An Almost Non-interactive Order Preserving Encryption Scheme

Jingjing Guo[1,2], Jianfeng Wang[1,3], Zhiwei Zhang[1], and Xiaofeng Chen[1](✉)

[1] State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an 710071, People's Republic of China
jiaozuoguojing@163.com, {jfwang,zwzhang,xfchen}@xidian.edu.cn
[2] The State Key Laboratory of Cryptology, PO Box 5159, Beijing 100878, People's Republic of China
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, People's Republic of China

Abstract. Order preserving encryption (OPE) is an encryption scheme that the ciphertexts retain the order of their underlying plaintexts. It could be used to perform the order comparison or the efficient range query over encrypted data. Recently, plenty of work has been proposed on the construction of OPE scheme. Nevertheless, many existing OPE schemes require multiple rounds ($O(\log n)$) of interaction. As a result, real-time online, network delay and communication transmission failure are the efficiency challenges for the order comparison or range query. In this paper, we propose an almost non-interactive OPE scheme called BF-OPE. The BF-OPE scheme works by integrating Bloom filter and prefix encoding. They enable the encrypted data items to be compared when a token is provided by the client. Furthermore, the padding technique has been used to hide the frequency information both in data items and query ranges on the ciphertexts. Finally, we prove that the proposed scheme is secure with respect to the leakage function $\mathcal{L}_I$.

Keywords: Range query · Order preserving encryption
Order revealing encryption · Ideal security

1 Introduction

Range query, a common query operation, is generally used to select contiguous elements according to a label such as a timestamp and index. In particular, it draws much attention in "Big Data" for its power of boosting the system to perform some analysis over the stored data. To support high efficiency and low cost, these databases are always stored on remote untrusted servers, which drives the need to secure outsource the database. It is well-known that the traditional encryption schemes provide a way to achieve data confidentiality, however, it also destroys the order information and makes query difficult without decryption, notably for range query.

C. Su and H. Kikuchi (Eds.): ISPEC 2018, LNCS 11125, pp. 87–100, 2018.
https://doi.org/10.1007/978-3-319-99807-7_6

OPE is a simple and efficient encryption scheme where the order of plaintexts remained in the ciphertexts, namely, $\text{Enc}(x) > \text{Enc}(y)$ if and only if $x > y$. It enables the cloud server to perform comparison and range query directly over encrypted database [9,11,13,31,32], thus makes it very suitable in the outsourcing [8,10,26,30] in cloud computing. Therefore, the study on designing a secure and efficient OPE scheme is of both theoretical and practical significance.

Agrawal et al. [1] firstly proposed the definition of OPE while gave no security analysis. Boldyreva et al. [4] provided a rigorous treatment about the security and proved that ideal security is infeasible for OPE under certain implicit assumptions. As a result, they proposed an OPE scheme named as BCLO, and it achieves random order preserving function (ROPF) security, a weaker security definition. The ROPF security was later shown to leak at least half of the plaintext bits [5]. Popa et al. [24] proposed the first ideal security OPE scheme which is a mutable order preserving encryption (mOPE) scheme. The mOPE ensures that it will not reveal no more information except the order of the plaintexts. The main idea of mOPE is to build a balanced tree containing the plaintexts encrypted by the traditional encryption scheme. In addition, the mOPE scheme is interactive and requires multiple rounds of communication between the client and the cloud server. When updating or querying, it requires $O(\log n)$ rounds of communication and $O(1)$ client storage, where n is the number of items in database. The multiple rounds of interaction bring the new challenge for the client real-time online and network communication.

Recently, different from the traditional OPE schemes, Boneh et al. [6] firstly formalized the notion of order revealing encryption (ORE) scheme, which reveals the order of the corresponding plaintexts and nothing else. It is a generalization of OPE scheme. However, based on multilinear maps, the scheme in [6] is too impractical for most applications and remains a theoretical result. Lewi et al. [19] proposed an efficient ORE scheme, which is secure with the leakage function $\mathcal{L}_I$. Due to the size of the ciphertext is linear to the plaintext space, it is limited to small message spaces.

1.1 Our Contributions

In this paper, we further study the problem on the construction of order preserving encryption scheme. The contributions of this paper are as follows:

- We present a new non-interactive OPE scheme named BF-OPE by integrating Bloom filter and prefix encoding. BF-OPE can achieve efficient comparison over encrypted data without requiring additional interactions between the client and the server.
- We introduce the padding technique [25] to randomize the data items and query ranges, which can hide the frequency information on both data items and ranges. Security analysis shows that the proposed scheme satisfies the desired security goal.

