J Inf Process Syst, Vol.15, No.5, pp.1211~1217, October 2019
https://doi.org/10.3745/JIPS.03.0128

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

New Construction of Order-Preserving Encryption Based on Order-Revealing Encryption

Kee Sung Kim*

Abstract
Developing methods to search over an encrypted database (EDB) have received a lot of attention in the last few years. Among them, order-revealing encryption (OREnc) and order-preserving encryption (OPEnc) are the core parts in the case of range queries. Recently, some ideally-secure OPEnc schemes whose ciphertexts reveal no additional information beyond the order of the underlying plaintexts have been proposed. However, these schemes either require a large round complexity or a large persistent client-side storage of size $O(n)$ where n denotes the number of encrypted items stored in EDB. In this work, we propose a new construction of an efficient OPEnc scheme based on an OREnc scheme. Security of our construction inherits the security of the underlying OREnc scheme. Moreover, we also show that the construction of a non-interactive ideally-secure OPEnc scheme with a constant client-side storage is theoretically possible from our construction.

Keywords
Database Encryption, Order-Preserving Encryption, Order-Revealing Encryption

1. Introduction

One of promising solutions to protect the confidentiality of sensitive data is to use property-preserving encryption schemes which preserve some property of plaintexts, then perform query evaluation over the encrypted database (EDB). Supporting efficient operations on EDB such as sorting and range queries, order-preserving encryption (OPEnc) [1,2] and order-revealing encryption (OREnc) [3-5] have been proposed.

OREnc is a special type of symmetric encryptions which leaks the order of the underlying plaintexts through a publicly computable comparison function. Recently, Boneh et al. [5] proposed the first stateless and non-interactive OREnc scheme which achieves the ideal security. However, their OREnc scheme relies on multilinear maps requiring heavy computation and strong assumption, also suffering from security analysis [6,7], thus is not efficiently implementable today. Chenette et al. [3] proposed the first efficiently-implementable OREnc scheme based on a pseudo-random function. They also provided a new security model that precisely quantifies what information of plaintexts is leaked. Later, Lewi and Wu [4] proposed a new OREnc scheme with reduced leakage as compared to [3].

OPEnc is a special case of OREnc whose ciphertexts have the same order as their plaintexts, thus it

Manuscript received December 21, 2018; accepted January 22, 2019.
Corresponding Author: Kee Sung Kim (kee21@cu.ac.kr)
* School of Information Technology Engineering, Daegu Catholic University, Gyeongsan, Korea (kee21@cu.ac.kr)

enables a server to get the same results as if it had operated on plaintexts without any modification on DBMS. It has been known that any immutable ideally-secure OPEnc schemes must have the exponentially large ciphertext space. Recently, some interactive and mutable order-preserving encryption (or encoding) schemes with the ideal security have been proposed. Kerschbaum [1] proposed the first frequency-hiding OPEnc scheme that randomizes the ciphertexts to hide the frequency of the underlying plaintexts. Recently, Roche et al. [2] gave a partial order-preserving encryption (POPEnc) scheme that is mainly for the application scenarios where a large number of insertions and a moderate number of range queries, and achieves even stronger security compare to [1]. However, these ideally-secure OPEnc schemes either suffer from a large round and client-side storage complexities or incomplete functionalities, as described in Table 1.

In this work, we propose a possible answer to the following question: Is it possible to design a non-interactive ideally-secure OPEnc with a constant-size client storage?

More specifically, we present how to improve the round and client-side storage complexities on the exiting ideally-secure mutable order-preserving encryption protocol using the public comparison functionality of an OREnc scheme. By composing our method with an existing OREnc scheme, we can obtain an efficient OPEnc scheme which security is at least as strong as one of the underlying OREnc scheme. Table 1 presents comparison our constructions based on [3-5] with the existing ideally-secure OPEnc schemes in terms of efficiency and security. Composing with [3,4], the resulting constructions show better efficiency both of the number of rounds and the client-side storage, but cannot guarantee the ideal security. Note that the construction based on [5] is the first non-interactive ideally-secure OPEnc scheme with a constant-size client storage, but it is currently impractical to implement as described before.

