison among MOPE, Lewi-Wu ORE, and our proposed PORE.

	Entity	MOPE	Lewi-Wu ORE	PORE
Security		IND-OCPA	IND-OCPA	IND-OCPA
Interaction		$\log n$	0	0
Frequency hiding		No	No	Yes
Encryption	DO	nE	nF	$nE + nr(w+1)H$
Range query	SU	$2\log n \cdot D$	2NF + 2NH	$4r(w-1)H$
	CSP	—	$2\log n \cdot H$	$4r\log n(w-1)H$
Insertion	DO	$E + \log n \cdot D$	$(N+1)F$ + NH	$E + 2r(w-1)H$
	CSP	—	$\log n \cdot H$	$2r\log n(w-1)H$
Deletion	DO	$\log n \cdot D$	NF + NH	$4r(w-1)H$
	CSP	—	$\log n \cdot H$	$4r\log n(w-1)H$
Decryption	SU	$t\,D$	tF	$t\,D$

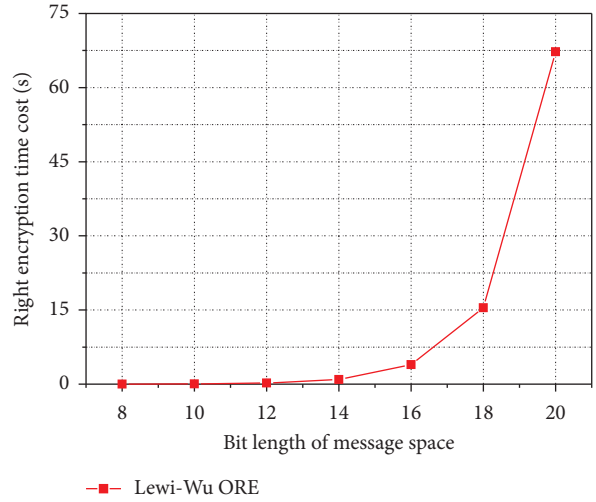

Figure 4: The average encryption time cost of the Lewi-Wu ORE scheme.

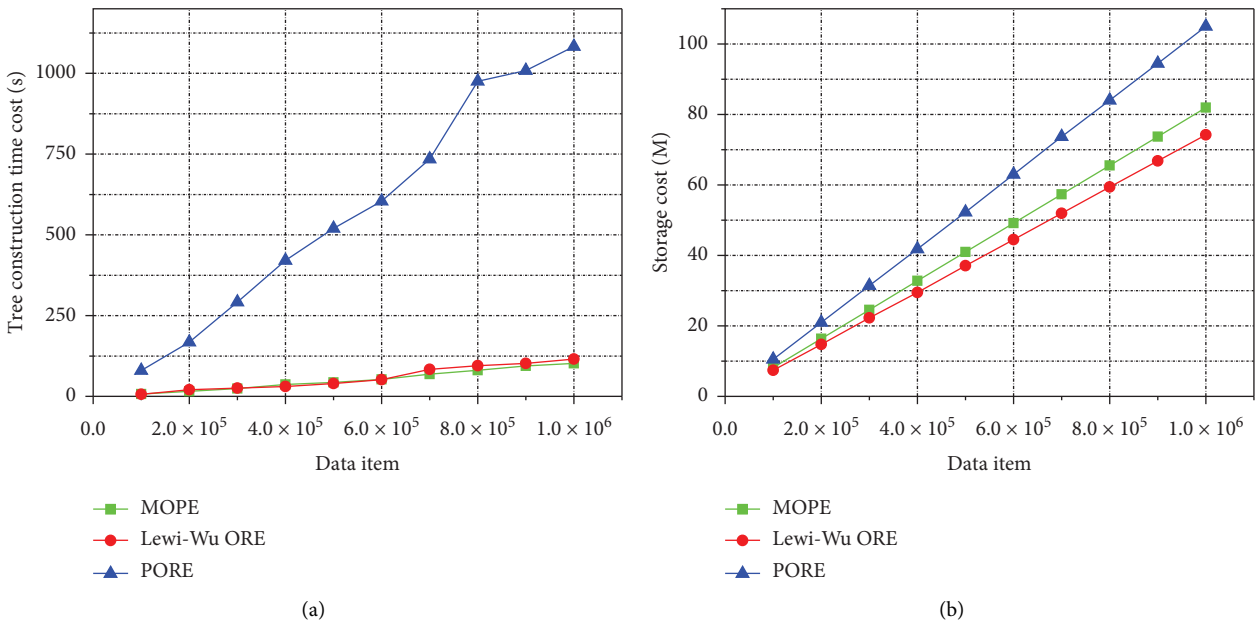

Figure 5: The time cost (a) and storage cost (b) of tree construction. Lower is better. The bit length of data item is 48 bit for MOPE and PORE and 12 bit for Lewi-Wu ORE.

