J Inf Process Syst, Vol.14, No.5, pp.1087~1101, October 2018
https://doi.org/10.3745/JIPS.01.0032

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

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

GOPES: Group Order-Preserving Encryption Scheme Supporting Query Processing over Encrypted Data

Hyunjo Lee*, Youngho Song*, and Jae-Woo Chang*

Abstract

As cloud computing has become a widespread technology, malicious attackers can obtain the private information of users that has leaked from the service provider in the outsourced databases. To resolve the problem, it is necessary to encrypt the database prior to outsourcing it to the service provider. However, the most existing data encryption schemes cannot process a query without decrypting the encrypted databases. Moreover, because the amount of the data is large, it takes too much time to decrypt all the data. For this, Programmable Order-Preserving Secure Index Scheme (POPIS) was proposed to hide the original data while performing query processing without decryption. However, POPIS is weak to both order matching attacks and data count attacks. To overcome the limitations, we propose a group order-preserving data encryption scheme (GOPES) that can support efficient query processing over the encrypted data. Since GOPES can preserve the order of each data group by generating the signatures of the encrypted data, it can provide a high degree of data privacy protection. Finally, it is shown that GOPES is better than the existing POPIS, with respect to both order matching attacks and data count attacks.

Keywords

Cloud Computing, Data Encryption, Query Processing over Encrypted Data

1. Introduction

As cloud computing has become a widespread technology, studies on database outsourcing have been spotlighted [1-4]. However, since the outsourced database may contain sensitive personal information of users, such as health and finances, malicious attackers can obtain the private information that has leaked from the service provider in the outsourced databases. To resolve this problem, it is important to encrypt the database prior to outsourcing it to the service provider. However, with the most data encryption schemes, it is impossible to process a query over the encrypted database without decrypting it. For this, the order-preserving encryption scheme (OPES) has been proposed to do query processing over the encrypted data [5-8]. Because OPES keeps the encrypted values in the same order as that of the original data while hiding the original data, a service provider can perform query processing without decryption. One of the most promising studies is a Programmable Order-Preserving Secure Index Scheme (POPIS) that divides the original data into a set of groups based on the range of value [9]. When POPIS encrypts the data with an encryption function for each data group, it adds a random noise

Manuscript received July 5, 2017; first revision August 24, 2017; accepted September 24, 2017.
Corresponding Author: Jae-Woo Chang (jwchang@jbnu.ac.kr)
* Dept. of Computer Engineering, Chonbuk National University, Jeonju, Korea (o2near@gmail.com, {songyoungho, jwchang}@jbnu.ac.kr)

to hide the exact value of the original data. However, since the encrypted data using POPIS are sorted by a given column, POPIS is weak to both order matching attacks and data counting attacks.

To solve the problem, we propose a group order-preserving data encryption scheme (GOPES) that can support efficient query processing over encrypted data. Our GOPES consists of data grouping, data transformation and signature generation. First, because of the data grouping with k-anonymity, GOPES can preserve the order of data groups by containing more than k number of data, thus protecting the encrypted data from the order matching attacks. Secondly, because the data transformation is based on periodic functions, GOPES can change the frequency of data by assigning the same transformation value to the data falling into the same encryption group. Thus, GOPES can preserve the encrypted data from the data counting attacks. Finally, GOPES can efficiently perform both an exact matching query and a range query over the encrypted data, due to the signature generation. Our contributions can be summarized as follows:

- We propose a GOPES that can support efficient query processing over encrypted data. GOPES consists of data grouping, data transformation and signature generation.
- We propose a k-anonymity based data grouping algorithm that can preserve the order of data groups by containing more than k number of data, thus protecting the encrypted data from the order matching attacks.
- We propose a periodic function based data transformation algorithm that can change the frequency of data by assigning the same transformation value to the data falling into the same encryption group, thus preserving the encrypted data from the data counting attacks.
- We propose a signature generation algorithm that can support both an exact matching query and a range query on the encrypted data efficiently.
- We show from our performance analysis that GOPES is better than the existing POPIS, with respect to the degree of data privacy protection.

The rest of the paper is organized as follows. First, we introduces related work in Section 2. Second, we describe our group order preserving data encryption scheme in Section 3. Thirdly, we compare our GOPES with the existing work in Section 4. Finally, we conclude the paper in Section 5.

2. Related Work

Liu and Wang [9] proposed a POPIS that is an OPES using a monotone increasing function as shown in Eq. (1).

$$E(x) = ax + b + noise \tag{1}$$

In Eq. (1), a noise means a randomly selected value with the range of [0, a) to preserve the order of data. Moreover, by using the noise, POPIS can hide the original data, even if the attacker estimates the transformation function with the known input value x. For example, assume that a coefficient value a is 3, a y-intercept value b is 4 and input values x_1 and x_2 are 3 and 2, respectively. Here, x_1 and x_2 become known to an attacker. With Eq. (1), POPIS can transform x_1 to $E(3) = 5 \times 3 + 4 + noise$ and x_2 to $E(2) = 5 \times 2 + 4 + noise$. Because the range of the noise is [0, 5), the scheme can randomly select the

value of $E(2)$ as 15, and that of $E(3)$ as 22. By using the results $E(2) = 15$ and $E(3) = 22$, the attacker estimates the transformation function as which is quite different from the original function. As a result, POPIS can preserve the original data.

