EncodeORE: Reducing Leakage and Preserving Practicality in Order-Revealing Encryption

Zheli Liu, Siyi Lv, Jin Li*, Yanyu Huang, Liang Guo, Yali Yuan and Changyu Dong

Abstract—Order-preserving encryption (OPE) is a cryptographic primitive that preserves the order of plaintexts. In the past few years, many OPE schemes were proposed to solve the problem of executing range queries in encrypted databases. However, OPE leaks some certain information (for example, the order of ciphertext), so it is vulnerable to many attacks. Subsequently, order-revealing encryption (ORE) was proposed by Boneh et al. (Eurocrypt 2015) as a generalization of order-preserving encryption. It breaks through the limitation of the numeric order of OPE plaintext. It implements ciphertext comparison for any specific form of plaintext through a publicly computable comparison function. In this work, we aim to design a new ORE scheme which reduces the leakages and preserves the practicality in terms of ciphertext length and encryption time. We first propose the hybrid model named *HybridORE*. Then, we propose an improved scheme named *EncodeORE* which achieves acceptable security and appropriate ciphertext length. They both explore the encode strategy of encoding plaintext into different parts and apply suitable ORE algorithms to each part according to its security characteristics to reduce leakages. Compared with the typical CLWW scheme (FSE 2016) and Lewi-Wu (CCS 2016) in large domain, they have fewer leakages. The experiment shows that the proposed *EncodeORE* is very practical.

Index Terms—Encrypted database, data privacy, order-preserving encryption, order-revealing encryption.

1 Introduction

Nowadays, the increasing amount of data makes companies and governments unable to bear the storage of all data on their own. They are more willing to store their private data in remote and potentially untrusted servers. In order to maintain the security of data, encrypted databases have become an effective way to solve the problems of big data storage and privacy protection, but this sacrifices availability of the data. Once the data is stored in the server in ciphertext, it is difficult to query the data without decrypting it. For supporting various queries in the encrypted database, many works [1–7] have been proposed. Order-preserving encryption (OPE) proposed by Agrawal et al. [1] can be directly deployed into encrypted databases to support range queries. It is a cryptographic primitive that ensures that the order of the ciphertext is consistent with the corresponding order of the plaintext.

The research of OPE has received widespread attention in the field of database and security. Boldyreva et al. [2] proposed the first formal security definition of OPE scheme. They introduced the concept of "best possible" semantic security, which stated that ciphertext did not reveal any information outside of the plaintext order. Furthermore, this paper defined the indistinguishability under an ordered chosen-plaintext attack (IND-OCPA) and a weaker notion of security (POPF-CCA security). However, Boldyreva et al. [2] pointed out that if the scheme is *stateless* and *immutable*, the ideal functionality cannot be achievable. Therefore, in order to achieve IND-OCPA security, many researchers have designed some stateful and immutable schemes, like Popa et al. [8], Kerschbaum et al. [9], and so on. However, the client storage and heavy frequent interactions make these schemes not practical.

Notice that OPE scheme still faces attacks [10–12] even if it reaches IND-OCPA security. Especially, Naveed et al. [10] have proposed a series of inference attacks against OPE. These attacks can successfully recover almost all of the plaintexts while only providing data dumps of encrypted databases and auxiliary information from public databases. Based on the existence of the above attacks and security considerations, Kerschbaum [13] proposed an OPE scheme (FH-OPE) that can hide the frequency of plaintext. Certainly, it can resist some inference attacks which are based on frequency. However, it still faces some sort based attack and the client storage is heavy. Furthermore, data-driven cyber security [14–17] is also the focus of protection.

1.1 Order-Revealing Encryption

In order to achieve higher security and improve practicality, Boneh et al. [18] proposed a more flexible concept called order-revealing encryption (ORE). ORE can be seen as a generalization of OPE. ORE achieves higher security by allowing some unimportant leakages. It is generally believed that these leakages will not cause a large security loss. Furthermore, ORE does not limit the ciphertext to any particular forms. That is, the ciphertext in ORE is not necessarily numerical as OPE scheme. The order of ciphertext

- *Z. Liu, S. Lv and Y. Huang are with the College of Cyber Science and the College of Computer Science, Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, China, 300071. Email: liuzheli@nankai.edu.cn, lv_si_yi@163.com, onlyerir@163.com.*
- *J. Li is with the School of Computer Science, Guangzhou University, China. E-mail: lijin@gzhu.edu.cn.*
- *L. Guo is in Huawei Technologies Co., Ltd. Email: blue.guo@huawei.com.*
- *Y. Yuan is with the institute of Computer Science, University of Goettingen, 37077 Goettingen, Germany. Emails: yali.yuan@cs.uni-goettingen.de.*
- *C. Dong is with School of Computing, Newcastle University, Newcastle upon Tyne, U.K. Email: changyu.dong@ncl.ac.uk.*

Manuscript received May 2, 2020; revised xxx 2020.

TABLE 1: Comparison with prior ORE schemes. Let n be the size of input in bits where l_1 is the size of value part and l_2 is the size of range part. Let d denote the number of bits per block, h denote the size of property-preserving hash, λ denote the size of PRF output size, and MSDB denote most significant differing bit. PPH is property-preserving hash, PRF is pseudorandom function, and PRP is pseudorandom permutation.

Scheme	Ciphertext length (bit)	Primitive usage		Leakage
		Encryption	Comparison	
CLWW [21]	$2n$	n PRF	none	MSDB
LEWI-WU-NORMAL [22]	$\frac{n}{d}(\lambda + n + 2^{d+1}) + \lambda$	$\frac{2n}{d}$ PRP, $2\frac{n}{d}(2^d + 1)$ PRF, $\frac{n}{d}2^d$ Hash	$\frac{n}{2d}$ Hash	The first differing block
CLOZ [24]	$n \cdot h$	n PRF, n PPH, 1 PRP	n^2 PPH	MSDB equality pattern
ENCODEORE	$2(l_1 + l_2)$	n PRF	none	MSDB of one part

in ORE is determined by a publicly computable comparison function. Lots of ORE schemes have been proposed, such as Wang et al. [19], Cash et al. [20], and so on. Like Cash et al. [20], most ORE schemes use complex cryptographic primitives (property preserving Hash, PPH) to reduce leakage. However, compared with the schemes that only use symmetric encryption primitives, the scheme of Cash et al. [20] is not practical in actual scenarios. Therefore, ORE scheme still faces the tradeoff between practicality and leakage.

The schemes proposed by Chenette et al. [21] and Lewi et al. [22] are the two most famous ORE schemes. In 2019, the evaluations in [23] pointed out that Chenette et al. [21] (CLWW) is the most efficient ORE scheme. However, it reveals the most significant differing bit between two ciphertexts. To further reduce the leakage, Lewi et al. [22] (Lewi-Wu) proposed two ORE schemes. One is for small plaintext domain (Lewi-Wu-Small) and the other is for large plaintext domain (Lewi-Wu-Normal). Lewi-Wu-Small [22] only leaks the order of plaintext. However, its ciphertext length increases exponentially with the number of plaintext domain. On the basis of CLWW [21], Lewi-Wu-Normal [22] reduces the leakage to the location of the first differing block. Unfortunately, [23] pointed out that the performance of Lewi-Wu-Normal [22] decreases exponentially with the increase of block size, which is mainly due to the exponential decrease in the number of indexes performed by pseudorandom function (PRF) and pseudorandom permutation (PRP).

Motivation. We are willing to further reduce leakage based on Lewi-Wu-Normal [22] and achieve the same performance as CLWW [21]. Therefore, we are committed to proposing a new ORE scheme which draws on the strengths of CLWW [21] and Lewi-Wu-Small [22]. We try to combine the advantages of these two schemes to make a good balance between leakage and practicality. In detail, our motivations are 1) presenting a new ORE scheme which achieves acceptable efficiency, 2) achieving less leakage than Lewi-Wu-Normal [22] and CLWW [21].

1.2 Our Contributions

In this paper, we propose a new ORE model which can reduce leakage and preserve practicality based on Lewi-Wu-Normal [22] and CLWW [21]. Based on the new ORE model, we propose a new ORE scheme which reduces the length of the ciphertext. Table 1 shows that our scheme is more efficient than Lewi-Wu-Normal [22] and CLOZ [24] in ciphertext length and primitive usage. In particular, CLOZ uses the property-preserving hash primitive to obtain smoothed CLWW leakage. However, CLOZ has many limitations that make it difficult to be applied in practice. Its limitations are reflected in the square number of calls to PPH and PPH itself. The order of magnitude is more expensive than any primitive in other schemes. Therefore, we only make a rough comparison in the Table 1, and will not elaborate on CLOZ scheme and make detailed comparisons in the security analysis and performance. We summarize the contributions in detail as follows.

Hybrid model and encode strategy. In this work, we present a basic hybrid model named *HybridORE* to reduce leakage. Its core idea is to encode the plaintext into the range part and the value part. The range part indicates the range of plaintext and it can be mapped to a small domain. The value part represents the relevant value and it can be expressed in binary form. These two parts are encrypted and compared by Lewi-Wu-Small [22] and CLWW [21] respectively. As we know, the comparison in the range part using Lewi-Wu-Small [22] does not leak additional information. Meanwhile, because the value part must be processed by alignment and other methods to meet the comparison requirements, the encoded value is no longer equal to the original value. Thus, it achieves the goal that has fewer leakages than CLWW [21] and Lewi-Wu-Normal [22].

The improved ORE construction with short ciphertext. In order to avoid ciphertext expansion caused by Lewi-Wu-Small [22] in *HybridORE* and preserve the same level of leakage, we propose a new scheme named *EncodeORE*. It follows the hybrid model *HybridORE* and achieves stronger security than CLWW [21]. In order to have short ciphertex length, it uses CLWW [21] to encrypt two parts together. Furthermore, as shown in Table 1, it has fewer leakages than CLWW [21] and Lewi-Wu-Normal [22]. Thus it is a practical ORE construction with small leakage and short ciphertext. In particular, with the encode strategy, *EncodeORE* is more efficient when the length of the plaintext is different.