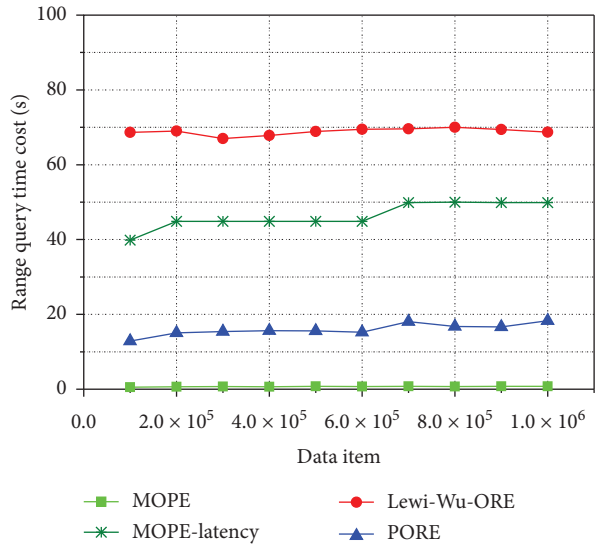

Figure 6: The time cost of range query. Lower is better. The number of range query is 1000 for MOPE, MOPE-latency, and PORE and 100 for Lewi-Wu ORE.

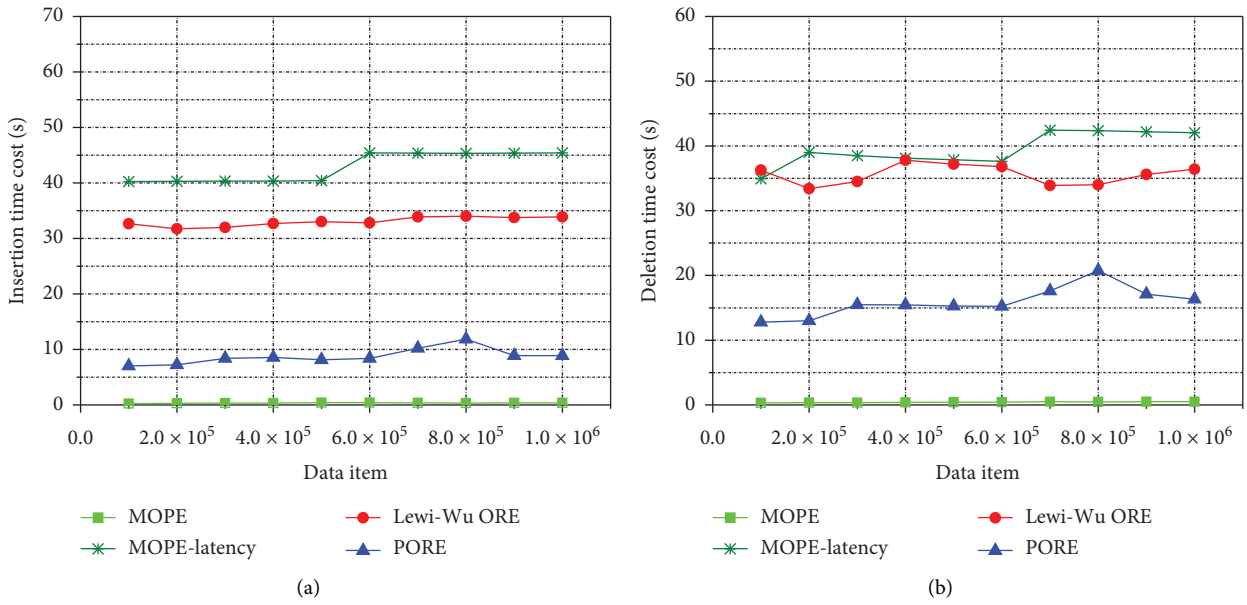

Figure 7: The time cost of (a) insertion and (b) deletion. Lower is better. The number of insertion and deletion is 1000 for MOPE, MOPE-latency, and PORE and 100 for Lewi-Wu ORE.

7.2. Network. For these experiments, we simulate all the data owner, search users, and cloud service provider on the same machine. Therefore, we test performance by measuring the theoretical communication cost instead of the realistic network with latency and bandwidth restrictions. In the theoretical experimental simulation, we assume that the network is slower than 5 ms of latency and 20 Mbps bandwidth.

Figures 5(a) and 5(b) show the time cost and storage cost on tree construction of MOPE, Lewi-Wu ORE, and our PORE scheme. We can easily see that time cost of MOPE is similar to that of Lewi-Wu ORE since the AES-ECB-128 is used to encrypt data items in MOPE and to instantiate the PRF in Lewi-Wu ORE. Furthermore, the time cost from 80s at 100,000 data items and 1082s at 1,000,000 data items of our PORE scheme is greater than that of the MOPE and Lewi-Wu ORE scheme. Meanwhile, the storage cost from 10.5 M to 105 M of the PORE scheme is a little greater than that of the MOPE and Lewi-Wu ORE scheme, from 8.2 M to 82M and 7.5 M to 75 M, respectively. The time cost and storage cost are acceptable of our PORE scheme because of one-time on tree construction.