On the other hand, to improve the data privacy, POPIS groups the data based on the range of the value and applies a differential encryption function for each group. For example, there exist two data groups G_i and G_j. The set of coefficient and y-intercept of G_i is (a_i, b_i) and that of G_j is (a_j, b_j). In POPIS, if $i \neq j$, then $a_i \neq a_j$ and $b_i \neq b_j$. Additionally, if $i < j$, then $a_i < a_j$ and $b_i < b_j$. Fig. 1 shows an example of encryption of data for each data group.

if ($x > 500$), a=10, b=7
else if ($x > 400$), a=8, b=6
else if ($x > 300$), a=6, b=5
else if ($x > 200$), a=4, b=4
else a=2, b=3

Fig. 1. An example of encryption of data for each data group.

However, POPIS has some problems. At first, POPIS is weak to order matching attacks because its encrypted data preserves the order of the original data. Secondly, due to the differential encryption functions, POPIS has a problem that it allows the attacker to estimate the exact range of the data group. Finally, the range of the noise is considerably small because the noise is selected based on both the coefficient value and the type of the original data. As a result, the probability of data leakage is increased by using the attacker's estimated function.

3. Group Order-Preserving Encryption Scheme

3.1 Problems of the Existing POPIS

As shown in Fig. 2(a), the existing POPIS preserves the order of the original data while encrypting them. Thus, it can process the query without decryption. However, the existing method has two main types of privacy threats. For the explanation of the two types of threats (i.e., attacks), we suppose that malicious attackers know both the original keys of Pname and the encrypted data that have leaked from the service provider. At first, it is weak to the order matching attack which can obtain the original data by matching the known data column to its corresponding encryption data column based on their order of values. For example, in Fig. 2(b), an attacker can match the known column Pname = {Alice, Bob, Charley} with its encrypted column Key(Pname) = {1, 5, 10}, based on the order of values. So he/she can infer that the encrypted key values for Alice, Bob, and Charley are 1, 5 and 10, respectively. Secondly, it is weak to the data count attack which can obtain the original data by using the frequency of values. For example, in Fig. 2(c), the attacker can count the frequency of each key value from the known column Pname and its corresponding column Key(Pname). With comparing the frequencies, he/she can infer that {1,1,1}=Alice, {5,5}=Bob, and {10}=Charley, respectively.

Pname	Disease	Addr.	Phone	Date
Alice	Heart	Seoul	123-456	03-08
Alice	Heart	Seoul	123-456	03-18
Alice	Heart	Seoul	123-456	03-26
Bob	Lung cancer	Daejeon	212-509	04-12
Bob	Lung cancer	Daejeon	212-509	04-15
Charley	Flu	Jeonju	112-114	02-20

Order Preserving Encryption

Key(Pname)	Disease	Addr.	Phone	Date
1	Heart	Seoul	123-456	03-08
1	Heart	Seoul	123-456	03-18
1	Heart	Seoul	123-456	03-26
5	Lung cancer	Daejeon	212-509	04-12
5	Lung cancer	Daejeon	212-509	04-15
10	Flu	Jeonju	112-114	02-20

(a)

Pname
Alice
Bob
Chalie

Key(Pname)	Disease	...
1	Heart	...
...	...	...
5	Lung cancer	...
...	...	...
10	Flu	...

(b)

Pname	Frequency
Alice	3
Bob	2
Chalie	1

Key(Pname)	Disease	...
1	Heart	...
1	Heart	...
1	Heart	...
5	Lung cancer	...
5	Lung cancer	...
10	Flu	...

(c)

Fig. 2. Problems of the existing order-preserving encryption scheme. (a) An example of the order-preserving encryption. (b) Order matching attack. (c) Data count attack.

3.2 Overall Architecture of GOPES

Fig. 3 shows the architecture of our GOPES that can support efficient query processing on encrypted data. GOPES consists of four parts: transformation based on data group, transformation based on function segment, transformation based on periodic function, and signature generation. First, GOPES makes groups of the original data by considering k-anonymity and transforms the data of a group. Our GOPES can protect the encrypted data from the order matching attack because it can preserve the order of data groups, instead of the order of the encrypted data. Secondly, because GOPES can transforms the data based on the segments by using a periodic function such that the data in the same segment have

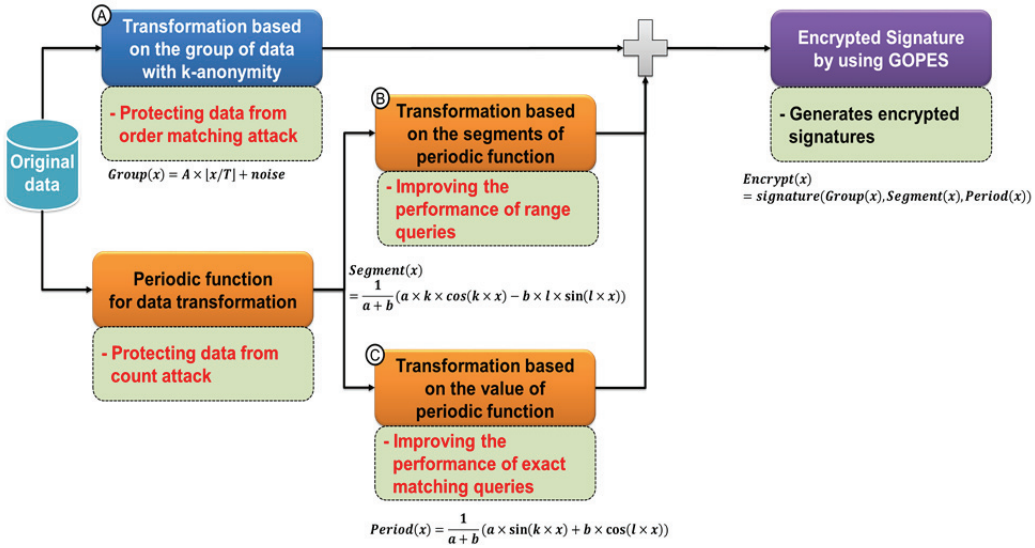


Fig. 3. Architecture of our GOPES.

the same encryption value, it can modify the frequency of the encrypted data, thus protecting the encrypted data from the data count attack. Thirdly, because GOPES can modify the data based on the periodic function for the exact matching query, it can shuffle the order of the encrypted data, thus reducing a probability that the original data can be disclosed from the order matching attack. Finally, GOPES generates the encrypted signatures by concatenating group transformation, segment transformation, and function transformation values where a signature means the abstract value of the data. Therefore, GOPES can change one-to-one relationship between the original data and encrypted data into one-to-many relationship, thus inhibiting attackers from obtaining the original data from the encrypted one.