1.2 Related Work

The problem of range query [7,21] is the most important query operation in "Big Data". The approach of range query can be divided into fully homomorphic encryption, searchable encryption [20,27,28] and order preserving encryption. OPE, as a practical approach for range query, has been studied in the past decades [1,4,14,18,24].

The concept of OPE was proposed by Agrawal et al.[1] in 2004. The basic idea is to take a target distribution which provided by a client and transform plaintext in such a way that the transformation preserves the order and follows target distribution. This scheme can only be used in a static system, and gave no security analysis. In [4], Boldyreva et al. introduced the ideal security definition and proposed the first provable OPE scheme, which we call BCLO scheme. They also proved that an OPE scheme is impossible to achieve ideal security unless its ciphertext space is extremely large exponential in the size of plaintext space. Therefore, the proposed BCLO scheme achieved a weaker security ROPF. Later, they showed that a ROPF scheme achieves security of window one-wayness. However, Boldyreva et al. [5] and Xiao et al. [29] proved that the BCLO scheme leaked at least half of plaintext bits. In [12], Dyer et al. designed an OPE scheme based on approximate integer common divisor problem. But scheme [12] only achieved window one-wayness security. There are other schemes [16,22,23,29] provided weaker security guarantee by making some assumptions about the attack, which donot hold in practice.

Popa et al. [24] proposed the first ideally-secure order preserving encoding scheme based on a data structure, where this scheme revealed no more information besides the lexicographical order. The main idea of their scheme is mutable ciphertext, which means order preserving encodings for several plaintexts change with updating a value. They proved that it is impossible for a linear length encryption scheme to achieve ideal security even if the encryption is stateful, namely, the mutable ciphertext is the precondition for ideal security. Kerschbaum [17] pointed out every (deterministic) OPE scheme suffer a simple attack from the frequency information. Therefore, he proposed a stronger security definition, indistinguishability under frequency analyzing ordered chosen plaintext attack (IND-FAOCPA), and settled for an encryption scheme with this stronger security. The main idea of this scheme is to randomize ciphertext to hide frequency information. Inspired by the buffer tree [2], Roche et al. [25] proposed a construction technique of order preserving tree and designed the partial order preserving encoding (POPE) scheme. The POPE scheme supports insert-heavy database and leaks less order information. However, all these schemes need multiple rounds of interaction.

1.3 Organization

The rest of this paper is organized as follows. The preliminaries are given in Sect. 2. The construction of BF-OPE scheme is given in Sect. 3. The security and efficiency analysis of the proposed scheme are given in Sect. 4. Finally, we give the conclusions in Sect. 5.

2 Preliminaries

In this section, we present some cryptographic tools and techniques used in our scheme, and the general definition of OPE scheme.

2.1 Bloom Filter

In [3], Bloom proposed the concept of Bloom filter (BF), which can be used to test whether an element belongs to a set. A Bloom filter contains a bit array of m bits and r independent hash functions defined as follows: $h_i : \{0,1\}^* \rightarrow [1,m]; i \in [1,r]$. In the initial phase, all the positions of the bit array are set to 0. If an element is added to the set, put it into the r hash functions to get r array positions, and turn these positions to 1. If the value of this position is 1, the operation does not work.

Given an element, the BF can test whether the element belongs to the set. Given an element x, the BF computes the r hash functions to get r array positions. If a value of these positions is 0, the element does not belong to the set. On the contrary, if all the positions are 1, we cannot decide whether this element belongs to the set or not. This is because of the existence of false positive.

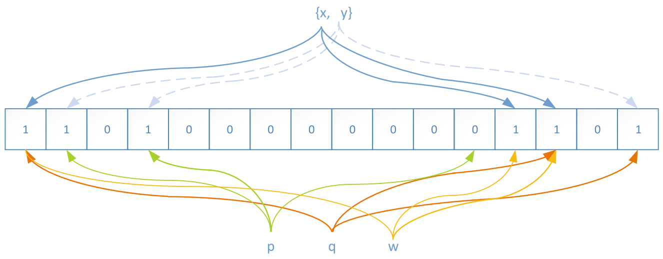

Fig. 1. A Bloom filter construction.

As shown in Fig. 1, it is easy to see that element p does not belong to the set and w belongs to. Although all the positions are 1, the element q does not belong to. That is called as false positive. The false positive satisfies $P_f = (1 - e^{-kn/m})^k$ and reaches its minimum value 2^{-r}, if $r = \ln 2 * (m/n)$, where n denotes the number of elements, r the number of hash functions, m the size of the Bloom filter.

2.2 Prefix Encoding

In [15], Gupta et al. utilized the prefix encoding to proceed range query over the plaintexts. The basic idea of prefix encoding is to convert the problem whether an element belongs to a range into the problem whether the intersection of two sets is empty.

