2021

A Practical and Secure Stateless Order Preserving Encryption for Outsourced Databases

Ning Shen
Boise State University

Jyh-Haw Yeh
Boise State University

Hung-Min Sun
National Tsing Hua University

Chien-Ming Chen
Shandong University of Science and Technology

 https://doi.org/10.1109/PRDC53464.2021.00025

A Practical and Secure Stateless Order Preserving Encryption for Outsourced Databases

Ning Shen
Ph.D in Computing
Boise State University, USA
ningshen@u.boisestate.edu

Jyh-haw Yeh
Department of Computer Science
Boise State University, USA
jhyeh@boisestate.edu

Hung-Min Sun
Department of Computer Science
National Tsing Hua University, Taiwan
hmsun@cs.nthu.edu.tw

Chien-Ming Chen
College of Computer Science and Engineering
Shandong University of Science and Technology, China
chienmingchen@ieee.org

***Abstract*—Order-preserving encryption (OPE) plays an important role in securing outsourced databases. OPE schemes can be either Stateless or Stateful. Stateful schemes can achieve the ideal security of order-preserving encryption, i.e., "reveal no information about the plaintexts besides order." However, comparing to stateless schemes, stateful schemes require maintaining some state information locally besides encryption keys and the ciphertexts are mutable. On the other hand, stateless schemes only require remembering encryption keys and thus is more efficient. It is a common belief that stateless schemes cannot provide the same level of security as stateful ones because stateless schemes reveal the relative distance among their corresponding plaintext. In real world applications, such security defects may lead to the leakage of statistical and sensitive information, e.g., the data distribution, or even negates the whole encryption. In this paper, we propose a practical and secure stateless order-preserving encryption scheme. With prior knowledge of the data to be encrypted, our scheme can achieve IND-CCPA (INDistinguishability under Committed ordered Chosen Plaintext Attacks) security for static data set. Though the IND-CCPA security can't be met for dynamic data set, our new scheme can still significantly improve the security in real world applications. Along with the encryption scheme, in this paper we also provide methods to eliminate access pattern leakage in communications and thus prevents some common attacks to OPE schemes in practice.**

***Index Terms*—Range Query over Encrypted Databases, Order Preserving Encryption, Non-deterministic OPE**

I. Introduction

Encryption is a powerful tool to protect data confidentiality and privacy, especially in the cloud environment where data is outsourced to a third party which can't be fully trusted. However, encryption usually will also disable or increase the difficulty of data processing. Traditional encryption schemes don't support sorting on the encrypted data because the order of plaintexts is not persevered in the corresponding ciphertexts. Under this circumstances, in order to perform sorting, the data needs to be decrypted first. Since the cloud is not fully trusted, allowing the server to perform data decryption will result in leaking sensitive information to untrusted third parties.

One important practice of computation primitives over encrypted data is order-preserving encryption (OPE). Such encryption allows ciphertexts to reserve the same order of plaintexts. That is, if $x > y$, then $Enc(x) > Enc(y)$. This characteristic of OPE schemes allows the untrusted server to perform comparison, ordering, ranking, and range queries over encrypted data. OPE already has a wide range of applications, including encrypted SQL database [4][5][6][17][8][9], mail servers [10], and web applications. For instance, in order to process a range query over encrypted data with a range condition on Age (e.g., $a < Age < b$), the untrusted server only needs to know the values of $Enc(a)$ and $Enc(b)$, and then selects all the encrypted data whose value sits in between $Enc(a)$ and $Enc(b)$. Thus, using order-preserving encryption, the range query processing will be much more efficient while data confidentiality can still be preserved.

The security notion IND-CPA (INDistinguishability against Chosen Plaintext Attack) is used to describe the security strength of traditional encryption algorithms. Different from traditional encryption schemes, OPE reveals the order-relations amongst ciphertexts and thus it cannot satisfy the traditional security notion IND-CPA. Boldryeva et al. [12] first proposed a security notion for OPE called *indistinguishability under ordered chosen-plaintext attack* or IND-OCPA. After IND-OCPA, some other more relaxed OPE security notions have been proposed in [12][14]. These security notions are weakened versions of IND-OCPA and thus leak more information than just ordering.

Boldyreva's proof that no pratical OPE scheme can achieve IND-OCPA is based on two assumptions: the encryption is stateless and the ciphertexts are immutable. Without these two assumptions, designing an OPE scheme satisfying IND-OCPA is possible. Popa et al. in [11] proposed the first such ideal-secure but stateful OPE scheme that achieves IND-OCPA. Popa's scheme is stateful because a tree structure is needed to trace all the previous plaintexts/ciphertexts assignments. On the contrary, Boldyreva's scheme is stateless because its encryption and decryption do not rely on existing assignments.

In real world, the choice from various OPE schemes depends on the applications. Stateful schemes can achieve higher security notion IND-OCPA, but they require retaining a local status copy of the data. Keeping a local status copy is

contradictory to the purpose of database outsourcing (storage outsourcing), and the mutable ciphertexts may cause large updates in existing database. On the other hand, for stateless schemes, several attacks proposed in literature have already shown that attackers may be able to recover a large portion of sensitive data from the encrypted database.

In this paper, we propose a practical and secure stateless OPE scheme. Though the scheme uses the same hypergeometric distribution (HGD) algorithm from [12], it introduces the concept of using *Density Function* to guide ciphertexts' distribution. Unlike previous approaches that tried to make stateless schemes more secure [4][13][17], our scheme has many advantages with the following contributions:

1) We propose a new OPE scheme which is able to encrypt data on the fly. The proposed scheme can achieve IND-CCPA. That is, if the dataset is pre-known before encryption, our scheme IND-CCPA secure (see Section V for the security analysis). Furthermore, even if the dataset is not pre-known, our algorithm can encrypt data on the fly. We argue that by using some public known auxiliary datasets, our scheme can still significantly improve the security of the encrypted databases.
2) We show that many attacks which can successfully compromise OPE encrypted databases can be easily relieved by our changes. Most of these attacks are powerful and some of them can even fully negate OPE encryptions. In this paper, we show that by just modifying existing OPE schemes using a proposed methodology, we can fully prevent such attacks without having to design any new encryption algorithm.
3) We prove that the security of our scheme is under IND-CCPA (*INDistinguishability under Committed ordered Chosen Plaintext Attacks*). We also explain how IND-CCPA can perfectly describe our scheme and show why it is meaningful in real world applications.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes the security model, followed by presenting our OPE scheme in Section IV. Section V proves the security of our scheme. In Section VI, we use our proposed schemes to encrypt real world databases and compare the results. Section VII concludes the paper.