3.3 Data Group based Transformation with k-Anonymity against Order Matching Attacks

We propose a data group based transformation algorithm supporting k-anonymity so that we can protect the encrypted data from order matching attacks. Our transformation algorithm consists of two parts: data group generation by clustering the k number of similar data and group data transformation by using a random value. Our algorithm can preserve the order of data groups since the range of the selected random value is set based on the order of the groups. Using the data group based transformation has two main advantages. First, the probability of data disclosure from the attacker is reduced to $1/k$ since each group contains at least k number of random values. Second, it is very difficult to find out the distribution of transformed data from the original data because GOPES does not preserve the order of the original data. Eq. (2) shows the data grouping function. Here, $u(x)$ means a unit function that outputs a result value 0 if $x<0$. Otherwise, a result value is 1. $f(x)$ means a unit step function based on $u(x)$ and T_i means the maximum value of the i-th data group.

$$u(x) = \begin{cases} 0, where\ x < 0 \\ 1, where\ x < 1 \end{cases}$$
$$f(x) = A(u(x) + u(x - T_i) + u(x - T_{i+1}) + \cdots) \quad (2)$$

Eq. (3) shows a regulation function of Eq. (2) where x means the original value, T means a size of the original data for each group, and A means a size of the encrypted data for each group. For the privacy reason, A has to be greater than 2^*T. To preserve the order of the data groups, the range of the *random_noise* is set to [0, A). For example, if T=10, A=2^*T, and x=100, the input x is randomly selected in a range [200, 220).

$$\text{Group(x)} = A \times \lfloor x/T \rfloor + random\_noise \quad (3)$$

In case of the skewed data set, some data groups may contain the less number of data than k. To solve this problem, the algorithm stores the number of data for each group. If the number of data for a group is less than k, our algorithm increases the size of the original data group and reconstructs all data groups. For example, if a group has less number of data than k when T=10, the algorithm reconstructs all the groups by increasing T. Fig. 4 shows an example of the data group based transformation for supporting k-anonymity. In Fig. 4, the order of the encrypted data in the same group is shuffled, so our algorithm can protect the data from the order matching attacks. Because our algorithm can preserve the order of data groups, it can support the encrypted query processing.

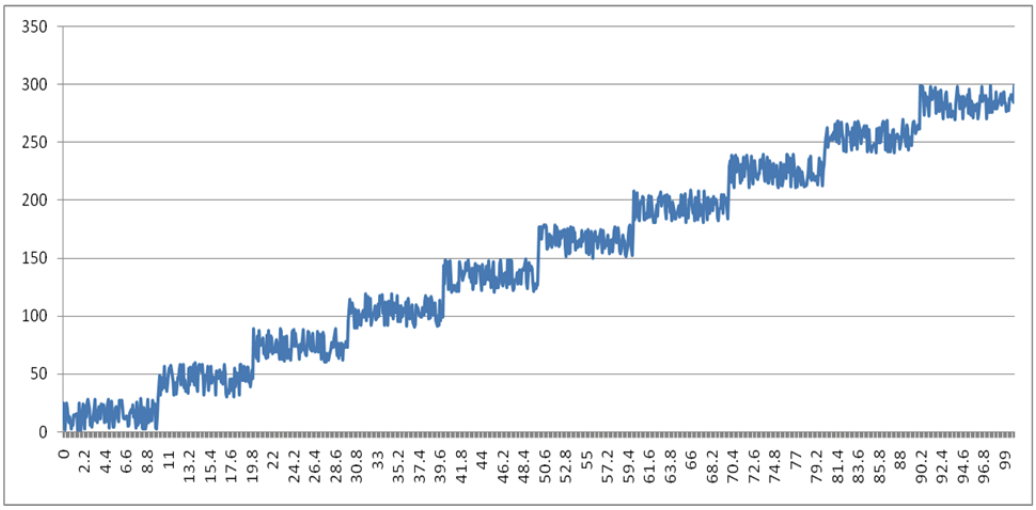


Fig. 4. Example of the data group based transformation supporting k-anonymity.

3.4 Function Segment based Transformation against Data Count Attacks

We propose a function segment based transformation algorithm that can protect the encrypted data from the data count attacks. If the original data reside in the same segment, GOPES encrypts the data into the same value since a segment is an area being generated by dividing an interval of a periodic function. Thus, GOPES can modify the frequency of the encrypted data. For segment generation, we choose a periodic function $p(x)$ repeating its values in regular intervals w, where the interval w (called basic period) is a positive constant value. There are some considerations for selecting $p(x)$. First, we select an interval w having no common multiple with the size of the original data (T) so that we can improve the privacy of the encrypted data. Second, we select a periodic function such that the encryption values using the function are not skewed in a specific range of the original data. If the encryption values are skewed, an attacker can easily estimate the transformation function. Finally, we select the periodic function whose pattern is not simple to inhibit attackers from estimating the function.