Security comparison with other schemes. In terms of security, *HybridORE* and *EncodeORE* leak less private information than CLWW [21] and Lewi-Wu-Normal [22]. As mentioned before, CLWW [21] leaks the first differing bit between two different ciphertexts and Lewi-Wu-Normal [22] reduces the leakage to the location of the first differing block between two different ciphertexts. *HybridORE* and *EncodeORE* only leak the first differing bit of one part but not

the whole plaintext in any case. In particular, when comparing two plaintexts with different range part, *EncodeORE* only leaks the first differing bit of range part and *HybirdORE* has no additional leakage. Furthermore, because we encode the value part of plaintext, when the value part is different, our schemes leak less information than CLWW [21] and Lewi-Wu-Normal [22].

Experimental evaluation. We implement experiments with *EncodeORE* for integer domain and run the existing codes of CLWW [21] and Lewi-Wu-Normal [22]. The length of the plaintext we set varies from 8 to 64 bits. The experimental results show that when the plaintext size is 8 bits and the block size of Lewi-Wu-Normal [22] is 2 bits, the ciphertext length of Lewi-Wu-Normal [22] is the shortest. At this time, the ciphertext length of Lewi-Wu-Normal [22] is 22 times that of *EncodeORE*. When the plaintext size is 64 bits and the block size of Lewi-Wu-Normal [22] is 16 bits, the ciphertext length of Lewi-Wu-Normal [22] is the longest. At this time, the ciphertext length of Lewi-Wu-Normal [22] is about 1800 times that of *EncodeORE*. Furthermore, the ciphertext length of *EncodeORE* is only two bytes longer than CLWW [21]. The encryption time of *EncodeORE* is about $0.9\ \mu s$ longer than CLWW [21] on average but much shorter than Lewi-Wu-Normal [22]. In terms of comparison execution time, it is related to the generation of random numbers. *EncodeORE* is much efficient when the range part of ciphertexts is different.

1.3 Related work

In this section, we describe the existing work on searching over encrypted data, including order-revealing encryption, order-preserving encryption, and other related work.

Order-preserving encryption. OPE was first proposed by Agrawal et al. [1] in 2004. Later, researchers [3, 4, 8, 9, 13, 25–28] have conducted many expansive studies on it. Some works [4, 28] are devoted to studying the security of order-preserving encryption. In recent years, many works have focused on ad hoc OPE schemes [29, 30] which generally lack formal security analysis. Stateful or interactive OPE schemes [3, 8, 9, 13, 27] avoid the lower bounds in [2, 4, 8], however, it is usually impractical in distributed storage environments. In 2013, Popa et al. [8] constructed the first order-preserving encryption scheme that can achieve ideal security, and give a strict security proof. Considering that the ciphertext database can be updated, Popa et al. [8] presented a security definition: same-time OPE (st-OPE) security which is stronger than ideal security.

Order-revealing encryption. In 2014, [31] regarded ORE as a special case of multi-input functional encryption (MIFE). In the subsequent works, ORE schemes were constructed with various techniques, such as indistinguishability obfuscation [31–35], cryptographic multilinear maps [18], symmetric or public key encryption [29, 36], bilinear mapping [37], and branch functions [38].

The security of ORE is also the focus of research. In 2016, CLWW [21] introduced the general ORE security definition. In detail, CLWW [21] leveraged leakage function to constrain the security of the scheme. Even it was the most practical ORE scheme, the leakage is proved to be able to be further restrained. In 2016, Lewi-Wu [22] proposed two schemes that satisfied ORE security definition introduced by CLWW [21] with the leakage of the order of plaintext or the location of the first differing block. However, the performance of Lewi-Wu [22] decreased exponentially as the block size increased and Lewi-Wu [22] would take a lot of time due to the use of hash primitives. Subsequent works [19, 24, 39, 40] had conducted many studies on other information leakages that compromise the confidentiality and privacy of the entire data. In 2017, Haagh et al. [40] generalized the cryptographic notion of ORE to arbitrary functions and did not rely on cryptographic obfuscation or multilinear map. However, it only allowed to determine the (partial) ordering of two vectors. In 2018, Wang et al. [19] further studied the leakage of OPE and ORE and their forward security. They proposed a practical compilation framework for achieving forward secure ORE in order to resist the perniciousness of file injection attack (FIA). Later, Cash et al. [24] used bilinear maps for a new primitive called property-preserving hash (PPH) to construct a new ORE scheme (CLOZ), which could hide the first differing bit in 2018. However, the complexity and impracticality of the PPH primitives made it impractical.

Other works on searching over encrypted data. Searchable symmetric encryption (SSE) [41–44] and property-preserving encryption (PPE) [25, 45] are proposed to support query function on encryption databases. In addition, fully homomorphic encryption (FHE) [46], hidden vector encryption [47], and oblivious RAMs (ORAM) [48] are also designed to solve various functional problems in encrypted databases.

2 PRELIMINARIES

2.1 NOTATIONS

Let $[n]$ denote the set of integers $\{1, \ldots, n\}$ for $n \in \mathbb{N}$. In this paper, $[N]$ denotes the plaintext domain, $[E]$ denote the range domain and $[V]$ denote the value domain. Let λ denote the security parameter. If $f = o(1/\lambda^c)$ for all $c \in N$, we can say that function $f(\lambda)$ is negligible. Let negl(λ) denote the negligible function in λ. Let $x \leftarrow \mathcal{D}$ denote a draw from a distribution $\mathcal{D}$. We review the standard definition of pseudorandom functions (PRFs) [49]. For a function $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$, if there is no efficient adversary can distinguish with the outputs of $F(k, \cdot)$ for a randomly chosen $k \xleftarrow{R} \mathcal{K}$ from that of a truly random function $f(\cdot)$ from $\mathcal{X}$ to $\mathcal{Y}$, we can say that F is a secure PRF.

2.2 ORE DEFINITIONS

An order-revealing encryption (ORE) scheme can be denoted as $\prod$ = (ORE.Setup, ORE.Encrypt, ORE.Compare) which consists of three algorithms. The algorithms of ORE are defined over an ordered domain $\mathcal{D}$. The algorithms are described as follows:

- ORE.Setup(1^λ)$\rightarrow$sk. The input is the security parameter λ. The output is the private key which is used for encryption.
- ORE.Encrypt(sk, m)$\rightarrow$ct. The inputs are private key sk and plaintext m. The output is the ciphertext ct.
- ORE.Compare($\mathrm{ct}_1, \mathrm{ct}_2$)$\rightarrow$ b. The inputs are two ciphertexts ct_1 and ct_2. The output is a bit $b \in \{0, 1\}$.

Correctness. For the security parameter λ, if there is an ORE scheme $\prod$ = (ORE.Setup, ORE.Encrypt, ORE.Compare) which is defined over an ordered domain $\mathcal{D}$ and $\mathrm{sk} \leftarrow$ ORE.Setup(1^λ), then for any messages $m_1, m_2 \in \mathcal{D}$,

$$\begin{aligned} &\Pr[\text{ORE.Compare}(\mathrm{ct}_1, \mathrm{ct}_2) \\ &= \mathbf{1}\ (m_1 > m_2)] \\ &= 1 - \mathrm{negl}(\lambda), \end{aligned} \tag{1}$$

where ct_1, ct_2 denote the ciphertexts and ct_1 = ORE.Encrypt(sk, m_1), ct_2 = ORE.Encrypt(sk, m_2).

Security. The ideal security goal for OPE is indistinguishability under ordered chosen-plaintext attack (IND-OCPA) as shown in which means ciphertexts only reveal the order of the plaintexts, but nothing else. Naveed et al. [10] proposed that the security proposed by OPE may be insufficient. Some adversary can use frequency analysis and sorting of ciphertexts to decrypt OPE encrypted values. In order to defend against such attacks, Kerschbaum [13] proposed a stronger notion of security IND-FAOCPA as shown in which hides the frequency of OPE values. For order-revealing encryption, the ''best-possible'' security which is shown in Remark 1 is the notion of indistinguishability under ordered chosen-plaintext attack (IND-OCPA).

Definition 1. (**ORE with Leakage [21]**). ORE scheme can be denoted as $\prod$ =(ORE.Setup, ORE.Encrypt, ORE.Compare). Let $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_q)$ denote a polynomial-size adversary and $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_q)$ denote a polynomial-size simulator. Let $\mathcal{L}$ denote the leakage function, sk denote the private key, m_i denote the plaintext, ct_i denote the ciphertext and st denote the state. If for $\mathcal{A}$, there exists $\mathcal{S}$ such that the outputs of the two distributions $\mathrm{REAL}_{\mathcal{A}}^{\mathsf{ORE}}(\lambda)$ and $\mathrm{SIM}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\mathsf{ORE}}(\lambda)$ are computationally indistinguishable, ORE scheme $\prod$ is secure with the leakage function $\mathcal{L}$. The experiments of $\mathrm{REAL}_{\mathcal{A}}^{\mathsf{ORE}}(\lambda)$ and $\mathrm{SIM}_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\mathsf{ORE}}(\lambda)$ are shown in Fig. 1.