II. Related Work

Some database encryption works have been done using symmetric-key and public-key cryptography [1][2][3]. However, these research work mainly focuses on the search with exact keyword match while left the range query still an unmet need. Agrawal et al. [4] was the first group to define the OPE primitive and developed a such encryption scheme. In OPE, because the ciphertexts reserve the ordering of plaintexts, range queries over the OPE encrypted database become possible. Besides proposing the order-preserving property, they also raised the concept of *estimation exposure* that an adversary can estimate the value of plaintext in a tight bound. Another problem of traditional OPE schemes is that the distribution of ciphertexts may somehow leak the data distribution of input plaintexts. Based on these observations, their algorithm takes a target distribution as input. The algorithm first partitions the plaintexts and target distribution data into several buckets, then flattens each bucket and builds mappings between each input and target distribution one by one. Their scheme could hide the data distribution and is robust against estimation exposure attack. However, this scheme is rather ad-hoc and is not able to encrypt data on the fly. All the bucket-to-bucket mapping is built consecutively and the previous states in the encryption have to be reused. According to their description, the key size grows with the number of buckets during the encryption process and thus it is not a stateless OPE scheme. Also, it requires the entire dataset as input before it can encrypt any data item, which is not practical.

The first formal security notion of order-preserving encryption was provided by Boldyreva et al. [12]. They introduced a security notion *indistinguishability under ordered chosed plaintext attack* (IND-OCPA), which requires ciphertexts revealing nothing except the ordering. They proved that no stateless scheme can achieve IND-OCPA and introduced a weakened notion *pseudorandom order-preserving function under chosen-ciphertext attacks* (POPF-CCA). They also designed a practical encryption algorithm which is POPF-CCA secure. Their algorithm uses hypergeometric distribution to generate mappings between plaintext and ciphertext, and the client only needs to store the encryption key (thus is stateless). However, the scheme is less secure than Agrawal et al's scheme [4] because it is vulnerable to the estimation exposure attack.

After IND-OCPA, some other security definitions and stateless schemes were proposed. Xiao et al. [14] weakened IND-OCPA to IND-OLCPA (*indistinguishability against ordered large chosen-plaintext attack*), where an adversary can only challenge "nearby" values. IND-OLCPA is a theoretical notion and no real world system can enforce it. Boldyreva et al. [13] also raised the concept of *indistinguishability under committed chosen plaintext attacks* (IND-CCPA), which allows the encryption key being generated after the adversary chooses the challenge vectors (i.e., data is known before generating the encryption key). Wozniak et al. [18] designed algorithms based on random offset addition, random uniform sampling, and random subrange selection. These schemes are either stateful or only secure under special prerequisite.

The first IND-OCPA scheme was proposed by Popa et al. [11]. This scheme is stated as a stateful OPE because there is a tree structure stored on the server side. In this scheme, they do not use any OPE encryption at all but use the standard and more secure algorithm such as AES. The tree structure stores all the unique ciphertexts in a binary search tree fashion and the relative order of each ciphertext is presented by the path from the root to the particular ciphertext node. To encrypt a number a, starting from the root, the server returns current node (i.e., the ciphertext contained in the node) to the client. Client decrypts the current node and compares it with a. The comparison result will be sent back to the server so that server can decide which child to search in the next round until the server finds the right place in the tree to store the new cipher-

text for a. One obvious shortcoming of this approach is that it is a multi-round protocol, which significantly increases the communication overhead. Furthermore, such scheme is also called *mutable order-preserving encoding* or mOPE, because the ciphertexts is mutable. Mutable schemes occasionally have to update the value of ciphertexts and these updates causes performance implications on the server.

To make mOPE more efficient, another mutable IND-OCPA scheme was provided by Kerschbaum and Schröpfer [15]. They move the tree structure from the server side to the client side, thus the communication cost is erased. Also, they reduce the possible rounds of ciphertext mutation to improve algorithm's efficiency. However, the information stored on the client side is still linear to the number of distinct plaintexts, which is the same as Popa et al.'s scheme. They also improved the scheme to provide non-deterministic OPE encryptions, where the same plaintext will be encrypted to different ciphertexts every time, thus hides the frequency of the plaintexts and improve the security. However, the trade off is that the number of nodes in the tree is increased to the number of unique ciphertexts in the database, which can be huge.

On the flip side, as various protocols have been proposed, researchers also show interests in how to compromise these OPE encrypted databases. Naveed et al. [20] proposed several inference attacks on property-preserving encrypted databases. They developed four attacks: *frequency analysis*, *l_p-optimization*, *sorting attack*, and *cumulative attack*. Both frequency analysis and l_p-optimization target at the DTE (deterministic)-encrypted columns. Frequency analysis analyzes some "well-correlated" auxiliary dataset and then compares the frequency similarity between the auxiliary dataset and the encrypted database, while l_p optimization uses l_p norms and combinatorial optimization techniques to reveal the mappings between ciphertext and plaintext. Sorting attack targets at dense columns and it also relies on the help of auxiliary datasets. Cumulative attack is more sophisticated and applicable to both high- and low-density columns.

Kellaris et al. [19] proposed another generic attack on encrypted databases. Under a weak passive adversarial model, this attack only requires the capability of eavesdropping network traffic of range queries between server and client, and no prior knowledge is needed. By observing the access pattern or even communication volume cumulatively, the eavesdropper can re-construct the secret attributes of the database. Lacharité et al. [21] improved the construction attack by reducing its cost. In Kellaris et al.'s attack, it expects at least $O(N^2 \log N)$ queries to recover every record (N is the number of distinct plaintexts). Lacharité et al. reduced the required queries to $N \log N + O(N)$. Also, They provided an *approximate reconstruction attack* which only requires $O(N)$ queries and the error ratio is within a constant. This more advanced construction attack can fully negate encryption in seconds.

All existing stateless OPE schemes suffer from the above attacks. In this paper, we developed our scheme and proposed some techniques to prevent or at least mitigate these attacks.

III. Security Model

1) Model architecture: The architecture in our paper is typical which contains two parties: client and server. The client is the data owner while the server is the host that provides storage and calculation capability of the data. The client can create, delete, query and update the data to the outsourced database through a communication protocol between them and all data transmitted in the protocol are encrypted. Though most service providers are not malicious, their honest behavior still cannot be guaranteed. Thus, it is appropriate to assume a semi-honest model in which the server will strictly execute the instructions from the client, but it may be curious to learn as much information as possible.

2) Adversarial model: We consider the outsourced database is encrypted by an OPE scheme and there are two different adversaries under the model: the honest-but-curious server and the eavesdropper over the Internet. Both the server and eavesdropper are treated as a persistent and passive adversary that they can observe all the communication between the server and client. For example, the volume of data being transmitted or what data is transmitted (eavesdropper may not be able to observe this). Besides the transmitted data, the server persistently has access to the data, which includes the rank information, density of the database, and the distribution of all encrypted data. With such information, the adversary can launch powerful attacks to recover the values of all encrypted records with high accuracy [19][21]. In Section 4, we will discuss how our proposed scheme can mitigate these attacks.