Based on three considerations, we choose a trigonometric function $p(x)$ as shown in Eq. (4), where the range of $p(x)$ is $-1 \le p(x) \le 1$ and the interval of $p(x)$ is a the least common multiple of k and l. Eq. (5) shows the segment generation function being calculated by differentiating Eq. (4). For example, if $a = b = 1$, $k = 2$, and $l = 1/2$ in Eq. (4), we can generate segments as shown in Table 1. For two original values $x_1 = 2.75$ and $x_2 = 3.64$, their encrypted values are the same because they reside in the same segment.

$$Periodic(x) = \frac{1}{a+b} \times (a \times sin(k \times x) \times b \times cos(l \times x)) \tag{4}$$

$$\{x|0 \le x \le w\} = \bigcup_{i=0}^{n}\{y|y \in Group_i\}(where\ x \notin R)$$
$$Segment = \begin{cases} \{x|0 \le x < x_0, where\ 0 < x_0 < x_1, \\ \{x|x_i \le x < x_{i+1}, where\ x_{i+1} < x_n, \\ \{x|x_i \le x < x_n, otherwise \end{cases} \tag{5}$$
$$(where\ x_i|0 \le x_i < w\}, Periodic'(x_i) = 0, 0 \le i \le n)$$

Table 1. Example of generating segments using periodic function

ID	Range of segments
1	$0 \leq x < 0.74$
2	$0.74 \leq x < 2.475$
3	$2.475 \leq x < 3.808$
4	$3.808 \leq x < 5.543$
5	$5.543 \leq x < 7.119$
6	$7.119 \leq x < 8.526$
7	$8.526 \leq x < 10.32$
8	$10.324 \leq x < 11.730$
9	$11.730 \leq x < 4\pi$

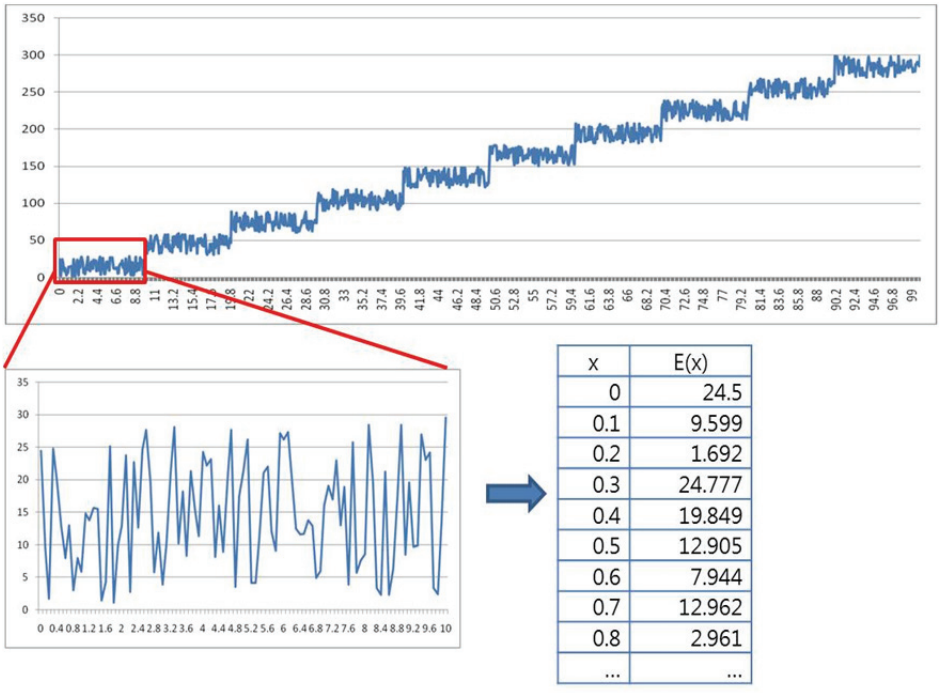


Fig. 5. Example of periodic function based data transformation.

3.5 Periodic Function based Transformation for Exact Matching Query

To improve the performance of the exact matching query, we propose a periodic function based transformation algorithm by using the Eq. (4). By analyzing the data to be encrypted, we select a periodic function, i.e., E, such that $E(x_1) \neq E(x_2)$ when $x_1 \neq x_2$. Here, it is impossible that two different original keys are transformed into the same encrypted key. This is because the original key is a rational number, while the value of the periodic function is an irrational number. Fig. 5 shows an example of data transformation based on the periodic function and the data group. Here, the range of the original value x is $0 \leq x \leq 100$. In Fig. 5, two original values $x_1 = 0.5$ and $x_2 = 0.7$, which are in the same data group, have the same random values by using our group transformation. However, their encryption

values based on our function transformation are 0.905 and 0.962, respectively. As a result, x_1 encrypted to 12.905 while x_2 is encrypted into 12.962. So our GOPES can reduce the communication overhead, which is required to send the result of the exact matching query, by filtering out the encrypted data based on the function transformation value. In addition, because our GOPES uses the tri shuffles the order of the encrypted data, it can reduce a probability that the original data can be disclosed from the order matching attack. For example we use the periodic function $E(x) = \frac{1}{2} \times (\sin(2x) + \cos(\frac{x}{2}))$, the order of the encrypted data can be reversed. If and the original values $x_1 < x_2$, then $E(x_1) > E(x_2)$ in the second segment.