In our main experiments, we test the performance of range query and update operation (insertion and deletion)

with 1,000 operations in MOPE and our PORE scheme and that with 100 operations in the Lewi-Wu ORE scheme. The time cost of range query is shown in Figure 6. It demonstrates that our PORE scheme is far less than MOPE-latency and the Lewi-Wu ORE scheme. Furthermore, our scheme achieves about 18 ms per-range query on the database with million data items vs. about 50 ms per-operation for MOPE-latency with communication latency.

Figures 7(a) and 7(b) show the time cost on insertion and deletion operation. As these figures present, the cost is dominated by communication on range query and insertion and deletion of the MOPE-latency scheme with 5 ms latency. The runtime is linear with the round of communication. Moreover, the time cost of MOPE-latency and Lewi-Wu ORE is about 5x greater than our scheme on insertion and 2x on deletion operation. We can see that our scheme is well-suited for range query and update operation.

8. Conclusions

In this paper, we introduce a noninteractive order-revealing encryption scheme with comparison token to execute privacy-preserving order comparison in cloud computing. The introduced ORE scheme requires that the cloud service provider could not perform order comparison until a search token provided. The security analysis shows that our ORE scheme achieves IND-OCPA security. Furthermore, we design a secure range query over the encrypted database through designing PORE tree structure from the proposed ORE scheme. The designed PORE tree reveals partial order information of plaintexts and leaves results incomparable even after query execution. Finally, the experimental result shows the high efficiency of our PORE scheme based on the designing of the ORE scheme.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

References

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the International Symposium of Theory of Computing (STOC)*, pp. 169–178, Bethesda, MD, USA, May 2009.

[2] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Proceedings of the International Conference on Cryptology (CRYPTO)*, pp. 850–867, Santa Barbara, CA, USA, August 2012.

[3] M. Li, "Leveled certificateless fully homomorphic encryption schemes from learning with errors," *IEEE Access*, vol. 8, pp. 26749–26763, 2020.

[4] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: definitions and challenges," in *Proceedings of the International Conference on Theory of Cryptography (TCC)*, pp. 253–273, Providence, RI, USA, March 2011.

[5] S. Goldwasser, S. D. Gordon, V. Goyal et al., "Multi-input functional encryption," in *Proceedings of the International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 578–602, Copenhagen, Denmark, May 2014.

[6] A. Jain, N. Manohar, and A. Sahai, "Combiners for functional encryption, unconditionally," in *Proceedings of the International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 141–168, Zagreb, Croatia, May 2020.

[7] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the International Conference on Security and Privacy (SP)*, pp. 44–55, Berkeley, CA, USA, May 2000.

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky et al., "Public key encryption with keyword search," in *Proceedings of the International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 506–522, Interlaken, Switzerland, May 2004.

[9] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3293–3303, 2015.

[10] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 546–556, 2015.

[11] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3184–3195, 2016.

[12] X. Wang, J. Ma, X. Liu et al., "Search me in the dark: privacy-preserving boolean range query over encrypted spatial data," in *Proceedings of the International Conference on Computer Communications (INFOCOM)*, pp. 2253–2262, Toronto, Canada, July 2020.

[13] Y. Peng, L. Wang, J. Cui et al., "LSRQ: A lightweight and forward-secure range query on geographically encrypted data," *IEEE Transactions on Dependable and Secure Computing*, vol. 99, Article ID 2974218, 2020.

[14] R. Agrawal, J. Kiernan, R. Srikant et al., "Order-preserving encryption for numeric data," in *Proceedings of the International Conference on Management of Data (SIGMOD)*, pp. 563–574, Paris, France, June 2004.

[15] A. Boldyreva, N. Chenette, Y. Lee et al., "Order-preserving symmetric encryption," in *Proceedings of the International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 224–241, Cologne, Germany, April 2009.

[16] R. A. Popa, F. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proceedings of the International Conference on Security and Privacy*, pp. 463–477, (SP), Berkeley, CA, USA, May 2013.

[17] J. Guo, J. Wang, Z. Zhang et al., "An almost non-interactive order preserving encryption scheme," in *Proceedings of the International Conference on Information Security Practice and Experience (ISPEC)*, pp. 87–100, Tokyo, Japan, September 2018.

[18] D. Boneh, K. Lewi, M. Raykova et al., "Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation," in *Proceedings of the International Conference on Theory and Applications of Cryptographic*

Techniques (EUROCRYPT), pp. 563–594, Sofia, Bulgaria, April 2015.