Table 1. Comparison of the existing ideally-secure OPEnc schemes

Scheme	Rounds		Client storage		Security	Note
	Insert	Query	Working	Persistent		
Previous constructions						
[1]	1 [a]	1 [a]	$O(n)$	$O(n)$	Ideal	F.H.
[2]	1 [b]	$O(1)$ [b]	$O(n^{\varepsilon})$ $(0 < \varepsilon < 1)$	$O(1)$	Ideal	F.H. Partial OPE
Our constructions						
Based on [3]	1	1	$O(1)$	$O(1)$	Diff. Bit	-
Based on [4]	1	1	$O(1)$	$O(1)$	Diff. Block	-
Based on [5]	1	1	$O(1)$	$O(1)$	Ideal	Impractical

n is the total number of encrypted data in EDB. Diff. Bit (Block) indicates the position of the first differing bit (block) of two corresponding plaintexts. F.H. means the frequency-hiding property. Partial OPE means a sizable fraction of ciphertexts in EDB remains incomparable.
[a]Worst case: $O(n)$, [b]worst case: $O(n^{1-\varepsilon})$.

The rest paper is organized as follows: Section 2 reviews the formal notion and security of OREnc (OPEnc). In Section 3, we propose our new construction of OPEnc based on OREnc. In Section 4, we analyze our construction. Section 5 concludes this paper.

2. Preliminaries

We write λ as a security parameter. For two bit strings $x, y \in \{0,1\}^*$, $x \,||\, y$ denotes the concatenation of x and y. We say that two distributions D_1 and D_2 are computationally indistinguishable if there is no efficient poly-time adversary can distinguish D_1 from D_2 except with negligible probability.

2.1 Formal Notion of Order-Revealing Encryption

An order-revealing encryption scheme $\prod_{\text{OREnc}}$ consists of the following three algorithms (OREnc.Setup, OREnc.Encrypt, OREnc.Compare) defined over a well-ordered domain D and a range R.

- OREnc.Setup(1^λ) → *skey* : For a security parameter λ, this setup algorithm generates a secret key *skey*.
- OREnc.Encrypt(*msg*, *skey*) → *ctx* : For a secret key *skey* and a message $msg \in D$, this encryption algorithm computes a ciphertext $ctx \in R$.
- OREnc.Compare(ctx_1, ctx_2) → b : On input two ciphertexts ctx_1 and ctx_2, this comparison algorithm outputs a bit $b \in \{0, 1\}$. ($b = 1$ means $msg_1 < msg_2$)

Correctness. For a fixed security parameter λ, a given $\prod_{\text{OREnc}}$ is correct if for $skey \leftarrow$ OREnc.Setup(1^λ), and any messages $msg_1, msg_2 \in D$ ($msg_1 < msg_2$), OREnc.Compare(ctx_1, ctx_2) = 1 where $ctx_1 \leftarrow$ OREnc.Encrypt(msg_1, *skey*) and $ctx_2 \leftarrow$ OREnc.Encrypt(msg_2, *skey*).

2.2 Formal Notion of Order-Preserving Encryption

An order-preserving encryption scheme $\prod_{\text{OPEnc}}$ consists of the following two algorithms (OPEnc.Setup, OPEnc.Encrypt) defined over a well-ordered domain D and a range R.

- OPEnc.Setup(1^λ) → *skey* : For a security parameter λ, this setup algorithm generates a secret key *skey*.
- OPEnc.Encrypt(*msg*, *skey*) → *ctx* : On input the secret key *skey* and a plaintext $msg \in D$, this encryption algorithm computes a ciphertext $ctx \in R$.

Correctness. For a fixed security parameter λ, a given $\prod_{\text{OPEnc}}$ is correct if for $skey \leftarrow$ OPEnc.Setup(1^λ), and any messages $msg_1, msg_2 \in D$ ($msg_1 < msg_2$), $ctx_1 < ctx_2$ where $ctx_1 \leftarrow$ OPEnc.Encrypt(msg_1, *skey*) and $ctx_2 \leftarrow$ OPEnc.Encrypt(msg_2, *skey*). Thus, OPEnc can be seen as a special case of OREnc where the comparison algorithm output 1 if $ctx_1 < ctx_2$.

As described above, these two schemes don't contain a decryption algorithm, but the following two options can be applied. Note that the secret key holder can generate a ciphertext *ctx* of any messages he choose, and he can also verify the output of OREnc.Compare(ctx, ctx^*) where ctx^* is the target ciphertext. Thus, he can decrypt ctx^* by performing the binary search. Another method which avoids the logarithmic scale binary search overhead is combining with a CPA-secure encryption scheme, i.e., inserting an encryption of the same message under this symmetric encryption scheme together. Note that this additional ciphertext doesn't reveal any information about the underlying plaintext due to the CPA security.