Given an element x of w bits and its binary representation $x = b_1b_2\cdots b_w$, the prefix family of x is $F(x) = \{b_1b_2\cdots b_w, b_1\cdots b_{w-1}*, \cdots, b_1*\cdots*, **\cdots*\}$, which has the size of $w+1$. Given a range $[a,b]$, we denoted the minimum cover set of prefixes as $S([a,b])$. Its size is at most $2w-2$, where a and b are two elements of w bits. For example, given 3 of 5 bits, its prefix family is $F(3) = \{00011, 0001*, 000**, 00***, 0****, *****\}$ and the range prefix of $[0,6]$ is $S([0,6]) = \{000**, 0010*, 00110\}$. For a data item x and range $[a,b]$, x belongs to range $[a,b]$ if and only if there exists a prefix $P \in F(x)$ such that $P \in S([a,b])$, i.e., $F(x) \cap S([a,b]) \neq \phi$. From the above example, we can draw the conclusion that $3 \in [0,6]$ because $F(3) \cap S([0,6]) = \{000**\}$.

2.3 Order Preserving Encryption

An order preserving encryption scheme is a tuple of polynomial-time algorithms OPE=(**KeyGen, BuildTree, Enc, Search, Update, Dec)** defined over a database $\mathcal{D}$ with the following properties:

- **KeyGen**$(1^\lambda) \rightarrow SK$: On input the security parameter λ, the **KeyGen** algorithm is run by the client to generate a secret key SK, which is secretly stored by the client.
- **BuildTree**$(\mathcal{D}) \rightarrow \Gamma$: On input the database $\mathcal{D}$, the **BuildTree** algorithm is run by the client to build an OPE tree Γ, which is used to store and index the data items.
- **Enc**$(SK, \Gamma) \rightarrow \Gamma^*$: On input the secret key SK and the OPE tree Γ, the **Enc** algorithm is run by the client to produce the encrypted tree Γ^*.
- **Search**$(SK, \Gamma^*, R) \rightarrow I^*$: On input the secret key SK, encrypted OPE tree Γ^* and query range R, the **Search** algorithm is run to output the encrypted results for the client.
- **Update**$(\Gamma^*, SK, a) \rightarrow \Gamma^*$: On input the secret key SK and the data item a, the **Update** algorithm inserts the new data item into the OPE tree.
- **Dec**$(SK, I^*) \rightarrow I$: On input the secret key SK and the encrypted results I^*, the **Dec** algorithm is run by the client to obtain the results I.

Remark 1. In this paper, we use Γ^* and I^* to denote the ciphertexts of OPE tree Γ and result I, respectively.

Following, we also introduce some necessary definitions in our scheme. As described in [19], the correctness and security definitions were proposed as follows.

Definition 1 *(Correctness). We say that an OPE scheme over a well-domain $\mathcal{D}$ is correct if for $SK \leftarrow$ **KeyGen**(1^λ) such that, for $\forall m \in \mathcal{D}$ and range R, if $m \in R$, $I^* =$ **Search**(SK, Γ^*, R) then $m \in$ **Dec**(SK, I^*).*

Definition 2 *(Security). We say that an OPE scheme is secure with leakage function $\mathcal{L}_I$ if for all adversaries $\mathcal{A}$, the scheme reveals no more information* *besides the leakage function $\mathcal{L}_I$. In particular, we define leakage function $\mathcal{L}_I$ as follows:*

$$\mathcal{L}_I(m_i, m_j) = \{position_{diff}(m_i, m_j)\},$$

where $position_{diff}(m_i, m_j)$ gives the position of the first bit where m_i and m_j differ.

3 Main Construction of BF-OPE Scheme

In this work, we consider an OPE scheme used in the range query system in the outsourcing computing model, as illustrated in Fig. 2. The system model contains two entities: the client and the cloud server. The client is an entity who wants to outsource his own database to the cloud server, and the cloud server is an entity who provides the storage service and stores encrypted database on behave of the clients. We can view the cloud server as "honest-but-curious". That means, the cloud server follows the proposed protocol and returns the answers honestly, but tries to learn information about the encrypted data.

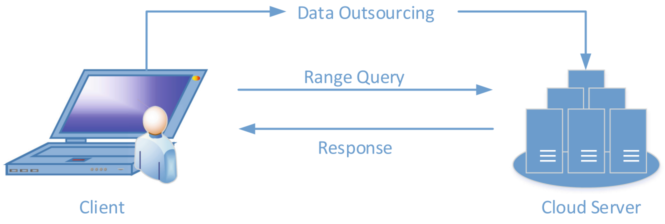

Fig. 2. Architecture of range query.

