MPOPE: Multi-provider Order-Preserving Encryption for Cloud Data Privacy

Jinwen Liang[1], Zheng Qin[1(✉)], Sheng Xiao[1], Jixin Zhang[1], Hui Yin[1], and Keqin Li[1,2]

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, China
{jimmieleung,zqin,xiaosheng,zhangjixin}@hnu.edu.cn, yhui@ccsu.edu.cn
[2] Department of Computer Science, State University of New York, New Paltz, NY, USA
lik@newpaltz.edu

Abstract. Order-preserving encryption (OPE) has been proposed as a privacy-preserving query method for cloud computing. Existing researches of OPE diverge into two groups. One group focuses on single data provider scenarios and achieves strong security notion such as indistinguishability under ordered chosen plaintext attack (IND-OCPA). Another group of research designs multi-provider schemes and provides weaker security guarantees than those of single provider schemes. In this paper, we propose a novel security notion for multi-provider scenario, indistinguishability under multi-provider ordered chosen plaintext attack (IND-MPOPCA), which guarantees equivalent security level as IND-OCPA while hiding the frequency of plaintexts and enabling multi-provider data submissions and queries. We develop a multi-provider randomized order technique to construct our MPOPE scheme to achieve the IND-MPOPCA security notion. We also conduct extensive experiments to prove the practicality and efficiency of our proposed scheme.

Keywords: Order-preserving encryption · Multiple data provider
Cloud security

1 Introduction

The flexibility of storing data on a cloud and making queries anywhere in the Internet is attractive. While the risk of data privacy breach severely weakens the desire of uploading data to the cloud [1]. With such a contention, a common solution is to encrypt data before uploading to the cloud. However, it becomes complicated to query the encrypted data, and even more difficult to hide the queries from being understood by the semi-trusted cloud.

Various methods had been proposed for privacy-preserving cloud queries, such as keyword query, fuzzy query, range query, etc. [2]. Among these categories of privacy-preserving query methods, range query gains the most research efforts because it is arguably the most promising direction to provide practically efficient and accurate solution for the privacy-preserving query problem [3–6].

X. Lin et al. (Eds.): SecureComm 2017, LNICST 238, pp. 808–822, 2018.
https://doi.org/10.1007/978-3-319-78813-5_44

Order-preserving encryption (OPE) is the main technique used in range query schemes. The plaintext and ciphertext are kept in the same order under some value-mapping function [7,8]. Although a significant amount of work on OPE has been proposed, most of these works focus on the single data provider scenario, such as [3–6,9]. Since collecting and storing a large amount of data provided by multiple data providers is a common work-flow for many cloud storage applications, these single data provider schemes are not widely applicable. Single data provider scheme are more of theoretic attempts to push the security notions to the limit, such as indistinguishability under ordered chosen plaintext attack (IND-OCPA).

On the other side, multiple data provider schemes (or abbreviated as multi-provider schemes or multi-user schemes), such as [10,11], focus on the practicality and achieve weaker security notions than IND-OCPA, which had been implemented in many single-provider order-preserving encryption schemes, such as [3–5]. Also, a common foe to the multi-provider schemes is frequency analysis attack. As a comparison, the security feature of frequency hiding had been implemented in Kerschbaum's single-provider scheme [5] but not in any of existing multi-provider scheme.

Therefore, it is desirable to design a multi-provider scheme that achieves security notion as strong as IND-OCPA in the single-provider schemes and ensure such a scheme also stands against frequency analysis attacks.

In this paper, we propose *multi-provider randomized order* technique for increasing the security of multi-provider order-preserving encryption. We propose a new security notion for multi-provider order-preserving encryption. We also develop a novel multi-provider order-preserving encryption scheme under this security notion.

We summarize our contributions as follows.

- We propose a stronger security notion for multi-provider order-preserving encryption than IND-OCPA: *indistinguishability under multi-provider ordered chosen plaintext attack* (IND-MPOCPA).
- We develop a novel multi-provider order-preserving encryption scheme under IND-MPOCPA by implementing the *multi-provider randomized order*.
- We provide theoretical analyses and experimental evaluation for our scheme.

2 Definitions

2.1 Definitions for Our Scheme

We provide Table 1 to summarize notations and their definitions for our scheme. Our (stateful) multi-provider order-preserving encryption (MPOPE) can be defined below:

- MPOPE.$KeyGen(N) \rightarrow T$: initialize the secret state T.
- MPOPE.$Enc(T, DETcipher_k, DP_k, n_k) \rightarrow T', C$: Compute an OPE ciphertext set C after encrypted n_k DETcipher, and update the state T to T'.
- MPOPE.$Dec(T, c_i) \rightarrow DETcipher$: Find the corresponding DETcipher for the OPE ciphertext c_i based on state T.