3.6 Signature Generation for Efficient Query Processing

We propose a signature generation algorithm to improve the query processing performance. Our signature generation algorithm has three parts. (i) It divides the range of group transformation, segment transformation, and function transformation values into 2^n, 2^m, and 2^l number of equal area. For each area, the algorithm sets the signature value which is presented as bits. (ii) The encrypted values are changed into the signatures based on their area. (iii) It generates the encrypted signatures by concatenating group transformation, segment transformation, and function transformation values. Because the encrypted signature is a set of bits, our GOPES can reduce the computational cost for processing queries by using bit operations. Fig. 6 shows an example of generating the encrypted signature. In this example, we use Eq. (6) for the periodic function. And we set n, m, and l, which are the length of signature bits for transformation values, to 8, 4 and 4, respectively.

$$p(x) = \frac{1}{2} \times (\sin(2x) + \cos\left(\frac{x}{2}\right)) \tag{6}$$

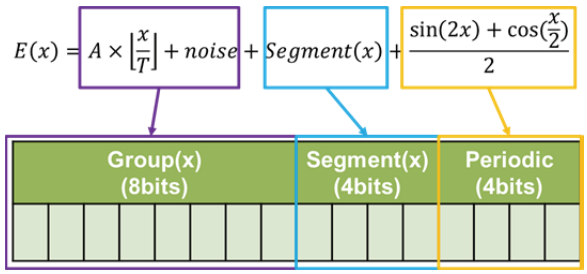


Fig. 6. An example of generating an encrypted signature.

For the segment transformation values, we can make segments by differentiating the Eq. (6) and set the signature values for the segments as shown in Fig. 7(a). For the function transformation values, we divide the signature areas and set their signature values as shown in Fig. 7(b). For example, assume that the transformation values {*Group*(x), *Segment*(x), *Periodic*(x)} for the original data x are 324, 321, and 0.654, respectively. If the size of group for the original data is 70, the value 324 is included in the fifth data group. So we can set the signature of the group as 00000101 which means 5. And by using the Fig. 7, we can set the signature of 321 to 0011 while setting that of 0.654 to 1101. So by concatenating all signatures, the final encrypted signature for x is 0000010100111101.

ID	Range of segments	Transformation value	Signature
1	$0 \leq x < 0.740$	000~099	0000
2	$0.740 \leq x < 2.475$	100~199	0001
3	$2.475 \leq x < 3.808$	200~299	0010
4	$3.808 \leq x < 5.543$	300~399	0011
5	$5.543 \leq x < 7.119$	400~499	0100
6	$7.119 \leq x < 8.526$	500~599	0101
7	$8.526 \leq x < 10.324$	600~699	0110
8	$10.324 \leq x < 11.730$	700~799	0111
9	$11.730 \leq x < 4\pi$	800~899	1000

(a)

ID	Range of segments	Signature	ID	Range of segments	Signature
1	$-1.000 \leq x < -0.875$	0000	9	$0.000 \leq x < 0.125$	1000
2	$-0.875 \leq x < -0.750$	0001	10	$0.125 \leq x < 0.250$	1001
3	$-0.750 \leq x < -0.625$	0010	11	$0.250 \leq x < 0.375$	1010
4	$-0.625 \leq x < -0.500$	0011	12	$0.375 \leq x < 0.500$	1011
5	$-0.500 \leq x < -0.375$	0100	13	$0.500 \leq x < 0.625$	1100
6	$-0.375 \leq x < -0.250$	0101	14	$0.625 \leq x < 0.750$	1101
7	$-0.250 \leq x < -0.125$	0110	15	$0.750 \leq x < 0.875$	1110
8	$-0.125 \leq x < 0.000$	0111	16	$0.875 \leq x < 1.000$	1111

(b)

Fig. 7. Signature generation using the selected periodic function in Eq. (6). (a) Segments and its signatures for the segment transformation. (b) Signatures and their ranges for the function transformation.

4. Experimental Analysis

In this section, we present the performance comparison of our GOPES with the existing POPIS [9], with respect to the degree of data privacy and query processing time. We implement our GOPES by using the periodic function shown in Eq. (6). For the encrypted query processing, all the datasets containing {key, value} pairs have to be encrypted. Here, in order to enforce the data security for keys and values, we use two different types of data encryption schemes, i.e., order-preserving encryption scheme (our GOPES) and Advanced Encryption Standard (AES) [10], respectively. The flow of the encrypted query processing is as follows. At first, a date owner encrypts the whole data with two schemes, i.e., keys with GOPES and values with AES-256. Then the data owner sends the encrypted {key, value} pairs to a service provider. Second, an authenticated user encrypts his/her query and issues it to the service provider. Third, the service provider processes the encrypted query without any decryption. Then the service provider returns the candidate result set to the query issuer. Finally, the user decrypts the received candidates and obtains the final result. For performance analysis, an exact matching query and a range query are used. We generate a dataset using the US Census data [11], containing 2 GB. Since our dataset includes sensitive attributes, like age, incomes, and jobs, we encrypt

the data of sensitive attributes by using the POPIS and our GOPES. We also use 100 queries being randomly generated from our dataset. Table 2 shows our experimental environment.