IV. Our Scheme

Our goal is to develop a secure and stateless (and thus practical) OPE scheme which is easy to adopt and able to defend most existing attacks. The construction of our scheme follows the security notion of IND-CCPA (*indistinguishability under committed chosen plaintext attacks*), where if database is available before encryption, the scheme can achieve IND-CCPA. If the database is not available in advance, but if an auxiliary sample dataset in the same application is available, our scheme can still significantly improve the security of the OPE encrypted databases. In reality, such auxiliary datasets can be historic data or sample data that share some common data distribution characteristics. For the worst case where data owner doesn't have any pre-knowledge or auxiliary datasets, our scheme can still make it more difficult to reveal the plaintext and statistical information from the encrypted data.

A. Approach

We built our scheme based on the Boldyreva et al's OPE scheme [12]. They relate the encryption scheme to a probabilistic game and generate the ciphertexts using *hypergeometric distribution* (HGD). To have an intuitive understanding about their approach, let's consider a probability game as follows. Suppose there are N balls in a pocket, where M out of N balls are marked red and the $N-M$ remaining balls are white. If someone draws balls without replacement, then after drawing y balls, the number of 'marked balls' picked x

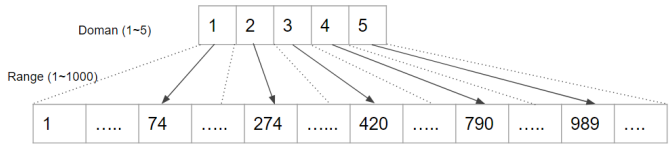


Fig. 1: An OPE demo with Domain 5 and Range 1000

should follow the hypergeometric distribution with the PMF (Probability Mass Function) as

$$PMF = \frac{\binom{y}{x}\binom{N-y}{M-x}}{\binom{N}{M}} \tag{1}$$

Now we just need to understand the connection between HGD and an *Order-preserving Function* (OPF). let's consider $M = \{1, 2, \ldots, m\}$ as the plaintext domain (Domain) and $N = \{1, 2, \ldots, n\}$ as the cipheretxt range (Range), where $n > m$. An order-preserving function creates ordered mappings from M to N. Fig. 1 demonstrates the mapping idea. Every number j in the $Domain$ M is mapped to a range R_j in $Range$ N (i.e., $R_j \subset N$) and a random number in R_j is returned as the ciphertext of j. To relate the encryption principle and the game, if we consider the total number of balls N as the ciphertext range and M marked balls as the plaintext domain, then we can figure out how many points in domain lies below a point in range by treating the range points as the "samples" in the hypergeometric game above. That is, given a key k to determine the randomness, N total balls with M of them are marked red. We use a function to deterministically simulate the hypergeometric distribution to decide the plaintext-ciphertext mapping. For instance, picking balls randomly without replacement, if after picking y-th ball we pick the x-th red ball and after picking y'-th ball we pick the $x+1$-th red ball, then the range $[y, y'-1]$ is mapped to x and a random number in $[y, y'-1]$ is the ciphertext of x.

Inspired by the design above, we change the algorithm that in each round, it requires one more step after the original HGD mapping. Besides drawing balls, there are some boxes acting as ball holders. The number of boxes is equal to the plaintext size in our algorithm and the number of marked balls increase to a value between plaintext and ciphertext size, e.g.,$100*$(plaintext size). These boxes are placed in a fixed order. The capacity of each box varies but the total capacity of all boxes is equal to the volume of all marked balls. In our model, for each round, we still use *hypergeometric distribution* to simulate the ball drawing process. However, after drawing balls, each of the marked balls being picked will be put into a box in the predetermined boxes' order. For example, if we draw y balls and x of them are marked balls, starting from the first box, we put the marked balls into a box until it is full and then continuously fill the next box in the predetermined order. If all the marked balls picked can fill t boxes (includes the last one which may not be full), then y is mapped to t rather than x. Next t will be compared to the plaintext input (to be encrypted) to determine which half will be used for the next recursive round of the binary search. In this new model, the plaintext-ciphertext mapping is not only determined by the randomness of HGD, but also the arrangement of all boxes' capacities. A box with a large capacity means more attempts to grab enough marked balls to fill it. Thus, from the order-preserving function perspective, the plaintext which this box represents will take more space in the ciphertext range.

B. Modeling the distribution

As described above, the capacity arrangement of all the boxes has a significant impact on our encryption scheme. Thus, our scheme keeps the capacity distribution of all boxes as a part of the key. Based on this, our algorithm requires an efficient way to model the capacity distribution of all boxes. However, we don't expect the modeling method to be very precise and an approximate modeling approach would be enough. In this paper we will use the *Gaussian mixture model* (GMM) to model the data distribution in our algorithm. In data science, *Gaussian Distribution* is one of the most basic and common technique used to model real-world unimodal data [26]. The Gaussian Distribution is written as

$$P(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{2}$$

where μ is the mean of the distribution and σ is the standard deviation. For complex datasets that a single Gaussian Distribution cannot represent, people use GMM to model multimodal data. GMM is the mixture distribution of many single *Gaussian distributions* and the total size is normalized to 1. There are many existing tools we can use to model the data with GMM, for example scikit-learn [25]. Since modeling the data approximately is good enough for our approach, we can limit the number of Gaussian distributions to a relatively small number, for example 3 or 5 with different accuracy. Thus, the key size of our scheme is limited to a constant number of (μ, σ) pairs regardless of the size and distribution of data. Thus, our scheme is stateless.

C. Algorithm

Since our algorithm is built based on Boldyreva et al's scheme, it is essential to explain their algorithm first. In order to implement an order-preserving function, they designed two algorithms **LazySample** and **LazySampleInv** to simulate the hypergeometric distribution and generate the plaintext-ciphertext assignments. The pseudocode of these two algorithms are given in Algorithms 1 & 2 in black and the codes in red are added/modified codes for our algorithm. LazySample accepts a plaintext m as one of its inputs, and returns the ciphertext of m. The LazySample algorithm builds mappings from "range gap" N to "domain gap" M in a binary search manner, where $M \ll N$. A "gap" is the boundary between two consecutive points in the range or domain. During encryption, the algorithm first maps the middle range gap $y = \frac{N}{2}$ to a domain gap. Line 13 describes this step that it uses hypergeometric distribution to decide the number of points in D which will be mapped to a range between the minimum ciphertext (i.e., r in the pseudocode) and the point y. The result from Line 13 is stored in a map I. The range point y is

between two gaps in M, i.e., $Enc(x) \leq y < Enc(x+1)$ and $x = d + I[D, R, y]$, where d is the minimum domain gap in M. This is equivalent to the fact that the range gap between y and $y+1$ is mapped to domain gap between x and $x+1$. In line 17, the **LazySample** algorithm compares domain gap x with the input m. Based on the result, the algorithm will be repeated recursively either on the left half $r+1, \ldots, y$ or the right half $y+1, \ldots, r+N$ until the ciphertext range cannot be divided anymore.