[19] D. Cash, F. Liu, A. ONeill et al., "Parameter-hiding order revealing encryption," in *Proceedings of the International Conference on Theory and Application of Cryptology and Information Security (Asia CRYPT)*, pp. 181–210, Brisbane, Australia, December 2018.

[20] Y. Li, H. Wang, and Y. Zhao, "Delegatable order-revealing encryption," in *Proceedings of the International Conference on Computer and Communications Security (AsiaCCS)*, pp. 134–147, Auckland, New Zealand, July 2019.

[21] H. Kadhem, T. Amagasa, and H. Kitagawa, "A secure and efficient order preserving encryption scheme for relational databases," in *Proceedings of the International Conference on Knowledge Management and Information Sharing (KMIS)*, pp. 25–35, Valencia, Spain, October 2010.

[22] D. Liu and S. Wang, "Programmable order-preserving secure index for encrypted database query," in *Proceedings of the International Conference on Cloud Computing (CLOUD)*, pp. 502–509, Honolulu, HI, USA, June 2012.

[23] D. Liu and S. Wang, "Nonlinear order preserving index for encrypted database query in service cloud environments," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 13, pp. 1967–1984, 2013.

[24] Perspecsys, "The PRS Server: Data Protection for Cloud Applications," http://www.perspecsys.com/perspecsys-cloud-protection-gateway/.

[25] F. Y. Rashid, "Salesforce.com Acquires SaaS Encryption Provider Navajo Systems", eWeek.Com, 2011. https://www.cioinsight.com/c/a/Latest-News/Salesforcecom-Acquires-SaaS-Encryption-Provider-Navajo-Systems-331154.

[26] N. Chenette, K. Lewi, S. A. Weis et al., "Practical order-revealing encryption with limited leakage," in *Proceedings of the International Conference on Fast Software Encryption (FSE)*, pp. 474–493, Bochum, Germany, March 2016.

[27] K. Lewi and D. J. Wu, "Order-revealing encryption: new constructions, applications, and lower bounds," in *Proceedings of the International Conference on Computer and Communications Security (CCS)*, pp. 1167–1178, Vienna, Austria, October 2016.

[28] X. Wang and Y. Zhao, "Order-revealing encryption: file-injection attack and forward security," in *Proceeding of International Conference on European Symposium on Research in Computer Security (ESORICS)*, pp. 101–121, Barcelona, Spain, September 2018.

[29] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proceedings of the International Conference on Theory of Cryptography (TCC)*, pp. 535–554, Amsterdam, Netherlands, February 2007.

[30] X. Boyen, X. Fan, and E. Shi, "Adaptively secure fully homomorphic signatures based on lattices," *IACR Cryptology*, Article ID 916, 2014.

[31] J. Furukawa, "Request-based comparable encryption," in *Proceedings of the International Conference on European Symposium on Research in Computer Security (ESORICS)*, pp. 129–146, Egham, UK, September 2013.

[32] A. Boldyreva, N. Chenette, and A. ONeill, "Order-preserving encryption revisited: improved security analysis and alternative solutions," in *Proceedings of the International Conference on Cryptology (CRYPTO)*, pp. 578–595, Santa Barbara, CA, USA, August 2011.

[33] L. Xiao, I. Yen, and D. T. Huynh, "Extending order preserving encryption for multi-user systems," *IACR Cryptology*, Article ID 192, 2012.

[34] J. Dyer, M. Dyer, and J. Xu, "Order-preserving encryption using approximate integer common divisors," in *Proceedings of the International Conference on Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pp. 257–274, Oslo, Norway, September 2017.

[35] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *Proceedings of the International Conference on Computer and Communications Security (CCS)*, pp. 656–667, Denver, CO, USA, October 2015.

[36] L. Arge, "The buffer tree: a technique for designing batched external data structures," *Algorithm*, vol. 37, no. 1, pp. 1–24, 2003.

[37] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich, "POPE: partial order preserving encoding," in *Proceedings of the of International Conference on Computer and Communications Security (CCS)*, pp. 1131–1142, Vienna, Austria, October 2016.

[38] J. Eom, D. H. Lee, and K. Lee, "Multi-client order-revealing encryption," *IEEE Access*, vol. 6, pp. 45458–45472, 2018.

[39] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proceedings of the International Conference on Computer and Communications Security (CCS)*, pp. 644–655, Denver, CO, USA, October 2015.

[40] Y. Pan, A. Efrat, M. Li et al., "Data inference from encrypted databases: a multi-dimensional order-preserving matching approach," 2020, https://arxiv.org/abs/2001.08773v1.

[41] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[42] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network*, vol. 15, no. 2, pp. 24–32, 2001.

[43] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast and scalable range query processing with strong privacy protection for cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2305–2318, 2016.