3.1 Main Idea

In this paper, we propose a new order preserving encryption scheme, BF-OPE scheme, for range query over encrypted database in cloud computing. The BF-OPE scheme can reduce the communication overhead between the client and the cloud server. Our main idea is that we insert an OPE index after the ciphertext of the data item. The OPE index can help the cloud server to decide the order information, therefore, the cloud server does not need to seek the order information by the interactions between him and the client.

We use the prefix encoding technique to change the problem on how to construct an OPE index to test whether the intersection of two sets is empty. Trivially, the problem can be solved by testing whether an element belongs to a set. In this condition, we use Bloom filter to test over the ciphertexts. However, the frequency information is revealed in the phase of **Update** and **Search**. This is because that the data item and range prefix is unique. To solve this problem, we use padding technique to hide the frequency information. Moreover, to protect the functionality of query, we take different padding techniques for the data item and query range. The detailed description is presented as follows.

3.2 The BF-OPE Scheme

A B-tree is a tree in which leaf node stores data items and internal node stores split points indicating the difference of subtrees. All data items in the left subtree of v are smaller than v and all data items in the right subtree are larger than v. The cloud server cannot obtain the order of two encrypted data in the same leaf node. In this paper, we build an OPE tree index which is inspired by the B-tree in the proposed scheme. For each node, we suppose that it has a storage limitation L, where L is the storage capacity of the client. Without loss of generality, we assume that all data items are w_1-bit and greater than 0. A detailed description of the proposed scheme is as follows.

- **KeyGen** (1^λ): Taking the security parameter λ as input, the client initializes the system parameters.
  1. By DET, we refer to an IND-CPA secure symmetric encryption scheme DET = (DET.Key, DET.Enc, DET.Dec). The client generates the secret key $sk_1 \leftarrow \text{DET.Key}(1^\lambda)$.
  2. A Bloom filter with r hash functions is initialized. At the same time, the client uses r secret keys $k_1, k_2, \cdots, k_r$ to compute r keyed hash functions.

  Therefore, the client has the secret key $SK = (sk_1, sk_2)$, where $sk_1 = \text{DET.Key}(1^\lambda), sk_2 = (k_1, k_2, \cdots, k_r)$.
- **BuildTree**$(\mathcal{D})$: Taking the database $\mathcal{D}$ as input, a client stores the data items in a B-tree in plaintext.
  1. The client firstly randomizes the data items d through padding a w_1-bit random number r as $d||01||r$. For simplicity, we always use d to represent its randomization. Therefore, d is a w-bit data, where $w = 2w_1 + 2$.
  2. Afterwards, the client computes the data prefix $F(d)$ and range prefix $S([0, d])$ (only for internal node), and then inserts the prefixes in the corresponding node. The client builds the OPE tree as Γ, where $\Gamma = \{(d_1, F(d_1)), \cdots, (d_n, F(d_n)), (t_1, F(t_1), S([0, t_1])), \cdots, (t_m, F(t_m), S([0, t_m]))\}$, n is the number of data items in the database and m denotes the number of split points in the OPE tree. A toy example of OPE tree Γ is shown in Fig. 3.
- **Enc** (SK, Γ): Taking secret key SK and OPE tree Γ as input, the client encrypts data items, data prefix and rang prefix to protect the privacy of data items.
  1. The client firstly encrypts data items as

$$C_{d_i} = \text{DET.Enc}(sk_1, d_i), \ (1 \le i \le n)$$
$$C_{t_j} = \text{DET.Enc}(sk_1, t_j), \ (1 \le j \le m).$$

  2. The client computes the Bloom filter for the set of data prefix and range prefix. For leaf node d, the client computes its Bloom filter of set $F(d)$ as BF_{1d}. For split point t, namely internal node, the client computes Bloom filters of $F(t)$ and $S([0, t])$ as BF_{1t} and BF_{2t}, respectively.

The client uploads encrypted OPE tree $\Gamma^* = \{(C_{d_1}, BF_{1d_1}), \cdots, (C_{d_n}, BF_{1d_n}), (C_{t_1}, BF_{1t_1}, BF_{2t_1}), \cdots, (C_{t_m}, BF_{1t_m}, BF_{2t_m})\}$ to the cloud server.