Table 2. Experimental environment

Item	Spec
CPU	Intel Core2 Quad, Q6600 2.4 GHz
Memory	2 GB
OS	Windows 7 Enterprise K
Compiler	Microsoft Visual Studio 2010

4.1 Experimental Results of Data Privacy against Order Matching Attacks

To evaluate data privacy against order matching attacks, we use two encryption functions as shown in Fig. 8. Here, the size of data group in each encryption function is 10. The encryption function for OPES have a cycle of 4π and the size of encryption group is 10. Thus, if $\min(g_i) < \min(p_j) < \min(g_{i+1})$ $(i \neq j)$ for a data group g_i and a cycle period p_j, the order of the encrypted data is not preserved at the same rate as $(4\,\pi * \mathrm{j} - 10 * \mathrm{i})/10$.

$$\begin{cases} 8x + 1.00 & ,where\ x \le 10.0 \\ 1.88x + 62.2 & ,where\ x \le 20.0 \\ 3.45x + 30.8 & ,where\ x \le 30.0 \\ 10.08x - 168.10 & ,where\ x \le 40.0 \\ 9.92x - 161.70 & ,where\ x \le 50.0 \\ 4.05x + 131.8 & ,where\ x \le 60.0 \\ 8.46x - 132.8 & ,where\ x \le 70.0 \\ 10.53x - 277.7 & ,where\ x \le 80.0 \\ 2.37x + 375.1 & ,where\ x \le 90.0 \\ 5.5x + 93.4 & ,\ where\ x \le 100.0 \end{cases}$$

(a)

$$E_1(x) = Sig(A \times \left\lfloor \frac{x}{10} \right\rfloor + random_{noise}$$

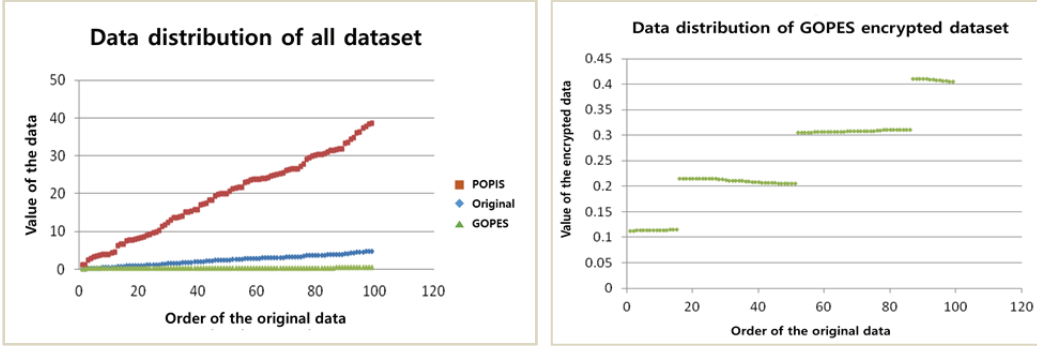
$$E_2(x) = Sig\left(Group\left(\frac{1}{2}\left(\sin 2x + \cos\frac{x}{2}\right)'\right)\right)$$

$$E_3(x) = Sig\left(Group\left(\frac{1}{2}\left(\sin 2x + \cos\frac{x}{2}\right)\right)\right)$$

$$E_4(x) = concatenate(E_1(x), E_2(x), E_3(x))$$

(b)

Fig. 8. Two encryption functions used in our experiment. (a) Encryption function for POPIS. (b) Encryption function for our GOPES.



(a) (b)

Fig. 9. Data distributions $(0 \le x \le 100)$. (a) Original and encrypted datasets. (b) Encrypted dataset using GOPES.

When we use two encryption functions in Fig. 8, Fig. 9(a) shows the data distributions of the original data, the encrypted data using POPIS, and the encrypted one using our GOPES. Specially, Fig. 9(b) shows the detailed data distribution of the encrypted data using our GOPES. Here, the range of the original dataset is $0 \leq x \leq 100$.

For our experiment, we measure the probability of data exposure against the order matching attack as follows.

1. An attacker obtains all the data of a specific column, containing sensitive data in the original database. An attacker also obtains all the encrypted data corresponding to all the data of the specific column.
2. Assuming that the encryption function is $E(x)$, the ith original data is Ori_i and the ith encrypted data is Enc_i. We sort both the original data and the encrypted data and make a pair of $<\text{Ori}_i, \text{Enc}_i>$ based on their sorted order.
3. We count the number of pairs being correctly matched. A correctly matched pair is a pair of $<\text{Ori}_i, \text{Enc}_i>$ that satisfies condition $E(\text{Ori}_i)=\text{Enc}_i$. By using the correctly matched pairs, an attacker can estimate $E(x)$. As the number of the correctly matched pairs is increased, the similarity between the original encryption function and the estimated one is increased.
4. For all correctly matched pairs, we measure the number of the indistinguishable pairs that satisfy both $\text{Ori}_i \neq \text{Ori}_j$ and $\text{Enc}_i \neq \text{Enc}_j$ where $i \neq j$.
5. We finally calculate the probability of data exposure against the order matching attack by using both the number of the correctly matched pairs and that of the indistinguishable pairs.

For example, there are two original data A and B that have the same encryption value $E(A)$. In this case, the attacker cannot clearly make a decision on whether $E(A)$ is generated by using the original data A or not. As a result, the probability of data exposure is decreased while the number of the indistinguishable pairs increases. Table 3 shows the result of the order matching attack. Since the order of the encrypted data in POPIS is the same as that of the original data, the ratio of the number of the correctly matched pairs over the total number of pairs is 100% and the average number of the indistinguishable data pairs is 0. Thus, POPIS is weak to the order matching attack because the probability of data exposure against the order matching attack is 100%. However, in our GOPES, the ratio of the number of the correctly matched pairs over the total number of pairs is reduced to 31.16%. This is because our GOPES preserves the order of each data group, but not each data, by using the group transformation. In addition, our GOPES shuffles the order of the encrypted data based on the function transformation. Since our GOPES sets the same encryption value for the original data in the same segment by using our segment transformation, the average number of the indistinguishable data pairs is increased to 2.56. Since our GOPES shows the 12.18% probability of data exposure against the order matching attack, our GOPES outperforms the POPIS in terms of the data privacy against the order matching attack.