Best-possible security [22]. When the leakage function is only the order of the plaintexts, we can say that the scheme achieves the best-possible security. It can be formally described as follows:

Remark 1. Iff ORE scheme achieves the best-possible security, the leakage function can be denoted as

$$\begin{aligned} \mathcal{L}_{\mathrm{CMP}}^{\mathsf{ORE}}(m_1, \ldots, m_t) = &\{(i, j, \mathrm{CMP}(m_i, m_j) | \\ &1 \le i < j \le t)\}, \end{aligned} \tag{2}$$

where m_i $(1 \le i \le t)$ denotes the plaintext. Furthermore, $\mathrm{CMP}(\cdot,\cdot)$ is a comparison function. If $m_i > m_j$, $\mathrm{CMP}(m_i, m_j)$ returns 1. If $m_i < m_j$, $\mathrm{CMP}(m_i, m_j)$ returns -1. If $m_i = m_j$, $\mathrm{CMP}(m_i, m_j)$ returns 0.

2.3 CLWW Scheme

The basic idea of CLWW [21] is implementing ORE from pseudorandom functions and comparing the ciphertexts bit by bit. The setup algorithm mainly does some preparation work to generate a private key sk for pseudorandom function.

As shown in Fig. 2, during the encryption process, the plaintext is represented in binary form, and then encrypted. The binary representation of the plaintext m is $b_1 b_2 \ldots b_n$. For each bit b_i $(1 \le i \le n)$, computing

$$u_i = F(\mathrm{sk}, (i, b_1 b_2 \ldots b_{i-1} || 0^{n-i})) + b_i \ (\mathrm{mod}\ M), \tag{3}$$

where the F denotes the pseudorandom function and $M \ge 3$. After encrypting the plaintext bit by bit, the ciphertext of m is $(u_1, u_2, \ldots, u_n)$, where $u_i \in \mathbb{Z}_M$.

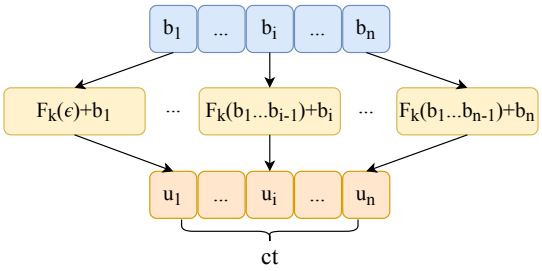


Fig. 2: Encryption algorithm of CLWW [21].

The comparison process is shown in Fig. 3. For two different plaintexts, same prefix will have same ciphertext block. If the prefix is different, the value is computationally hidden. When comparing two ciphertexts $\mathrm{ct}_1 = (u_1, u_2, \ldots, u_i, \ldots)$ and $\mathrm{ct}_2 = (u_1, u_2, \ldots, u'_i, \ldots)$, compare u_i with u'_i $(1 \le i \le n)$ and i is the smallest index where $u_i \ne u'_i$.

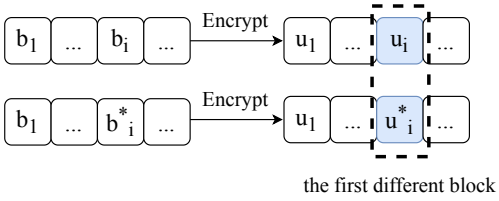


Fig. 3: Comparison of CLWW [21].

CLWW [21] can be extended to d-ary generation, but with a slight loss in security. Different from above, $m = b_1 b_2 \ldots b_n$ is the d-ary form representation of plaintext m. Furthermore, if the input length of d-ary CLWW [21] is l, then the ciphertext size is approximately $l \cdot log_d^M$, for any $M \ge 2d - 1$.

Leakage. The comparison operation of CLWW [21] reveals the order of the plaintexts and the first differing bit of the plaintexts. The leakage function can be written as follows:

$$\begin{aligned} \mathcal{L}(m_1, \ldots, m_t) := &\{(\mathrm{ind}_{diff}(m_i, m_j), 1(m_i < m_j)) : \\ &1 \le i < j \le t\}, \end{aligned} \tag{4}$$

where m_i denotes the plaintext. Denote $m = b_1 b_2 \ldots b_n$, $m' = b'_1 b'_2 \ldots b'_n$, $\mathrm{ind}_{diff}(m, m')$ gives the index of the first bit where m and m' is different. If $m \ne m'$, $\mathrm{ind}_{diff}(m, m')$ equals the smallest index i $(1 \le i \le n)$ for which $b_i \ne b'_i$. If $m = m'$, set the $\mathrm{ind}_{diff}(m, m') = n + 1$.

2.4 Lewi-Wu Scheme

In this section, we introduce small domain scheme Lewi-Wu-Small and large doamin scheme Lewi-Wu-Normal in [22].

$\underline{\mathrm{REAL}_{\mathcal{A}}^{ORE}(\lambda):}$
1: $\mathrm{sk} \leftarrow \mathrm{ORE.Setup}(1^\lambda)$
2: $(m_1, \mathrm{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(1^\lambda)$
3: $\mathrm{ct}_1 \leftarrow \mathrm{ORE.Encrypt}(\mathrm{sk}, m_1)$
4: **For** $2 \le i \le q$
5: $(B_i, \mathrm{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_i(\mathrm{st}_{\mathcal{A}}, \mathrm{ct}_1, \ldots, \mathrm{ct}_{i-1})$
6: $\mathrm{ct}_i \leftarrow \mathrm{ORE.Encrypt}(\mathrm{sk}, m_i)$
7: Output $(\mathrm{ct}_1, \ldots, \mathrm{ct}_q)$ and $\mathrm{st}_{\mathcal{A}}$

$\underline{SIM_{\mathcal{A},\mathcal{S},\mathcal{L}}^{\mathrm{ORE}}(\lambda):}$
1: $\mathrm{st}_{\mathcal{S}} \leftarrow \mathcal{S}_0(1^\lambda)$
2: $(m_1, \mathrm{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$
3: $(\mathrm{ct}_1, \mathrm{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_1(\mathrm{st}_{\mathcal{S}}, \mathcal{L}(m_1))$
4: **For** $2 \le i \le q$:
5: $(m_i, \mathrm{st}_{\mathcal{A}}) \leftarrow \mathcal{A}_i(\mathrm{st}_{\mathcal{A}}, \mathrm{ct}_1, \ldots, \mathrm{ct}_{i-1})$
6: $(\mathrm{ct}_i, \mathrm{st}_{\mathcal{S}}) \leftarrow \mathcal{S}_i(\mathrm{st}_{\mathcal{S}}, \mathcal{L}(m_1, \ldots, m_i))$
7: Output $(\mathrm{ct}_1, \ldots, \mathrm{ct}_q)$ and $\mathrm{st}_{\mathcal{A}}$

Fig. 1: ORE ideal-real security game.

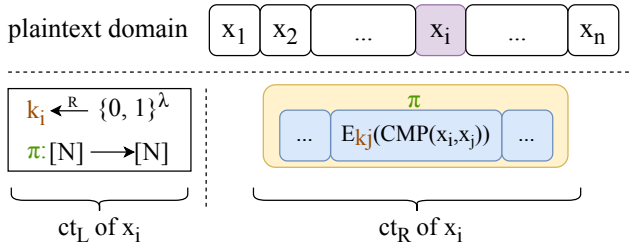


Fig. 4: Encryption of Lewi-Wu-Small [22].

2.4.1 Lewi-Wu-Small

The plaintext space of Lewi-Wu-Small [22] is $[N]$ where $N = poly(\lambda)$. In Fig. 4, the ciphertext is composed of the left ciphertext and the right ciphertext. The left ciphertext ct_L of a value x_i $(1 \le i \le n)$ stores the encryption key k_i which is used for the right ciphertext. The right ciphertext ct_R of value x_i stores the encryption of the comparison outputs between x_i and every other elements in the domain. The encryption of $\mathrm{CMP}(x_i, x_j)$ $(1 \le j \le n)$ is $E_{k_j}(\mathrm{CMP}(x_i, x_j))$. Then all the comparison components are permuted according to π. Furthermore, the left ciphertext of x_i includes this permuted position $\pi(x_i)$.

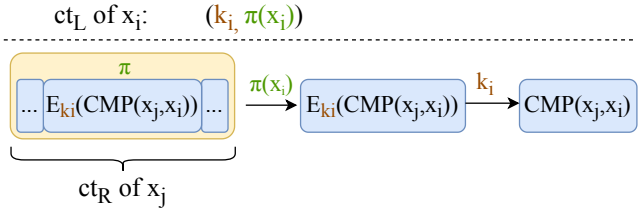


Fig. 5: Encryption of Lewi-Wu-Small [22].

As shown in Fig. 5, when getting the left ciphertext ct_L of x_i and the right ciphertext ct_R of x_j, we can compare the two elements x_i and x_j. With $\pi(x_i)$ which is in x_i's left ciphertext ct_L, we can get $E_{k_i}(\mathrm{CMP}(x_j, x_i))$ which is in x_j's rigth ciphertext ct_R and denotes the encryption of $\mathrm{CMP}(x_j, x_i)$. Then with k_i which is in x_i's left ciphertext ct_L, we can get $\mathrm{CMP}(x_j, x_i)$ which contains the order of x_i and x_j.

Leakage. The comparison operation of Lewi-Wu-Small [22] does not reveal other information except the order of the plaintexts. It achieves the best-possible security from Remark 1. The leakage function can be denoted as follows:

$$\mathcal{L}^{\mathrm{ORE}}(m_1, \ldots, m_t) = \{(i, j, \mathrm{CMP}(m_i, m_j) | 1 \le i < j \le t)\}. \quad (5)$$

2.4.2 Lewi-Wu-Normal

The large domain construction can be seen as consisting of small domain construction and CLWW [21] construction. CLWW [21] can be seen as a general transformation that takes the k-bit ORE scheme as input and outputs the kn-bit ORE scheme. Therefore, it can be seen as inputting a 1-bit ORE scheme and extends it to a n-bit ORE scheme. It implements ciphertext expansion, but the security is slightly reduced. The large domain construction applies the domain-extension technique to the small domain construction and expands the d-bit ORE to dn-bit ORE scheme. It regards d-bit as a block.