– **Search** $(SK, \Gamma^*, [a,b])$: Taking secret key $SK = (sk_1, (k_1, k_2, \cdots, k_r))$ and query range $[a,b]$ as input, the client generates the search token TK for the range $[a,b]$. Subsequently, the cloud server searches over the encrypted tree Γ^*, achieves the encrypted results I^*, and returns it to the client.
  1. For a query range $[a,b]$, the client firstly randomizes the range through padding w_1-bit random numbers r_1, r_2 as $a||0||r_1$ and $b||11||r_2$. For simplicity, we always use a and b to represent its randomization, respectively.
  2. The client computes range prefix as $S([a,b]) = \{P_1, P_2, \cdots, P_l\}$ and outputs its search token as a matrix to the cloud server.

$$M_{[a,b]} = \begin{pmatrix} H(k_1, P_1) & H(k_2, P_1) & \cdots & H(k_r, P_1) \\ \vdots & \vdots & \ddots & \vdots \\ H(k_1, P_l) & H(k_2, P_l) & \cdots & H(k_r, P_l) \end{pmatrix}$$

  3. After receiving the matrix $M_{[a,b]}$, the cloud server searches OPE tree from the root node to the leaf node. The cloud server could find the leftmost and rightmost leaf nodes that intersect with range. Subsequently, the cloud server outputs the ciphertexts in the leaf node between these two leaf nodes and tests the data items in these two leaf nodes whether belong to the range, if belongs to, outputs it. More concretely, the cloud server tests whether $S([a,b])$ intersects with data index $F(t)$, where t is a split point. If intersects, there exists a prefix $P_i \in S([a,b])$ such that $P_i \in F(t)$, namely, there exists a row i in matrix $M_{[a,b]}$ such that $BF_1(P_i) = 1$. The equation $BF_1(P_i) = 1$ means the values at the position $H(k_1, P_i), H(k_2, P_i), \cdots, H(k_r, P_i)$ of Bloom filter BF_1 are all equal to 1. After finding the leftmost and rightmost leaf nodes, the cloud server returns data in the leaf node between these two leaf nodes. The data items in these two nodes can be decided as above.

– **Update** (Γ^*, a): Taking encrypted OPE tree Γ^* and the new data item a, the client generates the ciphertext C_a and the token TK_a, and submits them to the cloud server. Afterwards, the cloud server uses the ciphertext C_a and the token TK_a to update the encrypted OPE tree Γ^*.
  1. The client computes ciphertext C_a, data index BF_1, and matrix $M_{F(a)}$, where

$$M_{F(a)} = \begin{pmatrix} H(k_1, P_1) & H(k_2, P_1) & \cdots & H(k_r, P_1) \\ \vdots & \vdots & \ddots & \vdots \\ H(k_1, P_{w+1}) & H(k_2, P_{w+1}) & \cdots & H(k_r, P_{w+1}) \end{pmatrix}$$

     Then the client submits them to the cloud server.
  2. After receiving $(C_a, BF_{1a}, M_{F(a)})$, the cloud server searches the leaf node to insert new data through texting whether there exists a row i satisfies $BF_1(P_i) = 1$. Finally, the cloud server inserts data a into its corresponding leaf node. The test algorithm is similar to the range query.

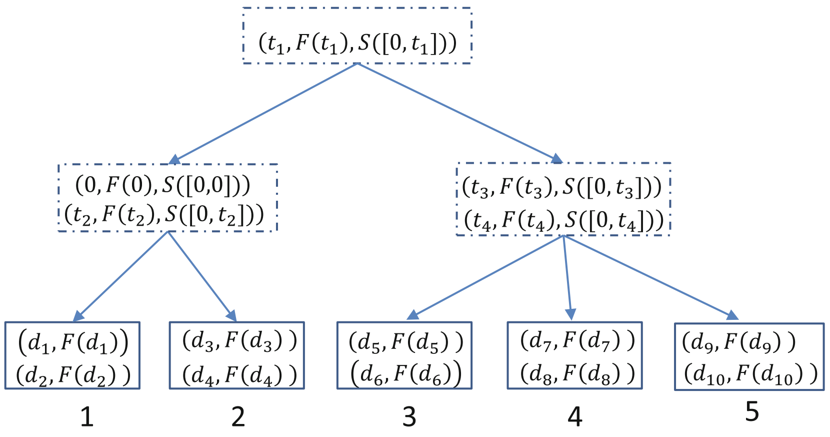

Fig. 3. The OPE tree construction.

When arriving at its storage limitation L, this leaf node splits into two leaf nodes and inserts a new split point into its parent node. The cloud server first returns left and right split point of this leaf node to the client, like C_{92}, C_{118}. If it is the rightmost leaf node, the cloud server only returns its left split point. After receiving these two split points, he decrypts them, selects a value a between these two split points, computes $(C_a, BF_1, BF_2, M_{[0,a]})$, and uploads it to cloud server. The cloud server splits the leaf node into two leaf nodes and inserts (C_a, BF_1, BF_2) as a new split point, as shown in Fig. 4. If the parent contains too many split points, the split propagates upward. Tree splitting is the only step which needs intersection in our scheme. The deletion operation is a little different from the insertion operation. In deletion, the client computes $M_{F(a)}$ and submits it to the cloud server. The cloud server finds the data items precisely. In the leaf node, the cloud server decides whether two elements are the same one. Namely, there is no row i in matrix $M_{F(a)}$ such that $BF_1(P_i) = 0$ (the Bloom filter of data in leaf node). In this paper, we divide modification into deletion and insertion. Thus, the proposed scheme can be used in a dynamic database. If we donot take data update into consideration, the range index BF_2 can be removed from split point.