Table 1. Summary of notations and definitions

Notation	Definition
S	The cloud server
K	The number of data providers
DP_k	The $k\,th$ data provider, k = 1, 2, ..., K
n_k	The number of plaintexts provided by data provider DP_k
P_k	The plaintext set with n_k values provided by DP_k
$p_{k,i}$	A plaintext provided by data provider DP_k, i = 1, 2, ..., n_k
D	The plaintext domain, namely, $\forall p_{k,i} \in [1, D]$
DET	A deterministic encryption scheme, which satisfies DET = (DET.$KeyGen$, DET.Enc, DET.Dec)
sk_k	The DET symmetric key generated by DP_k
$DETcipher_k$	The corresponding DET ciphertext set of P_k encrypted by DP_k, i.e., $DETcipher_k = \{DETcipher_{k,1}, DETcipher_{k,2}, \ldots, DETcipher_{k,n_k}\}$
$DETcipher_{k,i}$	The corresponding DET ciphertext of $p_{k,i}$
HOM	A homomorphic encryption scheme, which satisfies HOM = (HOM.$KeyGen$, HOM.Enc, HOM.Dec)
PK	The public key of HOM published to each data provider
SK	The secret key of HOM generated by S
MPOPE	Our (stateful) multi-provider order-preserving encryption
T	The secret state of MPOPE
N	The number of distinct ciphertexts
C	The OPE ciphertext set with N values
$c_{k,i}$	An OPE cipher provided by DP_k, i = 1, 2, ..., N
M	The ciphertext domain of our order-preserving encryption scheme, namely, $\forall c_{k,i} \in [0, M]$

2.2 Model

System Model. Our system model involves multiple data providers (multi-provider) and a semi-trusted cloud. As is shown in Fig. 1, multiple data providers outsource their data to the cloud server in the encrypted form, which still enables comparison operation.

Threat Model. In our threat model, an honest-but-curious adversary will follow our protocol honestly but try to analyze and extract information about data. Both the cloud server and the data providers are considered as honest-but-curious adversary.

We consider about three types of attacks:

1. Type 1: Ordered Chosen Plaintext Attack. The cloud server try to extract relation between plaintexts and ciphertexts by asking the challenger to encrypt plaintext sequences [12].
2. Type 2: Frequency analysis. The cloud server try to confirm some plaintexts by observing the distribution of ciphertexts [5].

3. Type 3: Analysis between data providers. A data provider try to detect whether other providers encrypted the same data by observing the ciphertexts.

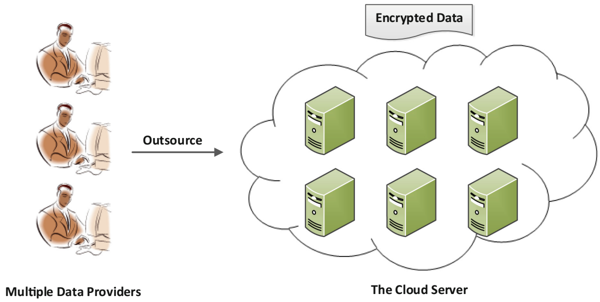


Fig. 1. Our system model

2.3 Security Definition

In order to resist those three types of attacks in our threat model, we propose a novel security notion for multi-provider order-preserving encryption: *indistinguishability under multi-provider ordered chosen plaintext attack* (IND-MPOCPA). Previous IND-OCPA security notion for order-preserving encryption is secure against Type 1 attack [8]. However, it has not considered about both Type 2 and Type 3 attacks. We define a multi-provider randomized order to enhance the ideal-security notion and resist both two attacks.

Definition 1 *(Multi-provider randomized order)***.** Let the plaintexts provided by different data providers are integrated into a sequence $W = \{w_{*,1}, w_{*,2}, \dots, w_{*,n}\}$ with n not necessarily distinct plaintexts, where $*$ denotes any data provider. A multi-provider randomized order $\Pi = \{\pi_{*,1}, \pi_{*,2}, \dots, \pi_{*,n}\}$ of W which satisfies that $\forall i \in [1, n], \pi_{*,i} \in [1, n]$ and $\forall i, j \in [1, n], i \neq j \Rightarrow \pi_{*,i} \neq \pi_{*,j}$, holds that

$$\forall i, j \,.\, w_{*,i} < w_{*,j} \Rightarrow \pi_{*,i} < \pi_{*,j}$$

and

$$\forall i, j \,.\, \pi_{*,i} < \pi_{*,j} \Rightarrow w_{*,i} \leq w_{*,j}$$

Our multi-provider randomized order is a permutation of the order of not necessarily distinct plaintexts uploaded by different data providers. Namely, the multi-provider randomized order not only preserve the order of distinct plaintexts but also randomize the order of identical plaintexts provided by different data providers. Therefore, the multi-provider randomized order can perfectly resist Type 2 and Type 3 attack.