Leakage. Compared to Lewi-Wu-Small [22], the index of the first block that differs between two ciphertexts is the additional leakage of Lewi-Wu-Normal [22]. For two plaintexts m and m', the d-ary representations of them are $m = b_1 b_2 \ldots b_n$ and $m' = b'_1 b'_2 \ldots b'_n$ respectively. $\mathrm{ind}_{diff}^{(d)}(m, m')$ denotes the first differing block function. $\mathrm{ind}_{diff}^{(d)}(m, m')$ is the first index i $(1 \le i \le n)$ such that $b_i \ne b'_i$ and $b_j = b'_j$ for all $j < i$. If $m = m'$, $\mathrm{ind}_{diff}^{(d)}(m, m')$ equals $n+1$. The leakage function can be denoted as follows:

$$\mathcal{L}_{\mathrm{BLK}}^{(d)}(m_1, \ldots, m_t) = \{(i, j, \mathrm{BLK}(m_i, m_j)) | 1 \le i < j \le t\}, \quad (6)$$

where m_i denotes the plaintext and BLK function can be denoted as $\mathrm{BLK}(m_i, m_j) = (\mathrm{CMP}(m_i, m_j), \mathrm{ind}_{diff}^{(d)}(m_i, m_j))$.

3 Hybrid Model for Reducing Leakage

In this section, we introduce our basic solution for reducing leakage and preserving practicality.

3.1 Hybrid model

Our basic solution is inspired by two well-known ORE schemes. One is CLWW scheme [21], which is the most practical ORE scheme. It provides a universal bit-by-bit encryption solution. It has short ciphertext and high efficiency, but it has more leakage. The other is Lewi-Wu-Small [22], which is the ideal solution for comparison operations in a small domain. Although it has ciphertext expansion, it does not reveal any other information except the order. Therefore, we consider how to combine these two schemes together to achieve the purpose of both reducing leakage and preserving practicality. An intuitive idea is to split the plaintext into two parts. The left part is mapped to a small domain to indicate the range of plaintext; the right part is

***HybridORE*.Setup**(λ) :
Client:
1: $k_1, k_2 \xleftarrow{\$} \{0,1\}^\lambda$
2: $\pi : [E] \to [E]$
3: $sk_1 = (k_1, \pi)$
4: $sk_2 = k_2$
5: **Return** $sk = (sk_1, sk_2)$

***HybridORE*.Encrypt**(sk_1, sk_2, m) :
Client:
1: $(E, V) \leftarrow$ *HybridORE*.Encode(m)//$V = v_1 v_2 \dots v_{l_1}$
2: $k_l \leftarrow F(k_1, \pi(E))$
3: $ct_E^L = (k_L, \pi(E))$
4: $r \xleftarrow{\$} \{0,1\}^\lambda$
5: **For** $i \in [E]$:
6: $w_i = \mathsf{CMP}(\pi^{-1}(i), E) + H(F(k_1, i), r) \pmod 3$
7: $ct_E^R = (r, w_1, w_2, \dots, w_E)$
8: **For** $i = 1$ *to* l_1 :
9: $u_i = F(sk_2, (i, v_1 v_2 \dots v_{i-1} || 0^{l_1 - i})) + v_i$
10: $ct_V = (u_1, u_2, \dots, u_{l_1})$
11: Send $ct = (ct_E^R, ct_V)$ to Server

***HybridORE*.Compare**(x, ct, ct') :
Client:
1: $h \leftarrow \pi(x)$
2: $k \leftarrow F(k_1, h)$
Server:
3: $r \leftarrow ct_E'^R.r$
4: **If** $ct_E'^R.w_h - H(k, r) \pmod 3 \neq 0$:
5: **Return** $ct_E^R.w_h - H(k, r) \pmod 3$
6: **For** $0 \le i \le l_1$:
7: **If** $u_i \neq u_i' \pmod 3$ and $u_i' = u_i + 1 \pmod 3$
8: **Return** 1
9: **Return** 0

***HybridORE*.Encode**(m) :
Client:
1: Denote m as $V \times d^E$
2: $V = v_1 \dots v_j$ //The binary representation of m.
3: **If** $j < l_1$:
4: $V = v_1 v_2 \dots v_j || 0^{l_1 - j}$
5: **Return** (E, V)

Fig. 6: The algorithms of *HybridORE*.

used to represent the relevant value. Then, we apply Lewi-Wu-Small [22] to the left part while CLWW [21] to the right part. Since different parts use different encryption methods, we call this model the hybrid model.

In order to describe the model better, we define the process of dividing the message into two parts as the *encode* process. Therefore, the entire encryption process in our hybrid model includes two processes, namely *encode* and *encryption*. The model is shown in Fig. 7. For convenience, we define the left part and the right part after encoding as the range part and the value part. When performing the comparison operation, the ciphertexts of the range part will be compared first. If they are not equal, the result of the range part comparison is the final result. Because this part uses an ideal ORE scheme (Lewi-Wu-Small [22]), there is no additional leakage. If they are equal, the ciphertexts of the value part will be compared. Usually, the value part will be processed by alignment and other methods to meet the new comparison requirements, so that the encoded value is no longer equal to the original value. Therefore, the comparison in this part also has fewer leakages.

Formal definition. We provide the formal definition of the basic hybrid model as follows:

- *HybridORE*.Setup(1^λ)$\to$sk: The algorithm samples the necessary private key. The input is the secure parameter. The output is the private key sk.
- *HybridORE*.Encode(m)$\to(E, V)$: The algorithm encodes the plaintext into the range part and the value part. The input is the plaintext m. The output is the coding of m which consists of the range part E and the value part V.
- *HybridORE*.Encrypt(sk, m)$\to$ct: The algorithm first calls the encode algorithm to get the encoding of plaintext m. Then the range part E and the value part V are encryptd with private key sk. The input is the plaintext m. The output is the ciphertext ct which consists of the ciphertext of the range part and value part.
- *HybridORE*.Compare(ct, ct′)$\to b$: The algorithm first compares the range part. If the range part is different, there is no necessary to compare the value part. Otherwise, it compares the value part. The input is two ciphertexts ct and ct′ and the output is a bit $b \in \{0, 1\}$.

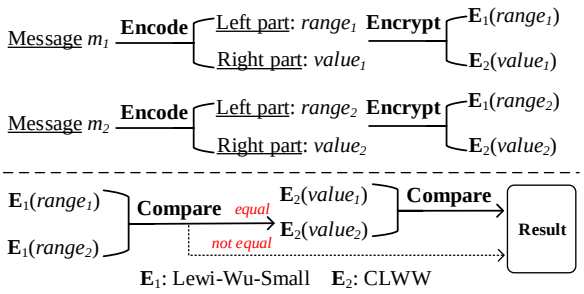


Fig. 7: The Hybrid Model for Reducing Leakage.

3.2 Challenges and our *HybridORE* solution

Our hybrid model looks very effective, but it is not easy to build an ORE scheme that meets the hybrid model. There are two challenges that must be solved. The first challenge is *how to encode the plaintext to the range part and the value part*. The second challenge is *how to compare the ciphertexts of the range part and the value part*. In Fig. 6, we present *HybridORE* scheme to demonstrate how it works. Note that when $x \neq y$, CMP(x,y) provides the same amount as **1**, therefore the correctness holds.

For the first challenge, we must ensure that the range part after encoding has as few values as possible. Because this part will be encrypted by Lewi-Wu-Small [22] scheme,

the length of the ciphertext will increase linearly with the number of the values. Meanwhile, the value part is no longer the original value, but it should be able to be compared. In *HybridORE*, we can select the representation of floating-point numbers in computer systems as the encoding method to meet this requirement. For a 32-bit floating-point number, it uses 1 bit to represent the sign, 8 bits to represent the exponent, and 23 bits to represent the value. That is, the first 9 bits represent the range part, and the last 23 bits represent the value part.

For the second challenge, we can divide it into two categories. One is *how to compare one by one*, and the other is *how to compare step by step*. The former follows the basic idea of the hybrid model and our *HybridORE* demonstrates its implementation. The latter puts forward higher security requirement. It requires that the ciphertexts of the value part cannot be directly compared, and the result of the range part comparison must be used as input. That is, we must conduct the connection between the ciphertext of the range part and the ciphertext of the value part. However, this kind of connection is difficult to establish directly on the two existing solutions. Because the ciphertext of Lewi-Wu-Small [22] is the encryption of comparison with all different values, it is difficult to use the output of these comparisons as the input of the ciphertext of value part. Considering Lewi-Wu-Small [22] produces left ciphertext and right ciphertext, a simple solution is to generate the left ciphertext for the ciphertext of the value part. Assume that the server is honest but curious, it will request the left ciphertext of the value part when the ciphertexts of the range part are equal. Therefore, this solution increases the interaction between the client and the server, and brings a lot of storage for the client. We introduce an improved construction to achieve ciphertexts comparison without interaction in the hybrid model in Section 4.