– **Dec(SK, I^*):** Taking the secret key SK and encrypted results I^* as input, the client decrypts the encrypted results and obtains results I. Furthermore, the client removes the padding numbers and obtains the data item.

4 Analysis

In this section, we present the security and efficiency analysis of the proposed BF-OPE scheme.

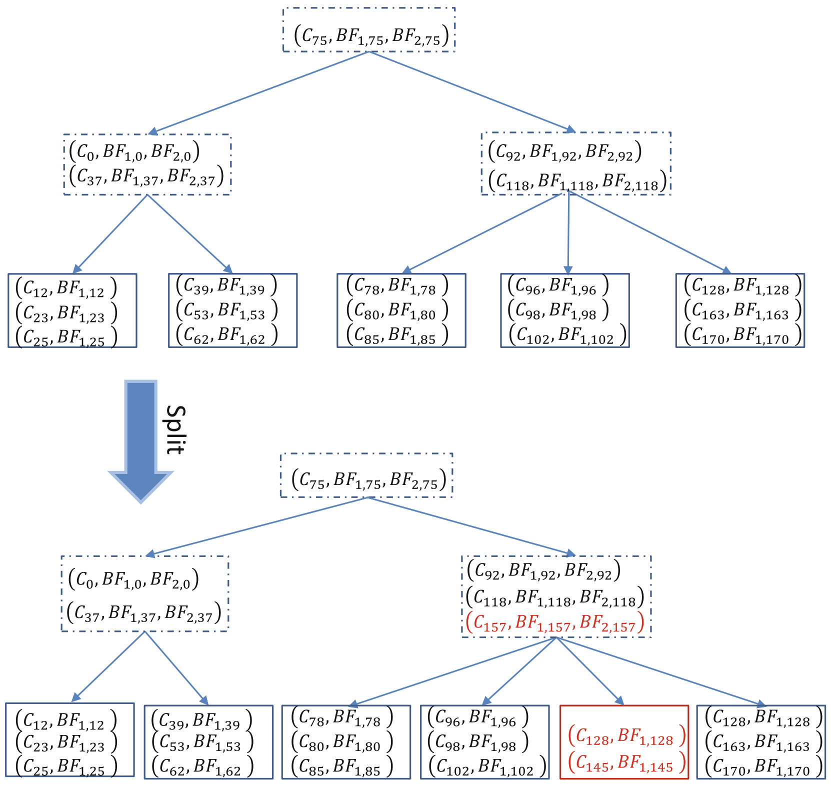
$(C_{75}, BF_{1,75}, BF_{2,75})$
$(C_0, BF_{1,0}, BF_{2,0})$
$(C_{37}, BF_{1,37}, BF_{2,37})$
$(C_{92}, BF_{1,92}, BF_{2,92})$
$(C_{118}, BF_{1,118}, BF_{2,118})$
$(C_{12}, BF_{1,12})$ $(C_{23}, BF_{1,23})$ $(C_{25}, BF_{1,25})$
$(C_{39}, BF_{1,39})$ $(C_{53}, BF_{1,53})$ $(C_{62}, BF_{1,62})$
$(C_{78}, BF_{1,78})$ $(C_{80}, BF_{1,80})$ $(C_{85}, BF_{1,85})$
$(C_{96}, BF_{1,96})$ $(C_{98}, BF_{1,98})$ $(C_{102}, BF_{1,102})$
$(C_{128}, BF_{1,128})$ $(C_{163}, BF_{1,163})$ $(C_{170}, BF_{1,170})$
Split
$(C_{157}, BF_{1,157}, BF_{2,157})$
$(C_{128}, BF_{1,128})$ $(C_{145}, BF_{1,145})$

Fig. 4. A toy example of OPE tree.

4.1 Security Analysis

Theorem 1. *The proposed BF-OPE scheme over a well-domain $\mathcal{D}$ is correct.*

Proof. We suppose that data item $d \in \mathcal{D}$, range $\in [a, b]$ and their randomization $d||01||r, a||00||r_1, b||11||r_2$, where r, r_1, r_2 are three w_1-bit random numbers. If $d \in [a, b]$, then we have $d||01||r \in [a||00||r_1, b||11||r_2]$ holds.