Our IND-MPOCPA security game involves an adversary, a challenger, and K data providers. The adversary generates two n value sequences $W^0 = \{w^0_{*,1}, w^0_{*,2}, \dots, w^0_{*,n}\}$ and $W^1 = \{w^1_{*,1}, w^1_{*,2}, \dots, w^1_{*,n}\}$, which have the same order relation (namely, $\forall i, j \in [1, n], w^0_{*,i} < w^0_{*,j} \Leftrightarrow w^1_{*,i} < w^1_{*,j}$). Therefore, those two sequences have at least one common multi-provider randomized order.

IND-MPOCPA Security Game.

(1) The adversary sends W^0 and W^1 to the challenger.
(2) The challenger chooses a random bit $b \in \{0, 1\}$.
(3) The challenger and the set of providers engage in n rounds. At round i:
  (a) The challenger sends $w^b_{k,i}$ to DP_k, where k denotes any provider who provides the i-th plaintext and is defined by the adversary.
  (b) DP_k returns $c_{k,i} = \text{MPOPE}.Enc(w^b_{k,i})$ to the challenger.
(4) The challenger returns the corresponding OPE ciphertext sequence $C = \{c_{*,1}, c_{*,2}, \ldots, c_{*,n}\}$ to the adversary, where $*$ denotes any provider from the provider set $DP_1, DP_2, \ldots, DP_k, \ldots, DP_K$.
(5) The adversary outputs b', its guess for b.

We say that the adversary wins the game if his guess is correct, i.e., $b' = b$. Let $win_{\mathcal{A}}$ be the random probability that indicates the success of the adversary wins the above game. We define the indistinguishability under a multi-provider ordered chosen plaintext attack (IND-MPOCPA) notion below:

Definition 2 *(IND-MPOCPA: indistinguishability under multi-provider ordered chosen plaintext attack)***.** A multi-provider order-preserving encryption scheme is IND-MPOCPA secure if for all p.p.t. adversaries, $Pr[win_{\mathcal{A}}] \leq \frac{1}{2}$.

Since the multi-provider randomized order only leaks the order of data and permutates the order of identical plaintexts provided by different data providers randomly, our IND-MPOCPA is secure against Type 1, Type 2, and Type 3 attack. Since the IND-OCPA security notion can only resist Type 1 attack, IND-MPOCPA security is strictly stronger than IND-OCPA security. Therefore, our IND-MPOCPA security notion is an enhancement of IND-OCPA security notion for multi-provider order-preserving encryption.

3 Our Scheme

We propose a secret state, which implements the multi-provider randomized order technique, to achieve this goal. Later, we construct a novel multi-provider order-preserving encryption scheme based on the secret state.

Our comparing protocol is the key technique to implement the multi-provider randomized order technique. The goal of our comparing protocol is: (1) to compare values from multiple data providers secretly, (2) to randomize the comparison result of two identical plaintexts provided by different data providers, and (3) to achieve IND-CPA security notion.

Our comparing protocol is a secure three-party computation protocol. We utilize Paillier cryptosystem [13] to construct it. We use E() and D() to denote HOM.Enc() and HOM.Dec() respectively. We provide our comparing protocol in Algorithm 1.

In Algorithm 1, DP_i uses b_i to randomize the compare result R. We show the relation between b_i and R in Table 2. Since DP_i chooses b_i randomly, the compare result R of two identical data is randomized. In our three-party comparing protocol, DP_i uses r_i and r'_i to randomized the ciphertext of $(-1)^{b_i} \cdot (p_{i,x} - p_{j,y})$, DP_j uses b_j, r_j, r'_j to re-randomize the result. Therefore, S cannot recover $(-1)^{b_i} \cdot (p_{i,x} - p_{j,y})$ by decrypting $v_{2,3}$.

Algorithm 1. Comparing Protocol

Input: DP_i, DP_j, S, $DETcipher_{i,x}$, $DETcipher_{j,y}$.

Output: A compare result R

initialization: The cloud server runs HOM.$KeyGen()$. Data provider DP_i and DP_j decrypt $DETcipher_{i,x}$ and $DETcipher_{j,y}$ and obtain the corresponding plaintexts $p_{i,x}$ and $p_{j,y}$ respectively.

1: DP_i computes $\mathrm{E}(p_{i,x})$.

2: DP_j computes $\mathrm{E}(-p_{j,y})$, and sends it to DP_i.