3.3 Analysis of *HybridORE*

Space Usage. For Lewi-Wu-Small [22], the ciphertext can be divided into the left ciphertext and the right ciphertext. The left ciphertext consists of a key for pseudorandom function and a permutation. The length of the left ciphertext is $\lambda + \lceil \log N \rceil$, where the λ denotes the security parameter and $[N]$ is the plaintext domain. The right ciphertext consists of a nonce and N elements in $\mathbb{Z}_3$. The length of the right ciphertext is $\lambda + \lceil Nlog_2^3 \rceil$. Therefore, the ciphertext of Lewi-Wu-Small [22] is $\lambda + \lceil \log N \rceil + \lceil Nlog_2^3 \rceil$. For CLWW [21], the ciphertext length is $\lceil n \cdot log_2^3 \rceil$, where n is the bit length of input. For our *HybridORE* model, suppose the value domain is $[V]$, the length of it is l_1, the domain of range part is $[E]$ and there are E elements. Because the range part is encrypted by Lewi-Wu-Small [22] and the value part is encrypted by CLWW [21]. Therefore, the ciphertext length of *HybridORE* is:

$$\lambda + \lceil \log E \rceil + \lceil Elog_2^3 \rceil + \lceil l_1 \cdot log_2^3 \rceil. \tag{7}$$

Leakage analysis. *HybridORE* leaks much less than Lewi-Wu-Normal [22] and CLWW [21]. The leakage can be denoted as follows:

- If the range part is the same but the value part is different, the leakage function can be denoted as

$$\mathcal{L}_{\mathrm{ORE}} = \text{the first differing bit of value part.} \tag{8}$$

- If the range part is different, the leakage function can be denoted as

$$\mathcal{L}_{\mathrm{ORE}} = \perp. \tag{9}$$

The comparison of the range part leaks nothing while the value part has not been compared. Therefore, the scheme leaks nothing.

4 EncodeORE: The Improved Construction

In this section, we propose an improved ORE construction called *EncodeORE*. It preserves the advantages of *HybridORE* and further shortens the length of the ciphertext.

4.1 Basic idea

HybridORE looks ideal, but there are still ciphertext extensions (the range part). Notice that *encode* is an important factor to reduce leakage in *HybridORE*, it changes the comparison operation to the range comparison and value comparison. Because of the ciphertext expansion, Lewi-Wu-Small [22] is not available. It means that the leakage will be increased, but our improved construction is still better than Lewi-Wu-Normal [22] scheme.

For the challenge of encoding, we adopt scientific notation instead of the representation of floating-point numbers in our *EncodeORE*. We know that scientific notation can represent a message m as $V \times d^E$, where d is equal to 10, V is the value part and E is the range part. It is more flexible because we can choose different cardinal number d according to different requirements. In order to allow the value part to be compared bit by bit, we need to fill the end of the value part with 0 to the same length.

For the challenge of *comparison step by step*, we encrypt the range part and value part through CLWW [21] scheme together. In this way, the connection between the two parts can be established without interaction. This improvement mainly stems from the *encode* strategy, so we call our scheme *EncodeORE*.

4.2 Concrete Construction

Fix a security parameter $\lambda \in N$, and let F be a secure PRF. Set d as the cardinal number of scientific notation, l_1 as the bit length of value part and l_2 as the bit length of range part. We define our *EncodeORE* scheme as $\prod_{EORE}$ = (*EncodeORE*.Setup, *EncodeORE*.Encode, *EncodeORE*.Encrypt, *EncodeORE*.Compare) and give the construction in the form of independent algorithm description.

Algorithm 1 ***EncodeORE*.Setup**(1^λ)

1: $\mathrm{sk} \xleftarrow{\$} \{0,1\}^\lambda$
2: **Return** sk

EncodeORE.Setup(1^λ). The algorithm chooses a uniformly random PRF key sk for F. Since both the range part and the value part are encrypted together, only one private key sk is generated.

Algorithm 2 *EncodeORE*.**Encode**(m)

Denote m as $V \times d^E$
$V = v_1 v_2 \dots v_i$ //The binary representation of m.
$E = e_1 e_2 \dots e_j$ //The binary representation of E.
If $i < l_1$
  $V = v_1 v_2 \dots v_i || 0^{l_1 - i}$
If $j < l_2$
  $E = 0^{l_2 - j} || e_1 e_2 \dots e_j$
$l = l_1 + l_2$
$B = E || V = b_1 b_2 \dots b_l$
Return B

EncodeORE.Encode(m). The algorithm adopts scientific notation to encode the plaintext m into range part E and value part V. The value part $V = v_1 v_2 \dots v_i$ is the binary representation of plaintext m. The range part $E = e_1 e_2 \dots e_j$ is the binary representation of exponential part. Padding the value part and the range part to the fixed length l_1, l_2 respectively. Therefore, the value part V is expressed as $v_1 v_2 \dots v_{l_1}$ and the range part E is expressed as $e_1 e_2 \dots e_{l_2}$. We encrypt the range part and value part together, so stitch them together and denote as B. The length of B is $l = l_1 + l_2$.

Algorithm 3 *EncodeORE*.**Encrypt**(sk, m)

$B \leftarrow$ *EncodeORE*.Encode(m) $//B = b_1 b_2 \dots b_l$
For $1 \le i \le l$:
  $mask = F(\mathrm{sk}, (i, b_1 b_2 \dots b_{i-1} || 0^{l-i}))$
  $u_i = mask + b_i \pmod 3$
$ct = (u_1, u_2, \dots, u_l)$
Return ct

EncodeORE.Encrypt(sk, m). The plaintext m is encoded as B which is the result of splicing the range part and value part together. When encrypting the i-th bit of B, generate a pseudorandom number as $mask$. The pseudorandom number $mask$ is generated by the pseudorandom function with the first $i-1$ elements as input. Then after adding the i-th bit with the $mask$, modulo 3 with the result to get the ciphertext for the i-th bit. In this way, we get the ciphertext which can be denoted as $\mathrm{ct} = (u_1, u_2, \dots, u_l)$.

Algorithm 4 *EncodeORE*.**Compare**$(\mathrm{ct}, \mathrm{ct}')$

For $1 \le i \le l$:
  If $u_i \neq u'_i$ *and* $u'_i = u_i + 1 \pmod 3$:
    Return 1
Return 0

EncodeORE.Compare$(\mathrm{ct}, \mathrm{ct}')$. When performing comparison algorithm, the range part is compared first then the value part is compared. Only when the range part is the same, compare the value part. Since encrypting the range part and the value part together, no other interaction between the server and the client is required.

4.3 Analysis of EncodeORE

Space Usage. For CLWW [21], the ciphertext length is $\lceil n \cdot log_2^3 \rceil$, where n denotes the length of input. For *EncodeORE* scheme, the length of value part is l_1, the length of range part is l_2 and $l = l_1 + l_2$. The ciphertext length of value part is $\lceil l_1 \cdot log_2^3 \rceil$, the ciphertext length of range part is $\lceil l_2 \cdot log_2^3 \rceil$. Therefore, the ciphertext length of *EncodeORE* is $\lceil l \cdot log_2^3 \rceil$. The ciphertext of range part is much shorter than *HybridORE* model and the ciphertext length of value part is the same. Therefore, the ciphertext length of *EncodeORE* is shorter than *HybridORE*.

Leakage analysis. We compare the leakage with CLWW [21] and Lewi-Wu-Normal [22]. Lewi-Wu-Small [22] is not suitable for large domain applications, because the length of the ciphertext is positively related to the length of the data domain, so it is not considered. CLWW [21] leaks the first unequal binary bit. Lewi-Wu-Normal [22] leaks the location of the first differing block. The leakage further leads to the disclosure of the range of the relative distance between two data. The leakage of our model is shown as following:

- **Same range, different value.** The scheme leaks the first differing bit of value part. Since the range part has not been leaked, the adversary cannot estimate the relative distance between the two plaintexts.
- **Different range.** The value part has not been compared. The comparison of the range part leaks the first differing bit of the range part.

The leakage of *EncodeORE* scheme consists of the leakage of Setup, Encode and Encrypt which can be denoted as $\mathcal{L}_{\mathsf{Setup,Encode,Encrypt}}$ and the leakage of Compare which can be denoted as $\mathcal{L}_{\mathsf{Cmp}}$. Our scheme applies CLWW [21] to encrypt the encoding of plaintext. The leakage of comparison algorithm is the same as CLWW [21]. It can be denoted as:

$$\mathcal{L}_{\mathsf{Cmp}}(B_1, \dots, B_t) := \{(\mathrm{ind}_{\mathit{diff}}(B_i, B_j), 1(m_i < m_j)) : 1 \le i < j < t\}, \tag{10}$$

where B_i denotes the encoding of the plaintext m_i, $\mathrm{ind}_{\mathit{diff}}(x, y)$ gives the index of the first bit where x and y is different.

The Setup, Encode and Encrypt algorithms are done by the client side. Therefore, the leakage of setup and encode algorithm can be denoted as:

$$\mathcal{L}_{\mathsf{Setup,\ Encode,\ Encrypt}}(m_1, \dots, m_t) = \perp . \tag{11}$$

In summary, we can say the leakage of our scheme can be denoted as:

$$\mathcal{L}_{\mathsf{EORE}}(m_1, \dots, m_t) := \{(\mathrm{ind}_{\mathit{diff}}(B_i, B_j), 1(m_i < m_j)) : 1 \le i < j \le t\}, \tag{12}$$

where B_i is the encoding of plaintext m_i and B_j is the encoding of plaintext m_j.

4.4 Security

We prove that *EncodeORE* is secure with the leakage $\mathcal{L}_{\mathsf{EORE}}$.

Theorem 1. As shown in Definition 1, *EncodeORE* scheme is secure with leakage function $\mathcal{L}_{\mathsf{EORE}}$ under the PRF security of F.