If $d||01||r \in [a||00||r_1, b||11||r_2]$ holds, then $F(d||01||r) \cap S([a||00||r_1, b||11||r_2]) \neq \phi$. In this condition, data $d||01||r$ will be returned as the search result, namely, $C_{d||01||r} \in I^*$, where $I^* = \textbf{Search}(SK, \Gamma^*, [a, b])$. After decryption, we have $d \in \textbf{Dec}(SK, I^*)$.

Theorem 2. *The proposed BF-OPE scheme is secure with respect to leakage function $\mathcal{L}_I$.*

Proof. The ciphertexts were composed of C_a and index. The ciphertext C_a was produced through a plaintext-indistinguishable encryption scheme. In this case, the ciphertexts have the property that they are semantically-secure encryptions.

Furthermore, we consider the security of index, which has not been solved by padding technique. If $|F(a) \cap F(b)| = m$, then the first bit, where m_i and m_j differ, occurs at $m-$th, namely, $a_1 = b_1, \cdots, a_{m-1} = b_{m-1}$, and $a_m \neq b_m$. Since padding technique runs through padding randomness number r behind data d, it cannot solve the above problem. The basic reason is that an adversary has the ability to decide whether $P_1 = P_2$, where $P_1 \in F(a)$ and $P_2 \in F(b)$. In insertion phase, the client provides the matrix $M_{F(a)}$ and $M_{F(b)}$. In this condition, the cloud server decides the identical row in matrix $M_{F(a)}$ and $M_{F(b)}$. Therefore, the cloud server knows the leakage function $\mathcal{L}_I(m_i, m_j) = \{position_{diff}(m_i, m_j)\}$.

4.2 Efficiency Analysis

For the convenience of discussion, some marks are introduced. We denote by E an encryption, D a decryption, H an operation of Hash, r the number of Hash in Bloom filter, n the size of a database, and w the bits of our data item. We omit other operations such as comparison of plaintexts.

Table 1. Computation cost of our scheme

Schemes	Scheme [19]	Scheme [24]	BF-OPE
Security	IND-rOCPA	IND-OCPA	$\mathcal{L}_I(m_i, m_j)$
Interaction	0	$\log n$	0
Client (Insert)	$E + 2^w \cdot (D + H)$	$E + \log n \cdot D$	$E + (w+1) \cdot rH$
Client (Delete)	E	$\log n \cdot D$	$(w+1) \cdot rH$
Client (Search)	$2E$	$2 \log n \cdot D$	$(2w-2) \cdot rH$
Cloud (Insert)	$\log n \cdot H$	0	0
Cloud (Delete)	$\log n \cdot H$	0	0
Cloud (Search)	$2 \log n \cdot H$	0	0

Table 1 presents the comparison among scheme [19], scheme [24] and BF-OPE scheme. It can be seen that scheme [24] achieves IND-OCPA security, which is the first scheme who achieved the ideal security. In scheme [19], the right components achieve IND-OCPA security denoted as IND-rOCPA security. The proposed BF-OPE scheme leaks the bit where the difference happens.

When inserting, deleting and querying, scheme [24] needs $\log n$ rounds of interaction. The client decrypts ciphertexts to help cloud server to decide the order of two ciphertexts in update and query phase. In scheme [24], the cloud server does nothing computation in update and query phase. In BF-OPE scheme, the cloud server checks whether the position of Bloom filter is equal to 1. Owing to the low computation overhead, both the computation of cloud server are denoted as 0.

5 Conclusions

In this paper, we propose an almost non-interactive order preserving encryption scheme for range query, which is a basic search operation in the outsourced database. Note that the state-of-the-art order preserving encryption scheme called mOPE needs multiple rounds of interaction between the clients and the cloud server. Therefore, the mOPE is easily influenced by the network failures and increases the communication burden. Based on these reasons, we designed a BF-OPE scheme, which is secure with respect to the leakage function $\mathcal{L}_I$ and can hide the frequency information of outsourced data. Furthermore, the proposed scheme leaks partial order information among the ciphertexts.

Acknowledgements. This work was supported by the National Cryptography Development Fund (No. MMJJ20180110).