3: DP_i computes a vector $V = (v_{1,1}, v_{1,2}, v_{1,3})$ and sends it to DP_j. Firstly, he flips a random coin $b_i \in \{0, 1\}$. Secondly, he randomly chooses two large random numbers r_i and r'_i, which satisfy $r_i > r'_i$. Then he calculates:

$$\begin{aligned} v_{1,1} &= \mathrm{E}(1) \\ v_{1,2} &= \mathrm{E}(0) \\ v_{1,3} &= (\mathrm{E}(p_{i,x}) \cdot \mathrm{E}(-p_{j,y}))^{(-1)^{b_i} \cdot r_i} \cdot \mathrm{E}(-r'_i) \\ &= \mathrm{E}(r_i \cdot (-1)^{b_i} \cdot (p_{i,x} - p_{j,y}) - r'_i) \end{aligned}$$

Finally, he sends V to DP_j.

4: DP_j re-randomized the vector $V = (v_{2,1}, v_{2,2}, v_{2,3})$ and sends it to S. Firstly, he flips a random coin $b_j \in \{0, 1\}$. Secondly, he randomly selects two large numbers r_j and r'_j which satisfy $r_j > r'_j$. Then he calculates:

$$\begin{aligned} v_{2,1} &= v_{1,1+b_j} \cdot \mathrm{E}(0) \\ v_{2,2} &= v_{1,2-b_j} \cdot \mathrm{E}(0) \\ v_{2,3} &= v_{1,3}^{(-1)^{b_j} \cdot r_j} \cdot \mathrm{E}((-1)^{1+b_j} \cdot r'_j) \\ &= \mathrm{E}((-1)^{b_j} \cdot (r_j \cdot v_{1,3} - r'_j)) \end{aligned}$$

Finally, he sends V to S.

5: S decrypts the vector V. If $\mathrm{D}(v_{2,3}) < 0$, then the cloud server sends $\mathrm{D}(v_{2,1})$ to DP_i. Else, the cloud server sends $\mathrm{D}(v_{2,2})$ to DP_i.

6: DP_i calculates $R = \mathrm{D}(v_{2,k})$ xor b_i, where $k = 1$ or 2.

We proceed as our secret state construction. Our secret state refers to an AVL tree T with a set of nodes $\{t\}$, which should be shared to the cloud server and multiple data providers. We show and explain the data structure of our AVL tree in Table 3. Then we provide a protocol to initialize and refresh the state of our scheme in Algorithm 2.

Table 2. A description of Algorithm 1

Case					
$p_{i,x} < p_{j,y}$		$p_{i,x} = p_{j,y}$		$p_{i,x} > p_{j,y}$	
b_i	R	b_i	R	b_i	R
0	**1**	0	**1**	0	**0**
1	**1**	1	**0**	1	**0**

Table 3. Parameters and explanation for tree node structure

Parameters	Explanations
Int *providerid*	A data provider, for example, DP_k
ElementType $DETcipher$	A DET ciphertext encrypted by *providerid*
ElementType $OPEcipher$	The OPE ciphertexts
AVLNode $*left$	A pointer point to the left child
AVLNode $*right$	A pointer point to the right child

Algorithm 2. Refreshing the secret state REFRESH

Input: An AVL tree T with nodes $\{t\}$, DP_k, $DETcipher_k$, S.
Output: An AVL tree T' with nodes $\{t\} \bigcup \{DETcipher_k\}$.
Initialization: Create an empty AVL tree.

1: **for** $i = 0$ to n_k **do**
2:   **if** $DETcipher_{k,i}$ was not in the set $\{t\}$. **then**
3:     DP_k asks the server for the root node of the AVL tree.
4:     S returns a node r to DP_k.
5:     **if** The node r was provided by DP_k. **then**
6:       DP_k decrypts both $r.DETcipher$ and $DETcipher_{k,i}$, and compare the corresponding plaintexts p_r with $p_{k,i}$.
7:     **else if** The node t was not provided by DP_k. **then**
8:       DP_k invokes the comparing protocol (Algorithm 1) to compare $p_{k,i}$ with p_r secretly.
9:     **end if**
10:     If $p_{k,i} < p_r$, DP_k asks S for the left child node; If $p_{k,i} > p_r$, DP_k asks S for the right child node.
11:     **if** S does not arrive at an empty spot in the AVL tree. **then**
12:       S returns the next node based on DP_k's information, and goes back to step 4.
13:     **end if**
14:     S inserts the new node into the AVL tree and balances the AVL tree.
15:   **end if**
16: **end for**
17: The algorithm outputs a new AVL tree T'.