Proof 1. Let λ denote the security parameter. As shown in Definition 1, let $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_q)$ denote an efficient adversary for ORE security game, where $q = poly(\lambda)$. Let $\mathcal{S} = (\mathcal{S}_0, \dots, \mathcal{S}_q)$ denote an efficient simulator for

which the outputs of distributions $\mathrm{REAL}_{\mathcal{A}}^{\mathrm{EORE}}(\lambda)$ and $\mathrm{SIM}_{\mathcal{A},\mathcal{S},\mathcal{L}_{\mathrm{EORE}}}^{\mathrm{EORE}}(\lambda)$ are computationally indistinguishable.
Game G_0. This is the real world game $\mathrm{REAL}_{\mathcal{A}}^{\mathrm{EORE}}(\lambda)$. That is to say

$$\mathsf{Pr}[\mathrm{REAL}_{\mathcal{A}}^{\mathrm{EORE}}(\lambda)=1]=\mathsf{Pr}[G_0=1] \tag{13}$$

Game G_1. The output of G_1 is $(ct_1,\ldots,ct_q)$. In setup algorithm, choose a random function

$$f \xleftarrow{R} \mathrm{Funs}[([n]\times\{0,1\}^{n-1}),\mathbb{Z}_M]. \tag{14}$$

In $EncodeORE.\mathsf{Encrypt}$, replace $F(k,\cdot)$ with $f(\cdot)$. Others are the same as G_0. Therefore, under the PRF security of F, G_0 and G_1 are computationally indistinguishable. Hence, we have

$$|\mathsf{Pr}[G_1=1]-\mathsf{Pr}[G_0=1]|=0. \tag{15}$$

Simulator. $\mathcal{S}=(\mathcal{S}_0,\ldots,\mathcal{S}_q)$ denotes the simulator. At first, $\mathcal{S}_0$ initializes an empty table $\mathtt{L}:[q]\times[n]\to\mathbb{Z}_M$ and outputs that $\mathrm{st}_{\mathcal{S}}=\mathtt{L}$. Then, for each $1\le i\le q$, the simulation algorithm takes inputs $\mathrm{st}_{\mathcal{S}}$ and $\mathcal{L}_{EORE}(m_1,\ldots,m_i)$ where B_i is the encoding of m_i and B_i is the query which is the output of adversary $\mathcal{A}$. $\mathcal{L}_{\mathrm{EORE}}(m_1,\ldots,m_i)$ contains $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)$ and the values *1* for $B_j<B_i$. $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)$ gives the index of the first bit where B_j and B_i are different. The output of simulator is $(\overline{\mathrm{ct}}_1,\ldots,\overline{\mathrm{ct}}_q)$.
For each $1\le s\le l$ (l denotes the encoding length of plaintext), if for $1\le j\le i-1$, there is $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)>s$, then the simulator sets $\overline{u}_{i,s}=\mathtt{L}(j,s)=\overline{u}_{j,s}$, where j is the smallest one for which $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)>s$. If for each $1\le h\le i-1$, there is $\mathrm{ind}_{\mathit{diff}}(B_h,B_i)\le s$, then the simulator sets

$$\begin{aligned}\overline{u}_{i,s}&=\mathtt{L}(j,s)-(1-2\cdot 1\,(B_j<B_i))\\&=\overline{u}_{j,s}-(1-2\cdot 1\,(B_j<B_i))\ (\mathrm{mod}\ M),\end{aligned} \tag{16}$$

where $j\in[i-1]$ is the smallest one for which $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)=s$. If for each $1\le h\le i-1$, there is $\mathrm{ind}_{\mathit{diff}}(B_h,B_i)<s$, then the simulator sets $\overline{u}_{i,s}=y$, where $y\xleftarrow{R}\mathbb{Z}_M$.
The simulator adds the mapping $(i,s)\to\overline{u}_{t,s}$ to $\mathtt{L}$. The simulator $\mathcal{S}_t$ outputs the ciphertext $\overline{\mathrm{ct}}_i$ and updates $\mathrm{st}_{\mathcal{S}}=\mathtt{L}$.
We proceed inductively in the number of queries q. For some $1\le i\le q$, suppose $(\mathrm{ct}_1,\ldots,\mathrm{ct}_i)\equiv(\overline{\mathrm{ct}}_1,\ldots,\overline{\mathrm{ct}}_i)$. The statement holds for $i+1$. Denote the ciphertext ct_j as $(u_{j,1},\ldots,u_{j,n})$ for $1\le j\le i$. Denote the ciphertext $\overline{\mathrm{ct}}_j$ as $(\overline{u}_{j,1},\ldots,\overline{u}_{j,n})$ for $1\le j\le i$. Denote the s-th bit of B_j as $b_{j,s}$.
If for a $1\le j\le i-1$, there is $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)>s$ and j is the smallest one for which $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)>s$. B_j and B_i has the same prefix of at least length s. Then, in the G_1, we have

$$u_{i,s}=f(s,p||0^{n-s})+b_{i,s}=u_{j,s}, \tag{17}$$

where $p\in\{0,1\}^{s-1}$ denotes the common prefix. Therefore, by inductive hypothesis, we can conclude that $\overline{u}_{i,s}$ and $u_{i,s}$ are identically distributed. If for each $1\le h\le i-1$, there is $\mathrm{ind}_{\mathit{diff}}(B_h,B_i)\le s$ and $1\le j\le i-1$ is the smallest one for which $\mathrm{ind}_{\mathit{diff}}(B_j,B_i)=s$. In this situation, B_j and B_i has the same prefix of at least length s. Then, in the G_1, we have

$$u_{i,s}=f(s,p||0^{n-s})+b_{i,s}\ (\mathrm{mod}\ M), \tag{18}$$

$$u_{j,s}=f(s,p||0^{n-s})+b_{j,s}\ (\mathrm{mod}\ M). \tag{19}$$

Assume $b_{j,s}\neq b_{i,s}$, so $b_{i,s}=b_{j,s}-(1-2\cdot\mathbf{1}(B_j<B_i))$. Therefore, in G_1, we have

$$\begin{aligned}u_{i,s}&=f(s,p||0^{n-s})+b_{i,s}\\&=u_{j,s}-(1-2\cdot\mathbf{1}(B_j<B_i))\ (\mathrm{mod}\ M).\end{aligned} \tag{20}$$

Therefore, by inductive hypothesis, we can conclude that $\overline{u}_{i,s}$ and $u_{i,s}$ are identically distributed. If for each $1\le h\le i-1$, there is $\mathrm{ind}_{\mathit{diff}}(B_h,B_i)<s$. By assumption, the encodings $B_1,\ldots,B_{i-1}$ do not have prefix p and f is a random function. Therefore, $u_{i,s}$ is independent of all other ciphertexts and is uniform in $\mathbb{Z}_M$. By inductive hypothesis, we can conclude that $\overline{u}_{i,s}$ and $u_{i,s}$ are identically distributed.
We can conclude that $u_{i,s}\equiv\overline{u}_{i,s}$ for all $1\le s\le n$. Hence, we have

$$|\mathsf{Pr}[G_1=1]-\mathsf{Pr}[\mathrm{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{L}_{\mathrm{EORE}}}^{\mathrm{EORE}}(\lambda)=1]|=0. \tag{21}$$

Conclusion. By combining all the contribution from all games, there exists an adversary $\mathcal{A}$ such that

$$|\mathsf{Pr}[\mathrm{REAL}_{\mathcal{A}}^{\mathrm{EORE}}(\lambda)=1]-\mathsf{Pr}[\mathrm{IDEAL}_{\mathcal{A},\mathcal{S},\mathcal{L}_{\mathrm{EORE}}}^{\mathrm{EORE}}(\lambda)=1]|=0. \tag{22}$$

5 Experiments and Evaluation

In this section, we present a benchmark of *EncodeORE* and run CLWW [21], Lewi-Wu-Normal [22] schemes to present the performance comparison. Furthermore, we measure the performance of *EncodeORE* under various parameter settings.

5.1 Experiment Details

We implement *EncodeORE* and run CLWW [21], Lewi-Wu-Normal [22] scheme according to the code in https://github.com/kevinlewi/fastore in the experimental evaluation. We focus on comparing the ciphertext length, encryption execution time, and comparison execution time of these three schemes. All experiments run single-threaded on the processor. The cost of network transmission and delay time is ignored. Our code is run on a computer device running Ubuntu 18.04 with an Intel(R) Core(TM) i5 − 8250U CPU @1.60 GHz 1.80 GHz CPU and 8 GB memory. Our experiment is entirely written in C. The length of the plaintext we set varies from 8 to 64 bits. When encoding the plaintext, the cardinality is 16. We operate under 128-bit security ($\lambda=128$) and we use AES-128 to instantiate PRF. We use OpenSSL [50] to implement the AES-128 algorithm in our experiments. At the same time, we use the GMP [51] library to implement large integer algorithms.

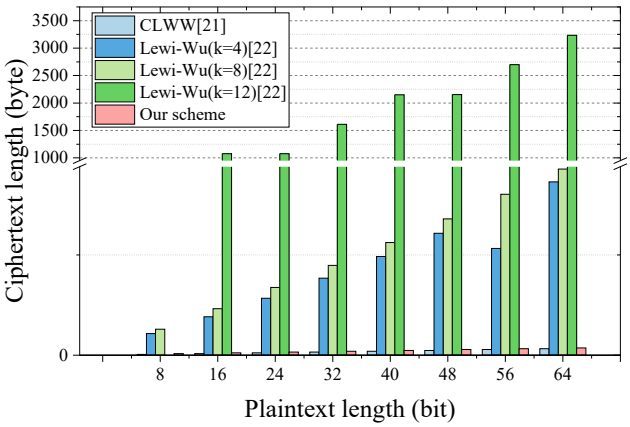


Fig. 8: Comparison of ciphertext length with different plaintext length.

5.2 Performance comparison

We first compare the ciphertext lengths of CLWW [21], Lewi-Wu-Normal [22], and our scheme. Notice that k in Lewi-Wu-Normal [22] denotes block size. Overall, Fig. 8 shows that the ciphertext length of these three schemes increases as the plaintext length increases. Meanwhile, the ciphertext lengths in our scheme are slightly longer than CLWW [21] and much smaller than Lewi-Wu-Normal [22]. Under different plaintext lengths, the ciphertext length of our scheme is always two bytes longer than CLWW [21]. However, the ciphertext length of our scheme is much shorter than Lewi-Wu-Normal [22]. When $k = 12$ and plaintext length is 64 bits, our scheme is on average $1/180$ of Lewi-Wu-Normal [22] in terms of ciphertext length.