Algorithm 1 LazySample_Revised$(D, T, R, Gauss, m)$

1: $M \leftarrow |D|; N \leftarrow |R|; O \leftarrow |T|$
2: $d \leftarrow min(D) - 1; r \leftarrow min(R) - 1; t \leftarrow min(T) - 1$
3: $y \leftarrow r + [N/2]$
4: **if** $|D| = 1$ **then**
5: **if** $F[D, R, m]$ is undefined **then**
6: $cc \leftarrow GetCoins(1^{l_R}, D, R, 1||m)$
7: $F[D, R, M] \leftarrow r$
8: Return $F[D, R, m]$
9: **end if**
10: **end if**
11: **if** $I[D, R, y]$ is undefined **then**
12: $cc \leftarrow GetCoins(1^{l_1}, D, R, 0||y)$
13: $I[D, R, y] \leftarrow HG(M, N, y - r; cc)$
14: **end if**
15: $x \leftarrow d + I[D, R, y]$
16: $z \leftarrow GetGMM(D, T, Gauss, x)$
17: **if** $m =\leq \not{z}z$ **then**
18: $D \leftarrow \{d+1, \ldots, x\}$
19: $R \leftarrow \{r+1, \ldots, y\}$
20: $T \leftarrow \{t+1, \ldots, z\}$
21: **else**
22: $D \leftarrow \{x+1, \ldots, d+M\}$
23: $R \leftarrow \{y+1, \ldots, r+N\}$
24: $T \leftarrow \{z+1, \ldots, t+0\}$
25: **end if**
26: Return **LazySample_Revised**$(D, T, R, Gauss, m)$

The main difference between two algorithms lies in Line 16. In the Boldyreva et al's algorithm, after each domain gap x is returned, x is directly compared with input m to decide the direction of next binary search. For our algorithm, after generating x, before next binary search, the function $GetGMM()$ will be called and it returns a new value z, which will be the value that compared with m to decide next binary search. Intuitively, $GetGMM()$ calculates how many boxes will be filled (including the last not-fully-filled box) by the picked marked balls. $GetGMM()$ takes four inputs:

1) A density function $\mathcal{F}$ which is depicted by a Gaussian mixture model. A density function describes the data density of the dataset to be encrypted. Intuitively, $\mathcal{F}$ can be considered as a function used to describe the capacity distribution of all the boxes. For example, if there are T boxes and total capacity are N balls, the total capacity from box 1 to box 5 can be calculated as $\frac{\int_0^5 \mathcal{F}}{\int_0^T \mathcal{F}} \times N$.
2) A domain T which indicates the domain of boxes for next binary search.
3) A range D which indicates the range (capacity of all boxes in T) for next binary search.
4) A number x, which is the output from HGD (number of marked ball picked).

Instead of using x, the output z from $GetGMM()$, which represents the index of the last filled boxes (i.e., the number of boxes required to hold all marked balls picked) will be compared with the input point m to determine the direction of next binary search (see Line 17 in Algorithm 1).

Algorithm 2 LazySampleInv_Revised$(D, T, R, Gauss, c)$

1: $M \leftarrow |D|; N \leftarrow |R|; O \leftarrow |T|$
2: $d \leftarrow min(D) - 1; r \leftarrow min(R) - 1; t \leftarrow min(T) - 1$
3: $y \leftarrow r + [N/2]$
4: **if** $|D| = 1$ **then** $m \leftarrow min(D)$
5: **if** $F[D, R, m]$ is undefined **then**
6: $cc \leftarrow GetCoins(1^{l_R}, D, R, 1||m)$
7: $F[D, R, M] \leftarrow r$
8: **if** $F[D, R, m] = c$ return m **then**
9: **else** return $\perp$
10: **end if**
11: Return $F[D, R, m]$
12: **end if**
13: **end if**
14: **if** $I[D, R, y]$ is undefined **then**
15: $cc \leftarrow GetCoins(1^{l_1}, D, R, 0||y)$
16: $I[D, R, y] \leftarrow HG(M, N, y - r; cc)$
17: **end if**
18: $x \leftarrow d + I[D, R, y]$
19: $z \leftarrow GetGMM(D, T, Gauss, x)$
20: **if** $m =\leq \not{z}z$ **then**
21: $D \leftarrow \{d+1, \ldots, x\}$
22: $R \leftarrow \{r+1, \ldots, y\}$
23: $T \leftarrow \{t+1, \ldots, z\}$
24: **else**
25: $D \leftarrow \{x+1, \ldots, d+M\}$
26: $R \leftarrow \{y+1, \ldots, r+N\}$
27: $T \leftarrow \{z+1, \ldots, t+0\}$
28: **end if**
29: Return **LazySampleInv_Revised**$(D, T, R, Gauss, m)$

Algorithm 3 describes how the function $GetGMM()$ works. It uses binary search to find the index of the last box filled. In Line 4, the function $GaussPercent()$ calculates the capacity percentage of boxes in $(min(T), p)$ over all boxes in $(min(T), max(T))$, which p is the index of middle box in $(min(T), max(T))$. Based on the density function $\mathcal{F}$, the capacity percentage can be calculated as $pct = \frac{\int_{min(T)}^{p} \mathcal{F}}{\int_{min(T)}^{max(T)} \mathcal{F}}$. After knowing the capacity percentage, Line 5 calculates the capacity x' from all boxes in $(min(T), p)$. After that, x' will be compared with x to decide the direction of next binary search. The algorithm will exit the binary search when all the marked balls have been filled and it returns the index

of last box used. The inverse function of *LazySample* is *LazySampleInv* which is used for decryption. The pseudocode of our modified *LazySampleInv* is in Algorithm 2.

Algorithm 3 GetGMM$(D, T, Gauss, x)$

```
M = |D|; O = |T|
t = min(T); step ← [O/2]; p ← t + step
while step ≥ 1 do
    pct = GaussPercent(Gauss, min(T), max(T), p)
    x' = pct * (max(D) − min(D))
    step ← step/2
    if x' < x then p ← p + step
    else p ← p − step
    end if
end while
if x' ≥ x then return p − t
else return p − t + 1
end if
```

D. Non-determinism

A function is deterministic if a plaintext will be encrypted to the same ciphertext every time. *Frequency analysis* and *sorting attacks* are two attacks associated with the deterministic characteristics. *Frequency analysis* not only targets at order-preserving encryption, but also other property-preserving encryptions, while a *sorting attack* exploits order-preserving encryption only. *frequency analysis* and *sorting attacks* to a real-world application were presented in [20].

To prevent these attacks, we propose a method to make the OPE algorithm non-deterministic. We revise Boldyreva et al's algorithm by shifting the mappings. In Boldyreva et al's algorithm, any number j in $Domain\, M$ is mapped to a range R_j in $Range\, N$, and a fixed number in the range R_j will be returned as the ciphertext for j every time. If the encryption algorithm returns a random number within the range R_j rather than a fixed number, then the OPE algorithm becomes non-deterministic. However, to the best of our knowledge, there is no existing stateless OPE scheme which is non-deterministic.