In Algorithm 2, we initialize and refresh the secret state by constructing an AVL tree. Each node in our AVL tree is arranged based on the order of the plaintext value. The AVL tree is constructed and stored on the cloud server. Multiple data providers help the cloud server to find the location for his plaintexts in the tree as well as to construct the AVL tree by using the DET ciphertexts.

We provide Algorithm 3 to produce OPE ciphertexts by utilizing the secret state. We initialize the lower and the upper bounders Min and Max in Algorithm 3 to be -1 and M respectively. Each node's $OPEcipher$ is the mean value of Min and Max, and is generated by recursion. Note that the update algorithm is run on the cloud server S. We provide our multi-provider order-preserving encryption scheme in Algorithms 4 and 5.

Algorithm 3. Update UPDATE

Input: S, AVLNode $*t$, Min, Max.
State: The AVLTree T of nodes $\{t\}$.

1: **if** $T \neq NULL$ **then**
2: $\quad t.OPEcipher = \lceil \frac{Max+Min}{2} \rceil$
3: $\quad Update(t.left, Min, t.OPEcipher)$
4: $\quad Update(t.right, t.OPEcipher, Max)$
5: **end if**

Algorithm 4. MPOPE Encryption ENCRYPTION

Input: DP_k, S, n_k, $DETcipher_k$.
State: The AVL tree T of nodes $\{t\}$.

1: DP_k invokes Algorithm 2 to refresh the secret state.
2: S invokes Algorithm 3 to update the OPE ciphertexts.

Algorithm 5. MPOPE Decryption DECRYPTION

Input: $OPEcipher$.
Output: $DETcipher$.
State: The AVL tree T of nodes $\{t\}$.

1: Search $OPEcipher$ on the AVL tree.
2: **if** $t.OPEcipher = OPEcipher$ **then**
3: $\quad$ **return** $t.DETcipher$
4: **end if**

We provide an example to describe our scheme in Fig. 2. In Fig. 2, DP_A, DP_B, and DP_C provide plaintexts $\{15, 19, 81\}$, $\{3, 1, 14\}$, and $\{91, 15, 15\}$ respectively. Later those three data providers use DET encryption scheme to encrypt their data respectively. Then each data provider helps the cloud server to construct the secret state (the AVL tree) by invoking the REFRESH Algorithm (Algorithm 2). Note that DP_C only insert $\{91, 15\}$ in the secret state because repeated plaintext 15 only insert once. In the secret state, we can find that identical plaintexts 15 provided by DP_A and DP_C have different position in

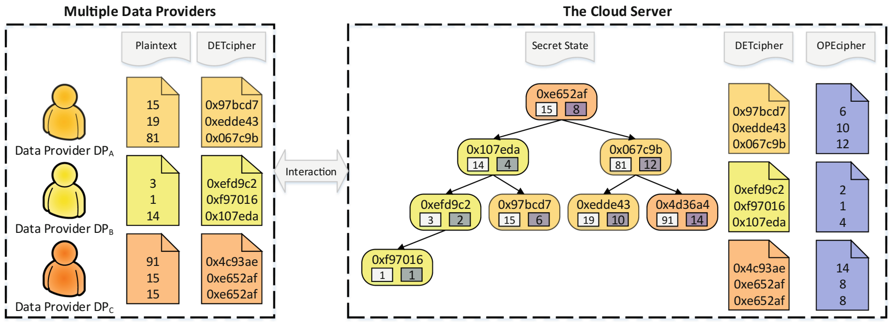


Fig. 2. Overview of our MPOPE scheme. Our MPOPE scheme involves 3 steps: Firstly, each data provider uses DET encryption to encrypt their data. Secondly, data providers help the cloud server to construct the secret state, which only involves the DET ciphertexts. Thirdly, the cloud server generates the OPE ciphertexts by using the secret state. Note that the left rectangles in the node of secret state denotes the plaintext provided by data providers, but there are not stored in the cloud server.

the AVL tree because our comparing protocol randomize the compare result of 15 provided by different data provider. After constructing the secret state, the cloud server invokes the UPDATE Algorithm (Algorithm 3) to generate the OPE ciphertexts. Finally, we can find that the corresponding ciphertexts of plaintexts $\{15, 19, 81, 3, 1, 14, 91, 15, 15\}$ are $\{6, 10, 12, 2, 1, 4, 14, 8, 8\}$.

4 Theoretical Analysis and Discussion

4.1 Security Analysis

Security Proof. We assume that DET encryptions are computationally indistinguishable from random values. Recall our security notion defined in Sect. 2.3. We provide the security goal of our scheme in Theorem 1.

Theorem 1. Our multi-provider order-preserving encryption scheme is secure against multi-provider ordered chosen plaintext attack. Namely, our scheme is IND-MPOCPA secure.