2.3 Security of OR(P)Enc

In this section, we review a simulation-based security model [3] of OREnc that precisely quantifies what information of plaintexts is leaked as defining a leakage function. We denote an adversary and a simulator for some q = poly(λ) by $Adv = (A_1,\ldots, A_q)$ and $Sim = (S_0,\ldots, S_q)$, respectively. $\prod_{\text{OREnc}}$ = (OREnc.Setup, OREnc.Encrypt, OREnc.Compare) be an OREnc scheme, and $Lkg(\cdot)$ denotes a leakage function of $\prod_{\text{OREnc}}$. For a security parameter λ, the experiment is defined as follow:

```
$REAL_A^{\Pi}(\lambda)$:
  1. $skey \leftarrow$ OREnc.Setup($1^\lambda$)
  2. $(msg_1, st_A) \leftarrow A_1(1^\lambda)$
  3. $ctx_1 \leftarrow$ OREnc.Encrypt($msg_1$, $skey$)
  4. for $2 \leq i \leq q$:
       $(msg_i, st_A) \leftarrow A_i(st_A, ctx_1,\ldots, ctx_{i-1})$
       $ctx_i \leftarrow$ OREnc.Encrypt($msg_i$, $skey$)
  5. output $(ctx_1,\ldots, ctx_q)$ and $st_A$
```

```
$SIM_S^{\Pi}(\lambda)$:
  1. $st_s \leftarrow S_0(1^\lambda)$
  2. $(msg_1, st_A) \leftarrow A_1(1^\lambda)$
  3. $(ctx_1, st_s) \leftarrow S_1(Lkg(msg_1), st_s)$
  4. for $2 \leq i \leq q$:
       $(msg_i, st_A) \leftarrow A_i(st_A, ctx_1,\ldots, ctx_{i-1})$
       $(ctx_i, st_s) \leftarrow S_i(Lkg(msg_1,\ldots, msg_i), st_s)$
  5. output $(ctx_1,\ldots, ctx_q)$ and $st_A$
```

We say that $\prod_{\text{OREnc}}$ is a secure with $Lkg(\cdot)$ if for all poly-size adversaries Adv, there exists a simulator Sim such that the two distributions $REAL_A^{\Pi}(\lambda)$ and $SIM_S^{\Pi}(\lambda)$ are computationally indistinguishable. From the security notion, we also say that $\prod_{\text{OREnc}}$ is a ideally-secure if the leakage function $Lkg(\cdot)$ reveals only the relative order of the underlying plaintexts. Note that we can apply the same experiment to an OPEnc scheme since it can be seen as a special case of OREnc.

3. Proposed Scheme

As described in Section 1, it is natural to obtain an OREnc scheme from a given OPEnc scheme since the comparison algorithm of the OREnc scheme can simply return 1 if $ctx_1 < ctx_2$ ciphertexts of OPEnc. The authors [3] showed how to compose an OPEnc scheme with an OREnc scheme. Their main idea is to encrypt a message with the OPEnc scheme, then use this ciphertext as an input of the encryption algorithm of OREnc. They also showed that the security of the resulting OREnc scheme is at least as strong as the security of the underlying OPEnc scheme. However, there has been no generic construction of converting an OREnc scheme into an OPEnc scheme yet. Although some previous works including [3] suggested methods to convert their OREnc scheme into OPEnc schemes, this is not generic construction. In this section, we propose a generic construction of an OPEnc scheme from a given ORE scheme.

For a security parameter λ, a domain $D = (1,\ldots, M)$, a range $R = (1,\ldots, N)$, and an encrypted database EDB, we define our construction of an OPEnc scheme based on an OREnc scheme as follows (note that OPEnc.Setup(1^λ) is identically defined as OREnc.Setup(1^λ) of the underlying OREnc).