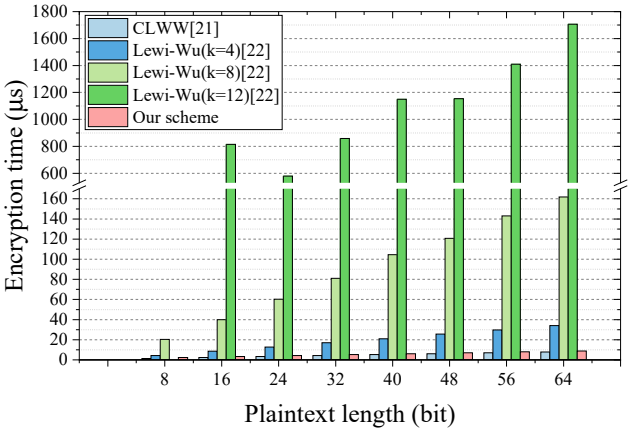


Fig. 9: Execution time of encryption process with different plaintext length.

We also evaluate the execution time of the encryption and comparison process of these three schemes. Fig. 9 presents the comparison of encryption time among CLWW [21], Lewi-Wu-Normal [22], and our scheme. Notice that the encryption time in our scheme includes the encoding process. The encryption time of thees three schemes increases with the length of the plaintext. The encryption execution time of Lewi-Wu-Normal [22] is much longer than CLWW [21] and our scheme. In detail, the encryption time of our solution is only about 0.9 μs longer than CLWW [21] on average. Because the ciphertext length of our scheme is always 2 bytes longer than CLWW [21]. The encryption time of Lewi-Wu-Normal [22] increases as k increases. When the plaintext length is 64 bits, the encryption execution time of our scheme is only 10% longer than CLWW [21]. When the plaintext length is 64 bits and $k = 4$, the encryption execution time of Lewi-Wu-Normal [22] is about 4 times that of our scheme. When the plaintext length is 64 bits and $k = 8$, the encryption execution time of Lewi-Wu-Normal [22] is about 18 times that of our scheme. When the plaintext length is 64 bits and $k = 12$, the encryption execution time of Lewi-Wu-Normal [22] is about 195 times that of our scheme.

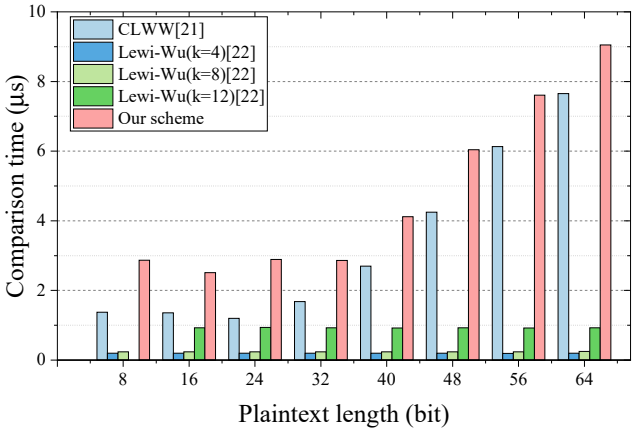


Fig. 10: Execution time of comparison process with different plaintext length.

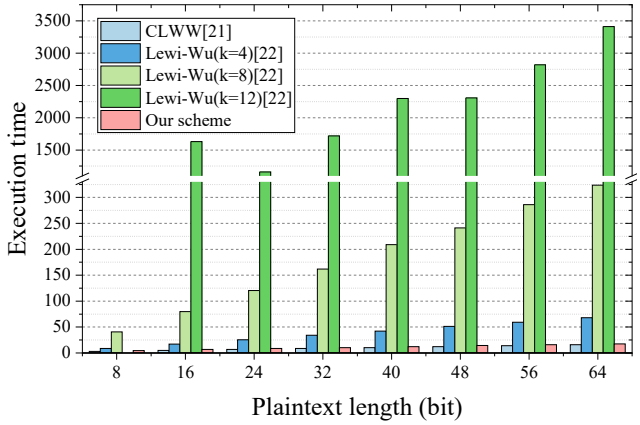


Fig. 11: Overall execution time with different plaintext length.

Fig. 10 shows the comparison of execution time in comparison process among CLWW [21], Lewi-Wu-Normal [22] and our scheme. The length of the two plaintexts being compared is equal. This result is tested with 200,000 iter-

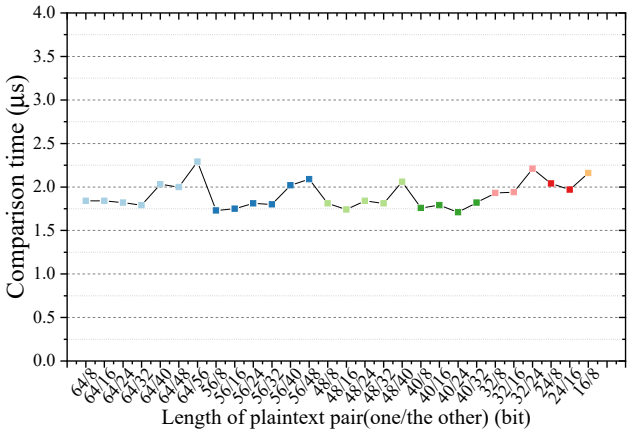


Fig. 12: Comparison time for plaintext over different lengths.

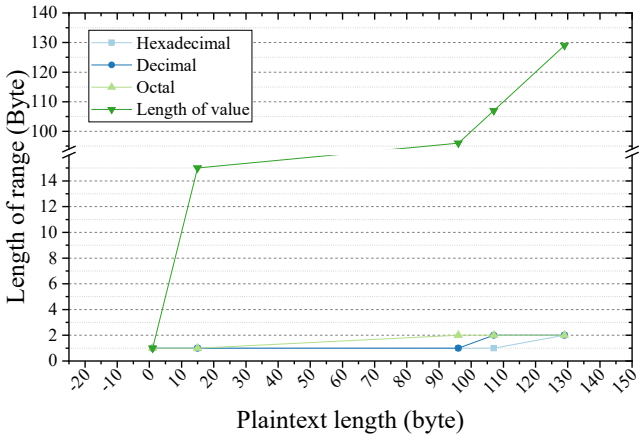


Fig. 13: Range length and value length under different base codes.

ations. The comparison of execution time is related with the plaintext. In our implementation, we choose random number as plaintext. The execution time of our scheme is longer than Lewi-Wu-Normal [22]. When $k = 12$ and plaintext length is 64 bits, the execution time of our scheme is 10 times that of Lewi-Wu-Normal [22]. Furthermore, the execution time of our scheme is longer than CLWW [21], because the range part is the same. The execution time of our scheme is $1.5\ \mu s$ longer than CLWW [21].

The comparison of the overall execution time among CLWW [21], Lewi-Wu-Normal [22] and our scheme is shown in Fig.11. Our execution time is almost equal to CLWW [21] and much less than Lewi-Wu-Normal [22]. The plaintext length of our scheme is longer than CLWW [21], so our overall implementation time will be slightly slower. Furthermore, for different plaintext lengths, the average execution time will be slower by $0.9\ \mu s$ than CLWW [21]. Compared to Lewi-Wu-Normal [22], the execution of our scheme is better. When k is equal to 4, 8, and 12, respectively, the execution time of Lewi-Wu-Normal [22] is 2, 14, and 180 times longer than that of our scheme.

5.3 Performance of our scheme

We evaluate the comparison time of plaintext with different range part in our scheme. Fig. 12 shows the time for comparing two plaintexts of different range part. The dots of the same color in the figure indicate that there is one plaintext in the compared plaintext pair whose length is the same. As shown in Fig. 12, we can find that the comparison time also slowly increases as the number of bits in another plaintext increases when one plaintext length is fixed, but the overall increase is not large. Computationally speaking, in the comparison of plaintext pairs of different lengths, the average comparison time is $1.9\ \mu$s. The experiment results show that our scheme is suitable for comparing plaintext with different range part. Table 2 presents the encode time of our scheme. Under different plaintext lengths, the encoding time of our scheme is basically the same. In fact, the encoding time of our scheme is affected by the length of the plaintext, but the encoding process is relatively efficient, so the results are relatively close. If the length of plaintext can reach hundreds or thousands of bits, the gap may be widened. On average, the encode time of our scheme is $12.7\ ns$.

TABLE 2: Encode time for plaintext of different lengths.

Plaintext length (bit)	Encode time (ns)
8	12.51
16	12.79
24	12.87
32	12.77
40	12.80
48	12.82
56	12.84
64	12.34

Fig. 13 shows the range part length and value part length of our scheme under different cardinality. The value part length of our scheme increases linearly with the length of the plaintext. The length of the range has an obvious inflection point as the length of the plaintext increases. Explain in detail, when using hexadecimal encoding, the range length will be mutated to 2 bytes when the plaintext length is 129 bytes. By analogy, the range length of decimal code and octal code will be changed to 2 bytes when the plaintext length is 107 bytes and 96 bytes respectively. Generally speaking, as the length of the plaintext increases, the range part length of our scheme is much shorter than the length of value part. Fig. 13 emphasizes that the range part always belongs to a small domain in the plaintext domain.

6 Conclusions

In this paper, we propose a segmentation coding technique that splits the encrypted plaintext into two parts: the range part and the value part. Based on the segmentation coding technique, we construct *HybridORE* model. Then, in order to further shorten the length of the ciphertext, we construct *EncodeORE* scheme. We analyze the space usage and the leakage of *HybridORE* and *EncodeORE*. Furthermore, we make experiments to compare the performance of *EncodeORE*, CLWW [21] and Lewi-Wu-Normal [22]. In the future, we will work on more practical ORE schemes.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No.61672300), National Natural Science Foundation of Tianjin (No. 18ZXZNGX00140) and National Natural Science Foundation for Outstanding Youth Foundation (No. 61722203).

REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD Conference*, pages 563-574, 2004.

[2] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224-241, 2009.

[3] T. Boelter, R. Poddar, and R. A. Popa. A Secure One-Roundtrip Index for Range Queries. In *IACR Cryptology ePrint Archive*, 2016.

[4] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, pages 578-595, 2011.

[5] W. Ding, R. Hu, Z. Yan, X. Qian, R. H. Deng, L. T. Yang, and M. Dong. An Extended Framework of Privacy-Preserving Computation With Flexible Access Control. In *IEEE Trans. Network and Service Management*, pages 17(2): 918-930, 2020.

[6] Z. Zhang, M. Dong, L. Zhu, Z. Guan, R. Chen, R. Xu, and K. Ota. Achieving Privacy-Friendly Storage and Secure Statistics for Smart Meter Data on Outsourced Clouds. In *IEEE Trans. Cloud Comput*, pages 7(3): 638-649, 2019.

[7] X. Liu, M. Dong, Y. Liu, A. Liu, and N. N. Xiong. Construction Low Complexity and Low Delay CDS for Big Data Code Dissemination. In *Complexity 2018*, pages 5429546:1-5429546:19, 2018.

[8] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *IEEE Symposium on Security and Privacy*, pages 463-477, 2013.

[9] F. Kerschbaum and A. Schröpfer. Optimal average-complexity ideal-security order-preserving encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 275-286, 2014.

[10] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *ACM Conference on Computer and Communications Security*, pages 644-655, 2015.

[11] G. Lin, S. Wen, Q. Han, J. Zhang, and Y. Xiang. Software Vulnerability Detection Using Deep Neural Networks: A Survey. In *Proceedings of the IEEE*, pages 1-24, 2020.

[12] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren. Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection. In *IEEE Transactions on Information Forensics and Security*, pages 15(1): 987-1001, 2020.

[13] F. Kerschbaum. Frequency-hiding order-preserving encryption. In *ACM Conference on Computer and Communications Security*, pages 656-667, 2015.

[14] R. Coulter, Q. Han, L. Pan, J. Zhang, and Y. Xiang. Data-Driven Cyber Security in Perspective-Intelligent Traffic Analysis. In *IEEE Transactions on Cybernetics*, pages 50(7): 3081-3093, 2020.

[15] N. Sun, J. Zhang, P. Rimba, S. Gao, and Y. Xiang. Data-driven Cybersecurity Incident Prediction and Discovery: A Survey. In *IEEE Communications Surveys and Tutorials*, pages 21(2): 1744-1772, 2019.

[16] L. Liu, O. D. Vel, Q. Han, J. Zhang, and Y. Xiang. Detecting and Preventing Cyber Insider Threats: A Survey. In *IEEE Communications Surveys and Tutorials*, pages 20(2): 1397-1417, 2018.

[17] J. Zhang, Y. Xiang, Y. Wang, W. L. Zhou, Y. Xiang, and Y. Guan. Network Traffic Classification Using Correlation Information. In *IEEE Transactions on Parallel and Distributed Systems*, pages 24(1): 104-117, 2013.

[18] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 563-594, 2015.

[19] X. Wang and Y. Zhao. Order-revealing encryption: file-injection attack and forward security. In *European Symposium on Research in Computer Security*, pages 101-121, 2018.

[20] D. Cash, F. Liu, A. O'Neill, and C. Zhang. Reducing the leakage in practical order-revealing encryption. In *IACR Cryptology ePrint Archive*, 2016.

[21] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *FSE*, pages 474-493, 2016.

[22] K. Lewi and D. J. Wu. Order-revealing encryption: new constructions, applications, and lower bounds. In *ACM Conference on Computer and Communications Security*, pages 1167-1178, 2016.

[23] D. Bogatov, G. Kollios, and L. Reyzin. A comparative evaluation of order-revealing encryption schemes and secure range-query protocols. In *VLDB*, pages 12(8):933-947, 2019.

[24] D. Cash, F. Liu, A. O'Neill, M. Zhandry, and C. Zhang. Parameter-hiding order revealing encryption. In *ASIACRYPT*, pages 181-210. 2018.

[25] O. Pandey and Y. Rouselakis. Property preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375-391, 2012.

[26] C. Mavroforakis, N. Chenette, A. O'Neill, G. Kollios, and R. Canetti. Modular order-preserving encryption, revisited. In *SIGMOD Conference*, pages 763-777, 2015.

[27] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich. POPE: Partial order preserving encoding. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1131-1142, 2016.

[28] I. Teranishi, M. Yung, and T. Malkin. Order-preserving encryption secure beyond one-wayness. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 42-61, 2014.

[29] P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO*, pages 308-326, 2015.

[30] H. Kadhem, T. Amagasa, and H. Kitagawa. A Secure

and Efficient Order Preserving Encryption Scheme for Relational Databases. In *KMIS*, pages 25-35, 2010.

[31] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F. Liu, A. Sahai, E. Shi, and H. Zhou. Multi-input functional encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 578-602, 2014.

[32] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *SIAM Journal on Computing*, pages 45(3): 882-929, 2016.

[33] L. Yehuda. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography Conference*, pages 1-25, 2014.

[34] B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 221-238, 2014.

[35] P. V. Ananth, D. Gupta, Y. Ishai, and A. Sahai. Optimizing obfuscation: avoiding Barrington's theorem. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 646-658, 2014.

[36] Z. Brakerski, I. Komargodski, and G. Segev. From Single-Input to Multi-Input Functional Encryption in the Private-Key Setting. In *IACR Cryptology ePrint Archive*, 2015.

[37] S.Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu. Function-hiding inner product encryption is practical. In *International Conference on Security and Cryptography for Networks*, pages 544-562, 2018.

[38] A. Sahai and M. Zhandry. Obfuscating Low-Rank Matrix Branching Programs. In *IACR Cryptology ePrint Archive*, 2014.

[39] M. Bun and M. Zhandry. Order-revealing encryption and the hardness of private learning. In *Theory of Cryptography Conference*, pages 176-206, 2016.

[40] H. Haagh, Y. Ji, C. Li, C. Orlandi, and Y. Song. Revealing encryption for partial ordering. In *IMA International Conference on Cryptography and Coding*, pages 3-22, 2017.

[41] F. Kerschbaum and A. Tueno. An efficiently searchable encrypted data structure for range queries. In *European Symposium on Research in Computer Security*, pages 344-364, 2017.

[42] D. X. Song, D. A. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy*, pages 44-55, 2000.

[43] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Journal of Computer Security*, pages 19(5): 895-934, 2011.

[44] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577-594, 2010.

[45] S. Chatterjee and M. P. L. Das. Property preserving symmetric encryption revisited. In *ASIACRYPT*, pages 658-682, 2015.

[46] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169-178, 2009.

[47] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference*, pages 535-554, 2007.

[48] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. In *Journal of the ACM (JACM)*, pages 43(3): 431-473, 1996.

[49] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. In *J. ACM*, pages 33(4): 792-807, 1986.

[50] The OpenSSL Project, "OpenSSL: The open source toolkit for SSL/TLS", URL: https://www.openssl.org, 2003.

[51] T. Granlund and the GMP development team," GNU MP: The GNU Multiple Precision Arithmetic Library", URL: https://gmplib.org/, 2012.

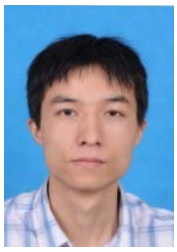
Zheli Liu received the BSc and MSc degrees in computer science from Jilin University, China, in 2002 and 2005, respectively. He received the PhD degree in computer application from Jilin University in 2009. After a postdoctoral fellowship in Nankai University, he joined the College of Computer and Control Engineering of Nankai University in 2011. Currently, he works at Nankai University as a Associate Professor. His current research interests include applied cryptography and data privacy protection.

Siyi Lv received Bachelor Degree of Information Security and Law from Nankai University, Tianjin, China, in 2016. Currently, she studies for the doctor degree in computer science at Nankai University. Her research interests include applied cryptography, data privacy protection.

Jin Li received the BS degree inmathematics from Southwest, University, in 2002 and the PhD degree in information security from Sun Yat-sen University, in 2007. Currently, he works at Guangzhou University as a Professor. He has been selected as one of science and technology new star in Guangdong province. His research interests include applied cryptography and security in cloud computing. He has published over 50 research papers in refereed international conferences and journals and has served as the program chair or program committee member in many international conferences.

Yanyu Huang received Bachelor Degree of Information Security from China University of Geosciences, Wuhan, China, in 2016. Currently, she studies for the doctor degree in computer science at Nankai University. Her research interests include applied cryptography, data privacy protection.

Yali Yuan received the M.Sc. degree from University of Lanzhou, Lanzhou, China, in 2015 and the Ph.D. degree in University of Göttingen, Göttingen, Germany in 2018 where she is currently working as a Postdoctoral Fellow. Her research interests include various topics related to wireless networks, in particular for the intelligent network and security.



Liang Guo received the MSc and PhD degree from Beijing University of Technology. He joined Huawei Technologies Co.,Ltd. in 2016. Currently he works at Huawei as a Chief Engineer. His current research interests include data privacy protection and database security.

Changyu Dong received the Ph.D. degree from Imperial College London. He is currently a Senior Lecturer with the School of Computing, Newcastle University. He has authored over 30 publications in international journals and conferences. His research interests include applied cryptography, trust management, data privacy, and security policies. His recent work focuses mostly on designing practical secure computation protocols. The application domains include secure cloud computing and privacy preserving data mining.