Proof. Due to space constraints, we provide a formalized proof in our extended paper, and we provide intuition here.

We prove Theorem 1 by induction. Consider that when no value was encrypted, then our scheme starts with the same initial state which is independent of the bit b. Then we assume that it holds for i encryptions. In the $(i+1)$-th encryption, we assume that $c_{*,i+1}$ was produced by DP_k and hence $c_{*,i+1}$ is $c_{DP_k,i+1}$. We have three possibilities.

The first one is $w^b_{DP_k,j} = w^b_{DP_k,i+1}$ and $j < i+1$. The secret state of both sequences will not change, and the OPE cipher of $w^b_{DP_k,i+1}$ will equal to $w^b_{DP_k,j}$. Since $c_{DP_k,j}$ is independent of b, $c_{DP_k,i+1}$ is independent of b.

The second is $w^b_{DP_k,i+1} = w^b_{DP_t,j}$, and $j < i+1$. Then the secret state will be refreshed, and the result of refreshment is depended on a random coin b_k. Since b_k is randomly chosen by DP_k and is independent of b, $c_{DP_k,i+1}$ is independent of b.

The last is that plaintext $w^b_{DP_k,i+1}$ has not been encrypted. DP_k interacts with the cloud server and refreshes the secret state. Since W^0 and W^1 have the same order relation, the secret state of both plaintexts are the same. Therefore, $c_{DP_k,i+1}$ is independent of b.

Therefore, our encryption algorithm produces the same OPE ciphertext sequence in both cases, and hence our scheme is IND-MPOCPA secure. □

4.2 Theoretical Performance Analysis

We analyze the time complexity of our scheme.

`Key Generation`. In our scheme, the secret state plays a role as the key of our encryption scheme. Hence, the complexity of our key generation algorithm is the complexity of the initiation of secret state, which requires $\mathcal{O}(1)$.

`Encryption`. The encryption involves Algorithms 2 and 3. Algorithm 2 requires to refresh distinct plaintexts. Kerschbaum and Schroepfer [4] investigated the expected number of distinct plaintexts, and we restate it in Theorem 2.

Theorem 2. Let D be the number of distinct plaintexts in the plaintext domain. For a uniformly chosen plaintext sequence of size n with S distinct plaintexts, the expected number of distinct plaintexts is

$$E[S] = D(1 - (\frac{D-1}{D})^n) \tag{1}$$

Let N be the total number of values in the secret state. We conclude the expected value of N in Lemma 1 by using the Eq. 1.

Lemma 1. The expected number of N is

$$E[N] = \sum_{k=1}^{K} D(1 - (\frac{D-1}{D})^{n_k}) \tag{2}$$

Since the secret state is an AVL tree, which has logarithmic height, the time complexity of Algorithm 2 is $\mathcal{O}(\log N)$. The update Algorithm (Algorithm 3) is a pre-order traversal of the AVL tree, and hence the time complexity of it is $\mathcal{O}(N \log N)$. Since Algorithm 2 requires 8 times modular exponentiation computation per comparison, each secret state refreshment requires $\log N$ times complex computation, which requires more time than the operation of OPE ciphertext update. Hence, our encryption requires $\mathcal{O}(\log N)$ complex computation.

`Decryption`. The decryption algorithm is to find the corresponding DET ciphertext of an OPE ciphertext in the AVL tree and decrypt the DET ciphertext.

Assume that there are N values in the AVL tree, the time complexity of the decryption algorithm is $\mathcal{O}(\log N)$.

Hence, the time complexity of our key generation algorithm, encryption algorithm, and decryption algorithm are $\mathcal{O}(1)$, $\mathcal{O}(\log N)$, and $\mathcal{O}(\log N)$ respectively.

4.3 Ciphertext Domain

The ciphertexts of our scheme are generated by the secret state with N values. Let H be the height of an AVL tree, then $M = 2^H$. Foster [14] has investigated the relation between N and H, and we restate his work in Theorem 3.

Theorem 3. *N and H satisfy the following inequality:*

$$H < \frac{3}{2}\log_2(N+1) - 1 \tag{3}$$

Then, we can conclude that:

Lemma 2. *In order to store N values in an AVL tree, the minimum height of the tree is $H_{min} = \lceil \frac{3}{2}\log_2(N+1) - 1 \rceil$.*

Lemma 2 shows that the minimum bit length of a ciphertext is H_{min}. Hence, for N plaintext values, the ciphertext space M should not less than $2^{H_{min}}$. For simplicity, We define that $M = 2^{\lceil H_{min} \rceil}$.