CryptDB [9] is a famous encrypted database project that has an OPE implementation based on Boldyreva et al.'s algorithm. We extracted and modified the OPE implementation from CryptDB and proposed the non-deterministic OPE algorithm in Algorithm 4 below. Let the "original OPE" denote the original deterministic OPE implementation from CryptDB.

Algorithm 4: Non-deterministic OPE

1) For encrypting a number $m \in M$,
   a) Use the original OPE to encrypt m and its next number $m+1$. Let the two ciphertexts be $OPE_{Enc}(m)$ and $OPE_{Enc}(m+1)$ respectively, where the function $OPE_{Enc}()$ denotes the original OPE encryption function.
   b) Return a random value c in the range $[OPE_{Enc}(m),\ OPE_{Enc}(m+1)]$ as the non-deterministic ciphertext for m.
2) For decrypting the ciphertext c,
   a) Use the original OPE to decrypt c. Let the decrypted value be p (where p could be m or $m+1$).
   b) Use the original OPE to re-encrypt p and let the new cipher be c'.

   If $c > c'$, then the plaintext m of c is p.
   else the plaintext m of c is $p-1$.

With non-deterministic algorithm, given a plaintext m, the ciphertext will be a random number c between $OPE_{Enc}(m)$ and $OPE_{Enc}(m+1)$, where $OPE_{Enc}()$ is CryptDB's deterministic OPE encryption. This non-deterministic algorithm should be able to fully prevent the frequency analysis attack (because all ciphers, or in other words, data points are unique now), and also make the sorting attack much more difficult to conduct. However, on the flip side, if using this non-deterministic encryption algorithm to encrypt a database, all queries are required to be converted to range queries.

E. Fuzzing querying

For frequently used queries, if the result returned from the server is always the same, adversaries are able to learn some sensitive information by eavesdropping the network traffic [19][21]. Such **re-construction attack** utilizes **access pattern** to re-construct sensitive information. Comparing to those deterministic OPE algorithms, our non-deterministic OPE scheme can definitely reduce the access patterns available to adversaries, but it cannot eliminate all patterns. Over time, with enough queries been observed, adversaries could learn all the possible boundaries and thus are able to guess the number of unique plaintexts, and their orderings and frequencies. These information might be enough for a powerful adversary to retrieve meaningful statistic data or even recover ciphertexts. One solution to hide this access pattern would be using a *fuzzy querying* over OPE encrypted data. The details of the proposed fuzzy querying technique is described below.

Algorithm 5: Fuzzy querying

Let $Enc()$, $Dec()$ be the encryption and decryption functions of our non-deterministic OPE algorithm, respectively. $Enc(a)$ encrypts a to a random ciphertext within a range [$Enc(a)_{min}$, $Enc(a)_{max}$], where $Enc(a)_{min}$ is the lower bound and $Enc(a)_{max}$ is the upper bound of the range.

1) To convert a range query with a range $[a, b]$, where $a \leq b$, instead of using the actual boundaries $(Enc(a)_{min}, Enc(b)_{max})$, the client uses a random number between $Enc(a-1)_{min}$ and $Enc(a)_{min}$ as the lower bound, and another random number between $Enc(b)_{max}$ and $Enc(b+1)_{max}$ as the new upper bound.
2) Once the client receives result set S from the server, it decrypts all the data and discards all the data outside $[a, b]$. Those discarded results included in the return set is because we want to hide both the access pattern and real boundaries from eavesdropper.

With fuzzy querying, even for the same query, the query actually being sent out is different every time. Each ciphertext in the ciphertext space has the equal probability to be selected as the extra data thus the real boundaries of queries will not be disclosed in our non-deterministic scheme. In addition, because a random number of irrelevant data are included in each query result, the added noise also helps in hiding the access pattern and the re-construction attack will be more difficult to be conducted to our scheme.

V. Security Analysis

Boldyreva et al's scheme [12] was designed to meet POPF-CCA that an adversary has a negligible advantage distinguishing between the scheme and a pseudorandom order-preserving function under chosen-ciphertext attacks. We develop our scheme based on Boldyreva's but with additional security features and thus we expect our scheme can achieve stronger security guarantee. Nevertheless, it will be better to start with reviewing Boldyrea et al's security proofs.

One important theorem from Boldyreva et al. is a new security notion POPF-CCA. Given an OPE algorithm and an *pseudorandom order-preserving function*, the adversary has no advantage in distinguishing the oracle access to the OPE algorithm and the access to a random function. The OPE scheme in their paper was proved to be POPF-CCA. The definition of POPF-CCA is given in [12] and quoted below.

POPF-CCA. Given an OPE scheme $\mathcal{SE} = (\mathcal{K}, Enc, Dec)$. For an adversary A against $\mathcal{SE}$, the *pseudorandom order-preserving function advantage under chosen-ciphertext attacks* (POPF-CCA) against $\mathcal{SE}$ is

$$\mathbf{Adv}^{popf\text{-}cca}_{\mathcal{SE}}(A) = \Pr_{K \xleftarrow{\$} \mathcal{K}}\left[\mathbf{A}^{\mathcal{E}nc(K,\cdot),\mathcal{D}ec(K,\cdot)} = 1\right] - \Pr_{g \xleftarrow{\$} OPF_{\mathcal{D},\mathcal{R}}}\left[\mathbf{A}^{g(\cdot),g^{-1}(\cdot)} = 1\right]$$

where $OPF_{\mathcal{D},\mathcal{R}}$ stands for the set of all order-preserving functions from plaintext space $\mathcal{D}$ to ciphertext space $\mathcal{R}$ where $\mathcal{D} \leq \mathcal{R}$. $\mathcal{SE}$ is POPF-CCA secure if for any adversary this advantage is negligible.

As Algorithm 1 shows, our algorithm uses binary search for encryption. In each round, the algorithm finds a plaintext-ciphertext mapping. If the plaintext-ciphertext mapping found is not for the input plaintext, then the algorithm moves to the left/right half for the next binary search until it finds the mapping for the input. As a result, if we are able to show each round of our algorithm is POPF-CCA secure, then our scheme with multiple such rounds is also POPF-CCA secure.

Theorem 1. *The proposed OPE scheme described in Section IV is at least POPF-CCA secure, even when the data owner has no pre-knowledge about the data and choose a* ***pseudorandom density function*** *$\mathcal{F}$ as input.*

Proof. Let T be the plaintext domain for proposed algorithm, D and R be the Domain and Range for the OPE^{HG} algorithm from Boldyreva et. al [12], respectively, where $|T| < |D| < |R|$. Our proposed encryption algorithm takes an input from T and outputs a ciphertext in the range R. There are two mappings in each encryption round, where the OPE^{HG} maps between R and D first, and then the function $GetGMM()$ maps between D and T. It has been proved in [12] that OPE^{HG} mapping between R and D is POPF-CCA secure. If the $GetGMM()$ is also POPF-CCA secure between D and T, then our proposed scheme is POPF-CCA secure between R and T.