Algorithm 1. OPEnc.Encrypt

```
Input : a plaintext $msg$ and a secret $skey$ of OREnc
Output : a ciphertext $c$

Client :
    1.  The client runs $ctx_1 \leftarrow$ OREnc.Encrypt($msg$, $skey$) and to sends $ctx_1$ to the server.

Server :
    1.  If EDB = $\emptyset$, then server inserts ($-1 \| \cdot$) and ($N \| \cdot$) .
    2.  Using OREnc.Compare and $ctx_1$, the server finds the largest $ctx_x = (ctx_{x,0} \| ctx_{x,1})$
        and the smallest $ctx_y = (ctx_{y,0} \| ctx_{y,1})$ such that $msg_x \leq msg < msg_y$.
        (Here, $msg_x$ and $msg_y$ denote the plaintexts of $ctx_x$ and $ctx_y$, respectively.)
    3.  If $msg_x = msg$, then the server inserts $(ctx_{x,0} \| ctx_{x,1})$ as an encryption $c$ of $msg$.
    4.  Else If $ctx_{y,0} - ctx_{x,0} = 1$, the server runs Update(EDB, -1, $N$), and goes to Step 2.
    5.  Else If the server inserts $c = (ctx_0 \| ctx_1)$ where $ctx_0 = \lceil (ctx_{x,0} + ctx_{y,0}) / 2 \rceil$.
```

Algorithm 2. Update

```
Input : sorted distinct set { $(ctx_{1,0} \| ctx_{1,1}), \ldots, (ctx_{n,0} \| ctx_{n,1})$ }, a min value $a$ and a max value $b$
Output : balanced sorted distinct set { $(ctx'_{1,0} \| ctx_{1,1}), \ldots, (ctx'_{n,0} \| ctx_{n,1})$ }

Server :
    1.  Compute $k \leftarrow \lceil (a + b) / 2 \rceil$.
    2.  If $n = 1$, the server updates all $ctx_{1,0}$ as $k$.
    3.  If $n = 2$, the server updates all $ctx_{1,0}$ as $k$ and runs Update({ $(ctx_{2,0} \| ctx_{2,1})$ }, $k$, $b$).
    4.  If $n > 3$, the server updates all $ctx_{i,0}$ as $k$ where $i = \lfloor n/2 \rfloor + 1$,
            runs Update({ $(ctx_{1,0} \| ctx_{1,1}), \ldots, (ctx_{i-1,0} \| ctx_{i-1,1})$ }, $a$, $k$) and
            runs Update({ $(ctx_{i+1,0} \| ctx_{i+1,1}), \ldots, (ctx_{n,0} \| ctx_{n,1})$ }, $k$, $b$).
```

The encryption of our construction consists of two parts ctx_0 and ctx_1 ciphertexts of OPEnc and OREnc, respectively. Basically, the encoding and updating methods of ctx_0 can be seen as the non-frequency hiding construction of [1], and it has been known to be ideally-secure. Due to the order-preserving property of ctx_0, the resulting ciphertext $ctx_0 \| ctx_1$ also preserves the relative order of the underlying plaintext. More specifically, to encrypt a given message msg, a client who has a secret key generates an encryption ctx_1 of an underlying OREnc scheme, then sends it to a server. Using OREnc.Compare and ctx_1, the server can find the largest $ctx_{x,0}$ and the smallest $ctx_{y,0}$ as describe in Step 2. Here, $(ctx_{x,0} + 1, \ldots, ctx_{y,0} - 1)$ means the range in which the ctx_0 can be exist. ctx_0 is simply computed as the intermediate value between $ctx_{x,0}$ and $ctx_{y,0}$. Note that if ciphertext space does not exist, i.e., $ctx_{y,0} - ctx_{x,0} = 1$, the server should perform the above update algorithm (Algorithm 2) to reconstruct ctx_0's in EDB.

4. Analysis

In this section, we analyze our construction in terms of correctness and security.

THEOREM 1. (Correctness) Our proposed OPEnc scheme is correct if the underlying OREnc scheme is correct.

Proof. It is sufficient to show that the underlying OREnc is not correct if the resulting OPEnc scheme is not correct. We assume that there exist ctx_a and ctx_b ($ctx_a > ctx_b$) encryptions of msg_a and msg_b ($msg_a < msg_b$). Without loss of generality, we also assume that msg_a is encrypted as $ctx_a = (ctx_{a,0}, ctx_{a,1})$ first. We know that $ctx_a > ctx_b$ implies $ctx_{a,0} > ctx_{b,0}$. When $ctx_{b,0}$ is generated during the OPEnc.Encrypt algorithm where $ctx_b = (ctx_{b,0}, ctx_{b,1})$, the part of determining the relative order of $ctx_{b,0}$ is only the OREnc.Compare algorithm in Step 2. As a result, it implies that the underlying OREnc scheme is not correct.