5 Experiments

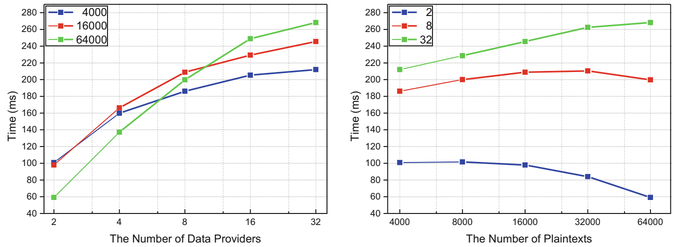
We evaluate the efficiency and the statistical security of our scheme (**MPOPE**). We use **DOPE** and **FHOPE** to denote the scheme in [4] and the scheme in [5] respectively. The result of our experiments answer the following questions:

- How is the MPOPE encrypting time affected by the number of data providers and the number of plaintexts?
- How does the encryption time of MPOPE compare with DOPE and FHOPE?
- How does the statistical security of MPOPE compare with DOPE and FHOPE?

We implement our experiments in Java 1.6. Our experiments are carried out on a 64-Bit workstation with an Intel Xeon E-1226 CPU with 3.30 GHz and 32 GB RAM. We set D and M to be 16000 and 2^{25} respectively. In our experiments, each data provider encrypts the same number of plaintexts. We set the key length of Paillier cryptosystem to be 1024 bits.

5.1 The Encrypting Time of Our Scheme

We evaluate the average encrypting time when 2, 4, 8, 16, 32 data providers encrypt 4000, 16000, 64000 total plaintexts in Fig. 3a. Figure 3a depicts that when the number of providers grows, the average encrypting time grows slightly. We also measure the average encrypting time when 2, 8, 32 data providers encrypt 4000, 8000, 16000, 32000, 64000 total possibly identical plaintexts in Fig. 3. Figure 3b depicts that the average encrypting time firstly increases and then decreases when the total number of plaintexts increases.



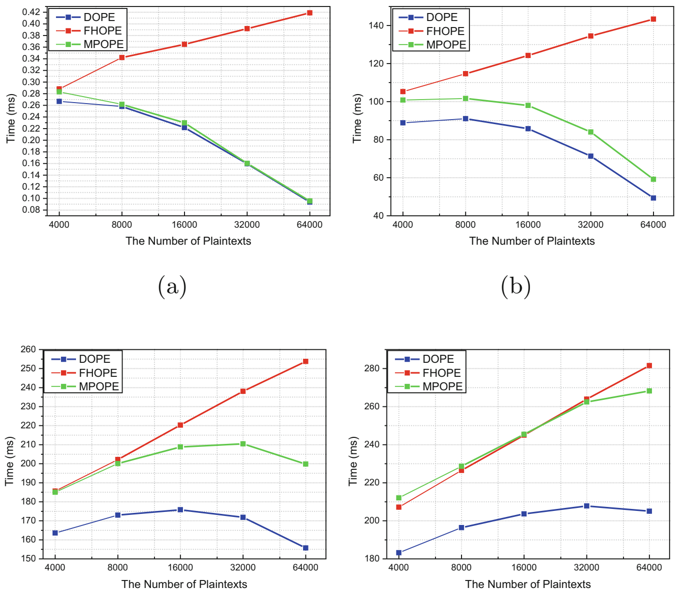
(a) (b)

Fig. 3. (a) and (b) depict the encrypting time of MPOPE affected by the number of data provider and the number of plaintext respectively.

5.2 A Comparison to Previous OPE Schemes

We extend DOPE and FHOPE to the multi-provider environment by using our comparing protocol. We use the AVL tree as the state of those schemes to improve the efficiency of insertion and searching.

We compare the average encrypting time of MPOPE with DOPE and FHOPE. We evaluate the average encrypting time of those three schemes when 1, 2, 8, 32 data providers encrypt 4000, 8000, 16000, 32000, 64000 possible repeated plaintexts in Fig. 4a, b, c, and d respectively.



(a) (b)

(c) (d)

Fig. 4. (a)–(d) depict a comparison of encrypting time between MPOPE, DOPE, and FHOPE in 1, 2, 8, 32 data provider environment respectively.

Overall, those figures depict that the time overhead of MPOPE is lower than FHOPE but higher than DOPE. Therefore, the efficiency of MPOPE is better than FHOPE but worse than DOPE.

5.3 Statistical Security

We measure the effectiveness of statistical attack for our scheme by estimating the Pearson correlation coefficient between plaintexts and ciphertexts. The smaller the correlation, the more secure against statistical cryptanalysis.