In our algorithm, we choose $\mathcal{F}$ as a pseudorandom function and $\mathcal{F}(t) > 0$ for $t \in \{0, |T|\}$. Given an input x (i.e., number of marked balls picked), $GetGMM()$ computes an output z (i.e., index of last box used when all x marked balls filled), where $x = \int_{min(T)}^{z} \mathcal{F}$. When $\mathcal{F}$ is pseudorandom and $\mathcal{F}(t) > 0$, it is clear that the mapping $x = \int_{min(T)}^{z} \mathcal{F}$ is a pseudorandom (and thus POPF-CCA secure) order-preserving function between space D and T. Since OPE^{HG} is a POPF-CCA secure mapping between R and D and now $GetGMM()$ is also POPF-CCA secure between D and T, our scheme is POPF-CCA secure between R and T.

Though our algorithm meets POPF-CCA, we want a security notion stronger than POPF-CCA that can describe our scheme better. In [13], Boldyreva et al. proposed a security notion IND-CCPA that weakens IND-OCPA by requiring the adversary to choose the challenge message spaces before key generation. The definition of *IND-CCPA* is quoted below.

IND-CCPA. Let $\mathcal{LR}(\cdot,\cdot,b)$ be the function that takes two message vectors m_0 and m_1 as inputs and returns m_b, where $m_0 = (m_0^1, m_0^2 \ldots, m_0^l)$ and $m_1 = (m_1^1, m_1^2 \ldots, m_1^l)$. $\mathcal{SE} = (\mathcal{K}, Enc, Dec)$ is an order-preserving encryption scheme on message space M. For an adversary A against $\mathcal{SE}$ with (m_0, m_1) and $b \in \{0,1\}$, where $|m_0| = |m_1| = l$, and also m_0, m_1 have the same *order pattern*, i.e., $\forall 1 \leq i,j \leq l$, $m_0^i < m_0^j$ and $m_1^i < m_1^j$. Consider the following experiment (σ denotes a state the adversary can preserve):

$\mathbf{Experiment}\ \mathbf{Exp}^{ind-ccpa-b}_{\mathcal{SE}}(A)$

$(m_0, m_1, \sigma) \xleftarrow{\$} A; \quad K \xleftarrow{\$} K(m_b)$

$c_j \leftarrow \mathcal{E}nc(K, m_b^j)\ for\ j = 1, \ldots, l$

$d \xleftarrow{\$} A(\sigma, c_1, c2, \ldots, c_l)$, where $d \in \{0,1\}$

$Return\ d$

For each query (m_0, m_1), an adversary A's *ind-ccpa advantage* $\mathcal{SE}$ is

$$\mathbf{Adv}^{ind\text{-}ccpa}_{\mathcal{SE}}(A) = Pr\left[\mathbf{Exp}^{ind\text{-}ccpa\text{-}1}_{\mathcal{SE}}(A) = 1\right] - Pr\left[\mathbf{Exp}^{ind\text{-}ccpa\text{-}0}_{\mathcal{SE}}(A) = 1\right]$$

We say $\mathcal{SE}$ is *indistinguishable under committed chosen plaintext attacks* (IND-CCPA-secure) if the ind-ccpa advantage against $\mathcal{SE}$ is negligible.

In [13] where IND-CCPA was proposed, it expects the key generation algorithm takes the appropriate message space from plaintext as input. In our scheme, the key generation generates a density function which can be derived from the plaintext. We consider the prerequisites are similar and they both require the datasets to be known before key generation phase. Theorem 2 below proves our proposed scheme is IND-CCPA secure.

Theorem 2. *The proposed OPE scheme can achieve IND-CCPA secure if the data is pre-known and static.*

Proof. According to the definition in Section V, IND-CCPA allows the challenger to choose the encryption key after receiving the message vectors from the adversary.

To prove IND-CCPA security, all we need to show is that, given any two ordered vector $m_0 = (m_0^1, m_0^2, \ldots, m_0^l)$ and $m_1 = (m_1^1, m_1^2, \ldots, m_1^l)$, the challenger can always choose a vector $m_i, i \in \{0,1\}$, and construct a density function $\mathcal{F}$ such that m_i can be encrypted to a fixed ordered ciphertext vector, for example $(\frac{R}{l+1}, \frac{2R}{l+1}, \cdots, \frac{lR}{l+1})$, where R is the ciphertext space and l is the length of m_i. With the fixed ciphertext vector been returned, the adversary definitely has no advantages to guess which vector m_i has been encrypted.

There are two different mappings in our algorithm, which are the mapping OPE^{HG} between R and D and the mapping $GetGMM()$ between D and T. Let the function $Dec()$ be the OPE^{HG} decryption mapping from R to D. Let $m_i = (m_i^1, m_i^2, \ldots, m_i^l)$, which is an increasing vector for $i \in \{0,1\}$, be the plaintext vector in T, and let $r = (\frac{R}{l+1}, \frac{2R}{l+1}, \cdots, \frac{lR}{l+1})$ be the fixed ordered ciphertext vector in R. Thus $d = (Dec(\frac{R}{l+1}), Dec(\frac{2R}{l+1}), \cdots, Dec(\frac{lR}{l+1}))$ is the fixed vector in D. For both m_i with $i \in \{0,1\}$, if we can show that

(I) the challenger can always construct an $\mathcal{F}$ so that $GetGMM()$ can map d to m_i, and

(II) our algorithm, based on constructed $GetGMM()$, will encrypt m_i to the fixed ciphertext r,

then our algorithm is IND-CCPA secure.

To show (I), let $\mathcal{F}$ be the density function so $\mathcal{F}(t) > 0$ for $t \in \{0, |T|\}$. The challenger can construct such a density function $\mathcal{F}(t)$ and a monotonically increasing function $u(z) = \int_0^z \mathcal{F}(t)$ as below:

$$\mathcal{F}(t) = \begin{cases} \frac{Dec(\frac{R}{l+1})}{m_i^1}, \text{ if } 0 < t \le m_i^1 \\ \frac{Dec(\frac{2R}{l+1}) - Dec(\frac{R}{l+1})}{m_i^2 - m_i^1}, \text{ if } m_i^1 < t \le m_i^2 \\ \cdots, \\ \frac{Dec(\frac{lR}{l+1}) - Dec(\frac{(l-1)R}{l+1})}{m_i^l - m_i^{l-1}} \text{ if } m_i^{l-1} < t \le m_i^l \end{cases}$$

$$u(z) = \begin{cases} \mathcal{F}(m_i^1) \cdot z, \text{ if } 0 < z \le m_i^1 \\ \mathcal{F}(m_i^2) \cdot (z - m_i^1) + u(m_i^1), \text{ if } m_i^1 \le z < m_i^2 \\ \cdots, \\ \mathcal{F}(m_i^l) \cdot (z - m_i^{l-1}) + u(m_i^{l-1}), \\ \text{ if } m_i^{l-1} \le z < m_i^l \end{cases}$$

For any m_i^j that $i \in \{0,1\}$ and $j \in \{1, 2, \ldots, l\}$, based on the functions $\mathcal{F}(t)$ and $u(z)$ above, we have $u(m_i^j) = Dec(\frac{jR}{l+1})$ is always true. In our algorithm, the function $GetGMM()$ is the inverse function of $u()$ and can be writen as $GetGMM(u(z)) = z$. That is, with such $u()$, given any $Dec(\frac{jR}{l+1}) \in d$, the function $GetGMM(Dec(\frac{jR}{l+1})) = GetGMM(u(m_i^j)) = m_i^j \in m_i$ is always true. This concludes the $GetGMM()$ can map the vector d to the vector m_i.