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-preserving encryption for numeric data. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France, pp. 563–574 (2004)
2. Arge, L.: The buffer tree: a technique for designing batched external data structures. Algorithmica **37**(1), 1–24 (2003)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
4. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Sofia, Bulgaria, pp. 563–594 (2015)
5. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_33
6. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_19
7. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
8. Chen, X., Huang, X., Li, J., Ma, J., Lou, W., Wong, D.S.: New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Trans. Inf. Forensics Secur. **10**(1), 69–78 (2015)
9. Chen, X., Li, J., Huang, X., Ma, J., Lou, W.: New publicly verifiable databases with efficient updates. IEEE Trans. Dependable Secure Comput. **12**(5), 546–556 (2015)
10. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. IEEE Trans. Parallel Distrib. Syst. **25**(9), 2386–2396 (2014)

11. Chen, X., Li, J., Weng, J., Ma, J., Lou, W.: Verifiable computation over large database with incremental updates. IEEE Trans. Comput. **65**(10), 3184–3195 (2016)
12. Dyer, J., Dyer, M., Xu, J.: Order-preserving encryption using approximate integer common divisors. In: Garcia-Alfaro, J., Navarro-Arribas, G., Hartenstein, H., Herrera-Joancomartí, J. (eds.) ESORICS/DPM/CBT -2017. LNCS, vol. 10436, pp. 257–274. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67816-0_15
13. B. Fuller, et al.: Sok: cryptographically protected database search. In: Proceedings of the IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, pp. 172–191 (2017)
14. Furukawa, J.: Request-based comparable encryption. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 129–146. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40203-6_8
15. Gupta, P., McKeown, N.: Algorithms for packet classification. IEEE Netw. **15**(2), 24–32 (2001)
16. Kadhem, H., Amagasa, T., Kitagawa, H.: A secure and efficient order preserving encryption scheme for relational databases. In: Proceedings of the International Conference on Knowledge Management and Information Sharing (KMIS), Valencia, Spain, pp. 25–35 (2010)
17. Kerschbaum, F.: Frequency-hiding order-preserving encryption. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), Denver, CO, USA, pp. 656–667 (2015)
18. Lee, S., Park, T., Lee, D., Nam, T., Kim, S.: Chaotic order preserving encryption for efficient and secure queries on databases. IEICE Trans. Inf. Syst. **92**(11), 2207–2217 (2009)
19. Lewi, K., Wu, D.J.: Order-revealing encryption: new constructions, applications, and lower bounds. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), Vienna, Austria, pp. 1167–1178 (2016)
20. Li, J., Chen, X., Xhafa, F., Barolli, L.: Secure deduplication storage systems supporting keyword search. J. Comput. Syst. Sci. **81**(8), 1532–1541 (2015)
21. Li, Y., Lai, J., Wang, C., Zhang, J., Xiong, J.: Verifiable range query processing for cloud computing. In: Liu, J.K., Samarati, P. (eds.) ISPEC 2017. LNCS, vol. 10701, pp. 333–349. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-72359-4_19
22. Liu, D., Wang, S.: Programmable order-preserving secure index for encrypted database query. In: Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, pp. 502–509 (2012)
23. Liu, D., Wang, S.: Nonlinear order preserving index for encrypted database query in service cloud environments. Concurr. Comput.: Pract. Exp. **25**(13), 1967–1984 (2013)
24. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: Proceedings of the IEEE Symposium on Security and Privacy (SP), Berkeley, CA, USA, pp. 463–477 (2013)
25. Roche, D.S., Apon, D., Choi, S.G., Yerukhimovich, A.: POPE: partial order preserving encoding. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), Vienna, Austria, pp. 1131–1142 (2016)
26. Wang, J., Chen, X., Huang, X., You, I., Xiang, Y.: Verifiable auditing for outsourced database in cloud computing. IEEE Trans. Comput. **64**(11), 3293–3303 (2015)

27. Wang, J., Chen, X., Li, J., Zhao, J., Shen, J.: Towards achieving flexible and verifiable search for outsourced database in cloud computing. Future Gener. Comput. Syst. **67**, 266–275 (2017)
28. Wang, Y., Wang, J., Chen, X.: Secure searchable encryption: a survey. J. Commun. Inf. Netw. **1**(4), 52–65 (2016)
29. Xiao, L., Yen, I.: Security analysis for order preserving encryption schemes. In: Proceedings of the Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, pp. 1–6 (2012)
30. Zhang, X., Jiang, T., Li, K.-C., Castiglione, A., Chen, X.: New publicly verifiable computation for batch matrix multiplication. Inf. Sci. (2017). https://doi.org/10.1016/j.ins.2017.11.063
31. Zhang, Z., Chen, X., Li, J., Tao, X., Ma, J.: HVDB: a hierarchical verifiable database scheme with scalable updates. J. Ambient Intell. Humaniz. Comput. (2018). https://doi.org/10.1007/s12652-018-0757-8
32. Zhang, Z., Chen, X., Ma, J., Shen, J.: SLDS: secure and location-sensitive data sharing scheme for cloud-assisted cyber-physical systems. Future Gener. Comput. Syst. (2018). https://doi.org/10.1016/j.future.2018.01.025