Theorem 2. (Security) Our proposed OPEnc scheme is secure with leakage function $Lkg(\cdot)$ of the underlying OREnc scheme.

Proof. We write $L_{\text{OPEnc}}(\cdot)$ and $L_{\text{OREnc}}(\cdot)$ to the leakage functions of our proposed OPEnc scheme and the underlying OREnc scheme, respectively. As described before, the generation and updating methods of ctx_0 can be seen as the non-frequency hiding construction of [1], and it has been known to be ideally secure. This means that the first part of the resulting ciphertext c_0 reveals only the relative ordering of the underlying plaintext. Therefore, a simulator Sim with $L_{\text{OREnc}}(\cdot)$ who can simulates ctx_1 is always able to simulate ctx_0 unless the underlying OREnc scheme provides more strong security than the ideal security.

Theorem 3. (Efficiency) Our proposed OPEnc scheme provides a non-interactive encryption and a range query with a constant client-side storage.

Proof. From the specification of Algorithm 1, we know that it doesn't require any additional rounds to encrypt data with a server. The client simply creates ctx_1, then sends it to the server. The rest of the encryption part is done by the server. Although we have not described in detail how the range query works, it is basically done by sending two encrypted boundary points to the server. It means that range query also does not require any additional rounds.

The client should maintain the secret key $skey$ of the OREnc scheme to generate ctx_1. Since the computation of ctx_0, which requires all of the existing ciphertext information, is performed on the server-side, the client does not need to maintain any additional state except $skey$.

Remarks 1. One of the interesting points of our proposed scheme is that the client and the server each compute half of the ciphertext. Some sensitive readers might think that it is not natural that the server who does not have a secret key generates the final ciphertext. However, this has no effect on functionalities of OPEnc such as decryption and range queries, and also no effect on the security since the part of ciphertext generated by the server is computed using only the information that has been already disclosed.

Remarks 2. The ciphertext of our proposed OPEnc scheme consists of two ciphertexts of OPEnc and OREnc. This means that each OPEnc and OREnc encryption algorithms have to be performed to generate the ciphertext, thus it can be think that it requires roughly more than twice computational overhead compared to previous works. However, we know that the client generates only half of the ciphertext, and the server who has powerful computation power completes the encryption process, thus it is hard to say that efficiency of our proposed scheme is worse than the existing OPEnc schemes.

5. Conclusions

In this work, we proposed a new construction of an OPEnc scheme based on an OREnc scheme with the optimal client storage and round complexities. The security of the resulting OPEnc scheme is at least as strong as the underlying OREnc's security. We also gave comparison result our construction with the existing ideally-secure OPEnc schemes in terms of efficiency and security. Finally, from our construction, we showed that it's theoretically possible to construct a non-interactive ideally-secure OPEnc scheme with a constant client-side storage.

Acknowledgement

This work was supported by research grants from Daegu Catholic University in 2019.

References

[1] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, CO, 2015, pp. 656-667.

[2] 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*, Vienna, Austria, 2016, pp. 1131-1142.

[3] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu, "Practical order-revealing encryption with limited leakage," in *Fast Software Encryption*. Heidelberg: Springer, 2016, pp. 474-493.

[4] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, 2016, pp. 1167-1178.

[5] 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 *Advances in Cryptology – EUROCRYPT 2015*. Heidelberg: Springer, 2015, pp. 563-594.

[6] E. Miles, A. Sahai, and M. Zhandry, "Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13," in *Advances in Cryptology – CRYPT 2016*. Heidelberg: Springer, 2016, pp. 629-658.

[7] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehle, "Cryptanalysis of the CLT13 multilinear map," *Journal of Cryptology*, vol. 32, no. 2, pp. 547-565, 2019.

Kee Sung Kim https://orcid.org/0000-0001-9160-8692

He received M.S. and Ph.D. degrees in Graduate School of Information Security from Korea University, Seoul, Korea, in 2011 and 2015, respectively. He is currently an assistant professor at School of Information Technology Engineering, Daegu Catholic University, Korea. His research interests focus on cryptography, database security, privacy enhancing technology, and secure protocols.