Secondly, we need to show (II) is also true. To encrypt a plaintext m_i^j for $i \in \{0,1\}$ and $j \in \{1, 2, \ldots, l\}$, starting from the mid range gap in R, our algorithm uses binary search that first maps the range gap in R to a value in D via OPE^{HG}, and then maps the value in D to a value in T via $GetGMM()$. The value in T will be compared to m_i^j to determine the next round of binary search. Since we have already shown in (I) that our constructed $GetGMM()$ maps $Dec(\frac{jR}{l+1})$ to m_i^j, and the OPE^{HG} maps $\frac{jR}{l+1}$ to $Dec(\frac{jR}{l+1})$, the binary search in our algorithm eventually will find the value $\frac{jR}{l+1}$ in R (and thus the value $Dec(\frac{jR}{l+1})$ in D) that maps to m_i^j in T. When m_i^j is found by the binary search, the encryption process ends and the ciphertext of m_i^j is $\frac{jR}{l+1}$ in R. This implies our algorithm can always encrypt both plaintext vectors $m_i = (m_i^1, m_i^2, \ldots, m_i^l)$, for $i = \{0,1\}$, to a fixed ciphertext vector $r = (\frac{R}{l+1}, \frac{2R}{l+1}, \cdots, \frac{lR}{l+1})$. Thus, our scheme is IND-CCPA secure.

1) Discussion: The benefits of building such function $GetGMM()$ are bountiful. First, it prevents adversaries from establishing accurate estimate for any ciphertext. Previous stateless schemes leak extra information about the relative distances of plaintexts by just observing their ciphertexts. As a result, the value of plaintext can be estimated by the position of its ciphertext in the ciphertext range.

Secondly, the data distribution can be hidden. With most of the previous stateless OPE schemes, the data distribution of plaintext dataset is also exposed due to the fact that, after encryption, the dense parts stay dense and sparse parts remain sparse. Thus, some sensitive statistic information can be derived from the data distribution disclosures. For example, in healthcare data, an adversary could learn in which age group most patients are located. Our strategy of building a good $GetGMM()$ is to make the data distribution flat for OPE ciphertext no matter what the input plaintext distribution is so that adversaries couldn't learn any information about the original data distribution.

To make the ciphertext flat, the dataset must be known

before encryption. If data is unavailable before encryption, historical datasets or sample datasets that shares common similarities can also help. For the worst scenario where the data owner knows nothing about their data, our solution may still make an adversary harder to retrieve statistical information from the encrypted dataset. By choosing a random $GetGMM()$, the distribution of encrypted data likely will be very different from the original plaintext distribution. Thus, it becomes more difficult to speculate the original data and thus the privacy and confidentiality of data can be better preserved.

VI. Similarity Comparison for Data Distribution

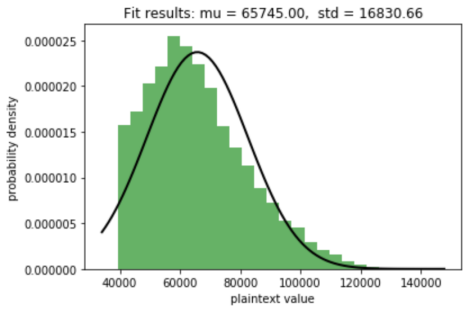
We found an OPE implementation from CryptDB [9], which implements Boldyreva et al's algorithm. We modified their source codes to satisfy our requirement. Their OPE implementation cannot decrypt arbitrary ciphertexts. Let n_1 and n_2 be the ciphertexts of m and $m+1$ (where $n_1 < n_2$). Their implementation can decrypt n_1 and n_2, but cannot decrypt the values between n_1 and n_2. We modified their source codes so that any number in the ciphertext space can be decrypted to its corresponding plaintext based on the position of gaps.

We use an online employee database [28] as sample data and encrypt the salary data in the database using three OPE algorithms: Boldyreva et al's algorithm, our flattened OPE algorithm (dataset is pre-known before encryption) and our randomized OPE algorithm (dataset is not known before encryption). We will then compare the similarity between the plaintext dataset against each encrypted version to see if the encrypted data preserves the data distribution (and thus reveals statistical information) from its plaintext counterpart. Before the similarity comparison, we have validated our OPE algorithms by checking whether all the encryption, decryption and range-query work correctly on the encrypted database.

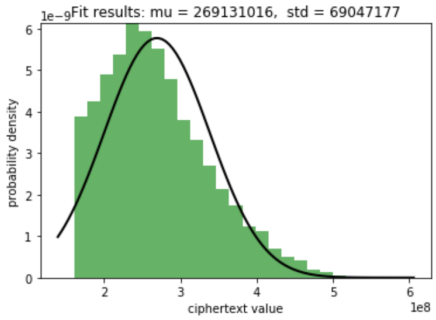
For the similarity comparison, we chose *Kolmogorov–Smirnov test* (K-S test) [27] to evaluate the similarity among different encrypted datasets. The test returns two values: *statistic* and *p-value*. The *statistic* is the absolute max distance between two Cumulative Distribution Functions (CDFs). The closer this number to zero, it is more likely the samples are from the same distribution. *p-value* is used to reject the null hypothesis that two samples are from the same distribution. In K-S test, the null hypothesis states that the distributions are the same. The lower *p-value* means the greater evidence that we can reject the null hypothesis.

Original plaintexts. Figure 2a shows the plaintext distribution of salary data. One Gaussian distribution is enough to describe the distribution approximately, where the mean μ is 65745 and the standard deviation σ is 16830.

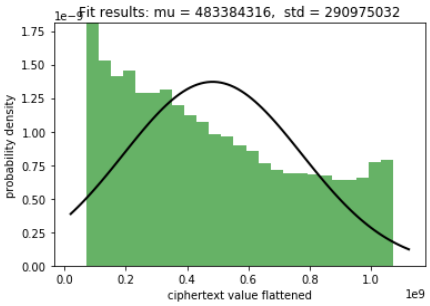
Encrypted by Boldyreva et al's OPE algorithm. For this OPE encryption, we set the plaintext space to 16 bits and the ciphertext space to 30 bits. Figure 2b shows the distribution of this encrypted database. The encrypted data and the original plaintexts have a great similarity. We also use normal distribution to fit this encrypted data and its mean μ is 269121016 and standard deviation σ is 69047177.