Table 3. Probability of data exposure against order matching attack

Encryption scheme	Ratio of matched pairs over total number of pairs (%)	Average number of the indistinguishable pairs	Probability of data exposure against order matching attack (%)
POPIS	100	0	100
GOPES	31.16	2.56	12.18

4.2 Experimental Results of Data Privacy against Data Count Attacks

To evaluate data privacy against data count attacks, if an attacker can infer the range of the encrypted data for an original data x, we count the number of the encrypted data falling into the range as the exposure of the original data. Using data counting, Fig. 10 shows the data frequency of the original data, the encrypted data using POPIS, and the encrypted one using our GOPES. In Fig. 10, it is shown that the frequency of the original data is very similar to that of the encrypted data using POPIS. To analyze the similarity between the frequency of the original data and that of the encrypted data, we measure the t-test by using Eq. (7). Here, x_i means the average value of dataset$_i$, s_i means the standard deviation of dataset$_i$, and n_i means the number of members in dataset$_i$. Table 4 shows the t-test results of not only the similarity between the frequency of the original data and that of the encrypted data using POPIS, but also the similarity between the frequency of the original data and that of the encrypted data using GOPES. Here, we set the level of significance to 0.05.

$$t = \frac{(\overline{X_1} - \overline{X_2})}{S_{(\overline{X_1} - \overline{X_2})}} (where\ S_{(\overline{X_1} - \overline{X_2})} = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}) \tag{7}$$

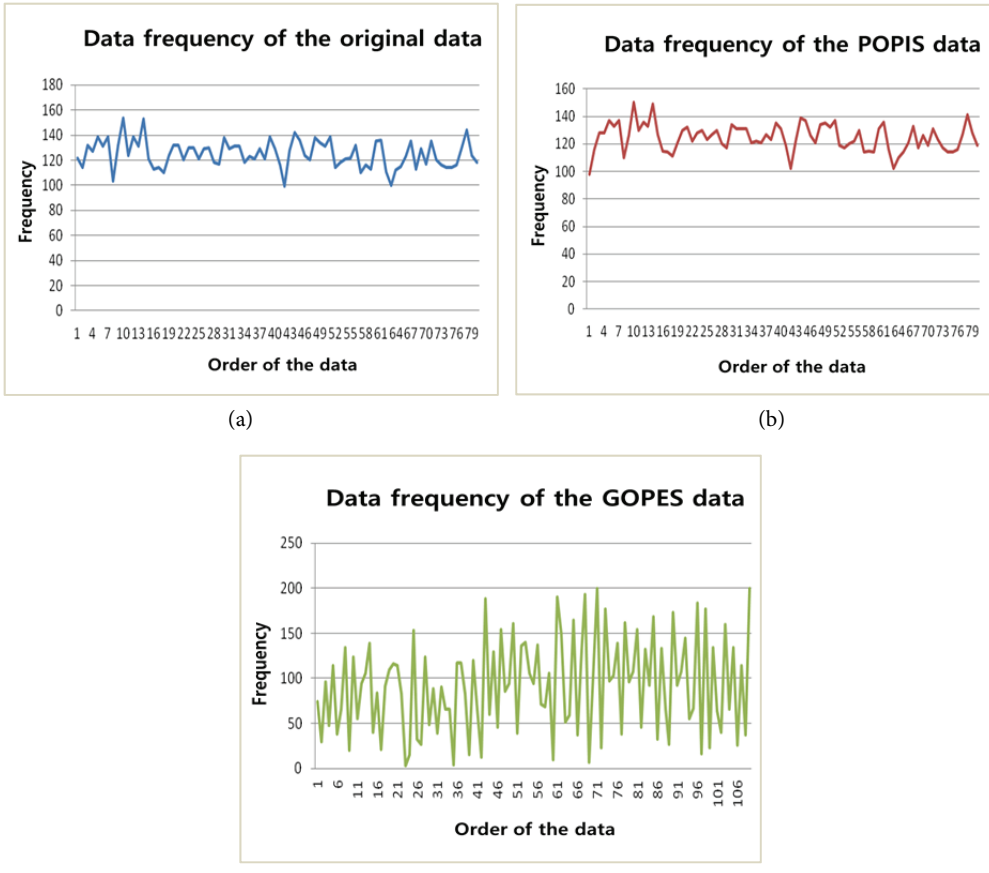


Fig. 10. Data frequency of dataset. (a) Frequency of the original dataset. (b) Encrypted data using POPIS. (c) Encrypted dataset using GOPES.

If t is greater than the p-value, the frequency of the encrypted data is different from that of the original data. Otherwise, the frequency of the encrypted data is the same as that of the original data. Since the t statistic and the p-value are 0.19 and 0.85 in POPIS, the data frequency of the encrypted data using POPIS is the same as that of the original data. However, since the t statistic and the p-value are 6.441 and 2.52×10^{-8} in GOPES, the frequency of the encrypted data using GOPES is different from that of the original data. This is because our GOPES generates the same encryption value for the original data in the same segment, by using our segment transformation algorithm. As a result, our GOPES outperforms the existing POPIS in terms of the data privacy against the data count attack.

Table 4. T-test result of the similarity between the frequency of the original data and those of two encrypted data

Encryption scheme	Comparator	*t* statistics	*p*-value
POPIS	Original data	0.19	0.85
GOPES	Original data	6.441	2.52×10-8

4.3 Experimental Results in Terms of the Query Processing Time

Fig. 11 shows the encrypted query processing times of both POPIS and GOPES. In case of the exact matching query, our GOPES requires 0.036 seconds for the query processing time while POPIS requires 0.037 seconds. In case of the range query, our GOPES requires 0.045 seconds for the query processing time while POPIS requires 0.048 seconds. There are two reasons why GOPES outperforms POPIS. First, GOPES can reduce the size of the candidate set to be transmitted by using both the periodic function and periodic interval values, while POPIS requires a large candidate set to be transmitted as the range of the noise is increased for high data protection. Second, GOPES can reduce the search time of the encrypted key by using bit operations, while POPIS requires long key search time by using numeric operations.