We make 300 experiments to evaluate the Pearson correlation coefficient for 4000, 8000, 16000, 32000, 64000 plaintext-ciphertext pairs. We compute the 90% confidence intervals as error bars. We compare the Pearson correlation coefficient of the plaintext-ciphertexts pairs generated by MPOPE to DOPE and FHOPE. The compare results in 1, 2, 8, 32 data provider environment are depicted in Fig. 5a, b, c, and d respectively. Overall, we find that the confidence intervals of the correlation coefficient for each different cases clearly overlap. Hence, we can conclude that MPOPE is no weaker than DOPE and FHOPE under the statistical attack.

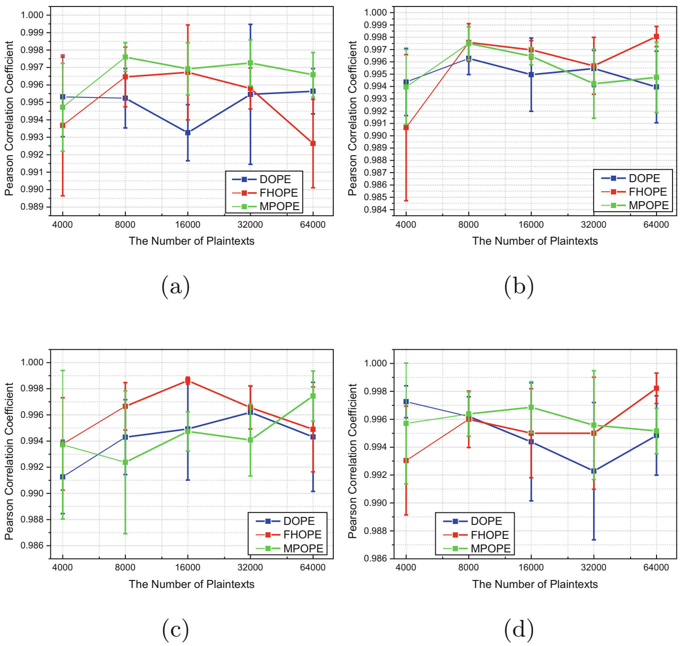


Fig. 5. (a)–(d) describe a comparison of Pearson correlation coefficient of the plaintext-ciphertexts pairs generated by MPOPE, DOPE, and FHOPE in 1, 2, 8, 32 data provider environment respectively.

6 Conclusions

We propose the IND-MPOCPA security notion for multi-provider order-preserving encryption. Moreover, we construct MPOPE which captures IND-MPOCPA. In summary, our scheme is a new option for order-preserving encryption in the cloud, which provides strong security guarantee with operation efficiency for cloud applications with multiple data providers.

Acknowledgement. This work is partially supported by the National Science Foundation of China under Grant No. 61472131, 61300218, 61472132, 61300217; Science and Technology Key Projects of Hunan Province (2015TP1004, 2015SK2087, 2015JC1001, 2016JC2012).

References

1. Zhang, Y., Chunxiang, X., Liang, X., Li, H., Yi, M., Zhang, X.: Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. IEEE Trans. Inf. Forensics Secur. **12**(3), 676–688 (2017)
2. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2010, 15–19 March 2010, San Diego, CA, USA, pp. 441–445. IEEE (2010)
3. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, 19–22 May 2013, pp. 463–477. IEEE Computer Society (2013)
4. Kerschbaum, F., Schroepfer, A.: Optimal average-complexity ideal-security order-preserving encryption. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 275–286. ACM, New York (2014)
5. Kerschbaum, F.: Frequency-hiding order-preserving encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015, pp. 656–667. ACM, New York (2015)
6. Roche, D.S., Apon, D., Choi, S.G., Yerukhimovich, A.: Pope: partial order preserving encoding. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 1131–1142. ACM, New York (2016)
7. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, SIGMOD 2004, pp. 563–574. ACM, New York (2004)
8. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_13
9. Lewi, K., Wu, D.J.: Order-revealing encryption: new constructions, applications, and lower bounds. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 1167–1178. ACM, New York (2016)
10. Xiao, L., Yen, I.-L., Huynh, D.T.: Extending order preserving encryption for multi-user systems. IACR Cryptology ePrint Archive, 2012:192 (2012)

11. Yao, X., Lin, Y., Liu, Q., Long, S.: Efficient and privacy-preserving search in multi-source personal health record clouds. In: 2015 IEEE Symposium on Computers and Communication, ISCC 2015, Larnaca, Cyprus, 6–9 July 2015, pp. 803–808. IEEE Computer Society (2015)
12. 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
13. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
14. Foster, C.C.: Information retrieval: information storage and retrieval using AVL trees. In: Proceedings of the 1965 20th National Conference, ACM 1965, pp. 192–205. ACM, New York (1965)