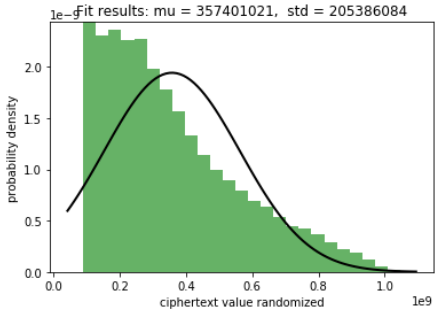
(a) Salary in plaintext



(b) Salary encrypted with Boldyreva et. al.'s algorithm



(c) Salary encrypted with our OPE algorithm: flattened



(d) Salary encrypted with our OPE algorithm: randomized

Fig. 2: Data distribution between plaintext and different ciphertexts

Encrypted by our flattened OPE algorithm. Since the sample dataset distribution has already been studied, we set the data distribution key as ($\sigma = 18000$, $\mu = 66000$) which is close to the original data distribution. The data distribution af-

TABLE I: The similarity comparison using K-S 2-sample test.

Data	Null Hypothesis	Test Statistic	p-value	Null Hypothesis Conclusion
Boldyreva et al's algorithm	Same	0.024788	3.39419	Accept
Proposed (flattened)	Same	0.376774	0.0	Decline
Proposed (randomized)	Same	0.356119	0.0	Decline

ter this encryption is shown in Figure 2c. The data distribution of this encrypted version is much more flattened compared to the ones in the plaintexts and in the Boldyreva et al's version. In Figure 2c, the encrypted data is not fully flattened because in Figure 2a the normal distribution used is not able to describe the data distribution 100% accurately. Even so, based on the K-S 2-Sample test results shown in Table I, the distributions of the plaintexts and the ciphertexts we encrypted using the flattened OPE are distinct-different.

Encrypted by our randomized OPE algorithm. Assume a DB manager who doesn't have the dataset before encryption but he makes a reasonable assumption that most people are middle class and salary are around $\$80,000$ with the standard deviation $\sigma = 30000$ (68% of people are within the range \$50,000-\$110,000). The distribution of this encrypted dataset is shown in Figure 2d.

To have a general and objective comparison, we conducted the K-S 2-Sample test against each plaintext and encrypted pair after normalization. The outcomes are shown in Table I. The results indicate that Boldyreva et al's scheme has the same distribution pattern as the plaintext, while the ciphertext encrypted by our schemes are considered to have different distributions to the plaintext.

VII. Conclusion

We proposed an IND-CCPA secure and practical stateless OPE scheme, where the position of each ciphertext is not only decided by the randomness of the HGD function, but also the data distribution. As a result, the estimation exposure attack can be prevented and the ciphertexts will not reveal the data distribution of the plaintexts. In addition, we proposed two techniques and showed that many real-world attacks can be alleviated by some small changes. The first technique allows any deterministic OPE scheme becoming non-deterministic. The second technique "fuzzy querying" hides the access patterns of outsourced databases from adversaries. As a result, most real-world attacks to encrypted databases, such as *sorting attacks*, *frequency attacks*, and *reconstruction attacks* (a popular attack in the research community recently), can be prevented or mitigated. Besides proposing an OPE scheme, we showed that our scheme is at least POPF-CCA secure. If the dataset is known before encryption, we also showed our scheme is IND-CCPA secure. In the end, we conducted empirical experiments to demonstrate the security improvement for our schemes, comparing to the previous work.

References

[1] Amanatidis, G., Boldyreva, A., O'Neill, A.: Provably-secure schemes for basic query support in outsourced databases. DBSec, 4602 of LNCS, pp. 14–30. Springer, 2007.

[2] Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. CRYPTO, 4622 of LNCS, pp. 535–552. Springer, 2007.

[3] Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. CRYPTO, 5157 of LNCS, pp. 335–359, Springer, 2008.

[4] Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. ACM SIGMOD, 2004.

[5] Ge, T., Zdonik, S. B.: Fast, secure encryption for indexing in a column-oriented DBMS. ICDE, 2017

[6] Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. ACM SIGMOD, 2002.

[7] Kadhem, H., Amagasa, T., Kitagawa, H.: A secure and efficient order preserving encryption scheme for relational database. Int. Conf. on Knowledge Management and Information Sharing, 2010.

[8] Liu, D., Wang, S.: Programmable order-preserving secure index for encrypted database query. IEEE Int. Conf. on Cloud Computing, 2012.

[9] Popa, R. A., Redfield, C. M. S., Zeldovich, N., Balakrishnan, H.: CryptDB: Protecting confidentiality with encrypted query processing. ACM SOSP, 2011.

[10] Ang, G.W., Woelfel, J.H., Woloszyn, T. P.: System and method of sort-order preserving tokenization. US Patent Application 12/450, 890, 2012.

[11] Popa, R. A., Li, F. H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding IEEE S&P, 2013.

[12] Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric Encryption. EUROCRYPT, 2009

[13] Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. CRYPTO, 2011

[14] Xiao, L., Yen, I.-L., Huynh, D. T.: A note for the ideal order-preserving encryption object and generalized order-preserving encryption. Cryptology ePrint Archive, Report 2012/350, 2012.

[15] Kerschbaum, F.: Frequency-hiding order-preserving Encryption. ACM CCS, 2015.

[16] Kerschbaum F., Schröpfer, A.: Optimal average-complexity ideal-security order-preserving encryption. ACM CCS, 2014.

[17] Kadhem, H., Amagasa, T., Kitagawa, H.: A secure and efficient order preserving encryption scheme for relational database. Int. Conf. on Knowledge Management and Information Sharing, 2010.

[18] Wozniak, S., Rossberg, M., Grau, S., Alshawish, A., Schaefer, G.: Beyond the ideal object: Towards disclosure-resilient order-preserving encryption schemes. ACM Cloud Computing Security Workshop, CCSW, 2013.

[19] Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: Generic attacks on secure outsourced database. ACM CCS, 2016.

[20] Naveed, M., Kamara, S., Wright, C. V.: Inference attacks on property-preserving encrypted databases. ACM CCS, 2015.

[21] Lacharité, M., Minaud, B., Paterson, K.G.: M. Lacharité, B. Minaud, K.G. Paterson Improved reconstruction attacks on encrypted data using range query leakage. IEEE S&P, pp. 297–314, 2018

[22] Islam, M. S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. NDSS, 2012.

[23] Islam, M. S., Kuzu, M., Kantarcioglu, M.: Inference attack against encrypted range queries on outsourced database. CODASPY, 2014.

[24] Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. ACM CCS, 2015.

[25] https://scikit-learn.org/stable/modules/mixture.html

[26] https://brilliant.org/wiki/gaussian-mixture-model

[27] https://en.wikipedia.org/wiki/Kolmogorov-Smirnov_test

[28] test_db: A sample MySQL database with an integrated test suite, used to test your applications and database servers. https://github.com/datacharmer/test_db