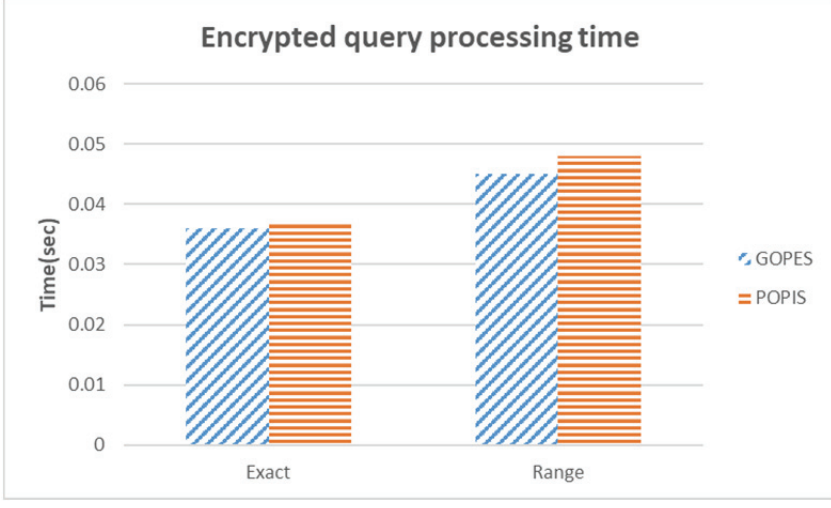


Fig. 11. Query processing time over the encrypted data.

5. Conclusions

To support advanced user-customized services in cloud computing, companies build the analyzed data of users by using outsourced databases. But, the data of users contain sensitive personal information that may infringe the privacy of users. To protect the sensitive data, it is required to encrypt the database prior to outsourcing it to the service provider. However, it is inefficient to perform query processing on the encrypted data by using the existing data encryption schemes. To overcome this limitation, we proposed data encryption scheme, called GOPES, which can support efficient query processing over encrypted data. Since our GOPES can preserve the order of each data group by generating the signatures of the encrypted data, it can protect data from both order matching attacks and counting attacks. We also showed that our GOPES achieved a higher degree of data privacy protection than the existing POPIS.

Acknowledgement

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. R0113-17-0005, Development of an Unified Data Engineering Technology for Large-scale Transaction Processing and Real-time Complex Analytics) and this work was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2016R1D1A3B03935298).

References

[1] H. Hacigumus, S. Mehrotra, and B. Iyer, "Providing database as a service," in *Proceedings 18th International Conference on Data Engineering,* San Jose, CA, 2002, pp. 29-38.

[2] C. Curino, E. P. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich, "Relational cloud: a database-as-a-service for the cloud," in *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, 2011, pp. 235-240.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009.

[4] Y. Zhu, G. J. Ahn, H. Hu, S. S. Yau, H. G. An, and C. J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227-238, 2013.

[5] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, Paris, France, 2004, pp. 563-574.

[6] L. Xiao and I. L. Yen, "Security analysis for order preserving encryption schemes," in *Proceedings of 2012 46th Annual Conference on Information Sciences and Systems (CISS),* Princeton, NJ, 2012, pp. 1-6.

[7] L. Xiao, I. L. Yen, and D. T. Huynh, "Extending order preserving encryption for multi-user systems," 2012; https://eprint.iacr.org/2012/192.pdf.

[8] D. H. Yum, D. S. Kim, J. S. Kim, P. J. Lee, and S. J. Hong, "Order-preserving encryption for non-uniformly distributed plaintexts," in *Information Security Applications*. Heidelberg: Springer, 2011, pp. 84-97.

[9] D. Liu and S. Wang, "Programmable order-preserving secure index for encrypted database query," in *Proceedings of 2012 IEEE 5th International Conference on Cloud Computing*, Honolulu, HI, 2012, pp. 502-509.

[10] National Institute of Standards and Technology, "FIPS 197: Announcing the advanced encryption standard (AES)," 2001; https://csrc.nist.gov/publications/detail/fips/197/final.

[11] US Census Bureau [Online]. Available: https://www.census.gov/.

Hyunjo Lee https://orcid.org/0000-0002-1316-6822

He is a researcher and adjunct instructor in the Chonbuk National University. He received the B.S., M.S., and Ph.D, degrees in Computer Engineering from Chonbuk National University in 2006, 2008, and 2014, respectively. His research interests include spatial network database, parallel query processing, and encrypted query processing

Youngho Song https://orcid.org/0000-0002-0757-0802

He is a PhD course in the Chonbuk National University. He received the B.S. and M.S. degrees in Chonbuk National University in 2014 and 2016, respectively. His research interests include big data analysis, distributed parallel processing, and encrypted query processing

Jae-Woo Chang https://orcid.org/0000-0002-0037-6812

He is a professor in the Department of Information and Technology, Chonbuk National University, Korea from 1991. He received the B.S. degrees in Computer Engineering from Seoul National University in 1984. He received the M.S. and Ph.D. degrees in Computer Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1986 and 1991, respectively. During 1996–1997, he stayed in University of Minnesota for visiting scholar. And during 2003–2004, he worked for Penn State University (PSU) as a visiting professor. His research interests include sensor networks, spatial network database, context awareness and storage system.