Private Range Query by Perturbation and Matrix Based Encryption

Junpei Kawamoto, Masatoshi Yoshikawa
Graduate School of Informatics, Kyoto University
Kyoto, Japan 606-8501
Email: j.kawamoto@db.soc.i.kyoto-u.ac.jp, yoshikawa@i.kyoto-u.ac.jp

***Abstract*—In this paper, we propose a novel approach for private query; IPP (inner product predicate) method. Private query is a query processing protocol to obtain requesting tuples without exposing any information about what users request to third persons including service providers. Existing works about private query such as PIR, which ensure information theoretic safety, have severe restriction because they do not support range queries nor allow tuples having a same value in queried attributes. Our IPP method, on the other hands, focuses range queries mainly and it allows tuples having a same value in any attributes. IPP method employs a query transform by trusted clients (QT) scheme and proposes transformation algorithms which make the correlation between plain queries and transformed queries and the correlation between plain attribute values and transformed attribute values small enough. Thus, the transformed queries and attribute values have resistance to frequency analysis attacks which implies IPP method prevents attackers, who know the plain distribution of them, from computing the plain queries and attribute values from transformed values. IPP method adds perturbations to queries and attribute values and gives them a matrix based encryption to achieve the above property. We also confirm the computational cost on servers belongs to $O(n)$ with the number of tuples n and is virtually no correlation between the distributions of transformed queries and queried attribute values and the plain distributions of them by experimental evaluations.**

I. Introduction

In Database as a Service (DaaS), security about data stored in servers and privacy of users are two of the most important topics discussed for a long time [1], [2]. Among them, there is a requirement of users to obtain data without exposing any information about what the users request to third persons including service providers. This requirement is called *private query*. One of the famous researches about private query is known as PIR (Private Information Retrieval) [3]. In PIR, a database server has n bits and the goal is to assure that users are able to know the i-th bit value with keeping any information about the requesting i information theoretical secret.

However, this PIR does not allow two things which are important to real applications; *range queries* and *multi key*. In other words, most existing PIR work considers exact match query and each index i associates with only one tuple. In this paper, we propose a novel approach for private query, which is named IPP (inner-products predicate) method. It allows range queries and tuples to have a same attribute value. Our IPP method assumes that databases are constructed from two attributes Key and $Value$. Users request queries over the only Key attribute. The domain of key attribute $\mathbb{D}_K$ is l_K bit natural numbers. IPP method assumes database servers have tuples which have the same Key attribute values. Queries of IPP method is range queries i.e. users request tuples of which Key attribute values are in a closed range $[a, b] (a \leq b \in \mathbb{D}_K)$. In the rest of this paper, we only discuss about the range queries but exact match queries. However, exact match queries are special cases of range queries ($a = b$). Therefore, any discussions about range queries are applicable for exact match queries.

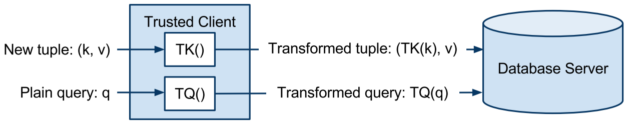

Fig. 1. The QT scheme which IPP method employes.

To achieve private query, IPP method employes a query transform by trusted client (QT) scheme. Fig. 1 shows this scheme. When an user adds a tuple (k, v), of which the Key attribute value is k and the $Value$ attribute value is v, to a database server, the trusted client of the user transforms k to the secure Key attribute value TK(k) using an algorithm TK. Then, the client sends the transformed tuple (TK$(k), v$) to the server. When an user sends a plain query q to a database server, the trusted client of the user transforms the plain query q to the secure query TQ(q) using an algorithm TQ. Then, the client sends the transformed query to the server. Note that those algorithms TK and TQ were shared by all trusted clients of users at the beginning.

The attack model which we consider in this paper is a frequency analysis for the transformed queries and the transformed Key attribute values. It means that the propose of attackers is to obtain how many times each transformed query is requested and how many tuples have a same transformed Key attribute value. The frequency analysis enables attackers who know the distributions of plain queries and Key attribute values to compute the original values from transformed values. On this QT scheme, IPP method proposes transformation algorithms which make the distributions of transformed queries and

This work was supported by Grant-in-Aid for JSPS Fellows(21-3998).

Key attribute values different from the original distributions of the plain queries and Key attribute values. To ensure this properties, IPP method adds perturbations to queries and Key attribute values and then encrypts them. Perturbations are random values added to prevent attackers from computing the plain values, which are the distributions of plain queries and Key attribute values. IPP method uses an encryption scheme based on matrix algebra. Therefore, queries and key attribute values are expressed by vectors and the query evaluation process on servers is mapped to inner products.

II. Related work

Among many kinds of PIR work [4], the most related work to our problem is cPIR (Computational Private Information Retrieval) [5] which assumes only one database server and ensures a private query based on complexity theory. As we discussed in section I, basic cPIR approach does not support range queries nor allow tuples have same Key attribute values. cPIR requires $O(n)$ computational cost on servers with the total number of tuples n. bbPIR [6] is a weak cPIR protocol which relaxes the security to reduce the computational cost on servers. It brings the idea of k-anonymity [7] in cPIR. Clients on bbPIR protocol sends a k-width bounding box containing the index which the clients want to request. After that, the clients and the server communicate with basic cPIR protocol on the sub-database consists of tuples in the bounding box. bbPIR is able to hide only which tuples in the k tuples the clients exactly request but the computational cost on servers is smaller than basic cPIR protocol.

For location-based services (LBS), there is a kind of private range query using cPIR protocol [8]. In LBS, users request some region to obtain points of interests (POIs) such as restaurants, gas stations, etc. On the approach in [8], the 2-dimensional space in the LBS is divided into n sub areas, in such a way that each sub area has at most one POI. Clients firstly compute the index of the sub area which contains the really requesting region and then the clients request the index using basic cPIR protocol. Therefore, this approach support a limited range query and users cannot request flexible regions.

Some work about encrypted database (EDB) [2] is also achieving weak private query. In the context of EDB, all tuples on servers are encrypted and the main interest is how to execute queries over the encrypted tuples. Therefore, queries also do not contain plain values and in this meaning, EDB is achieving a kind of private query. Bucketization described in [2], [9] splits the domain of each attribute into some labeled buckets and each attribute value is updated to the label of which the bucket contains the plain value by clients. This approach needs statistic information about the domains to make each bucket have almost same number of tuples, and re-builting buckets may be necessary after lots of new tuples are inserted. [10] points a problem that attackers can obtain some information about plain values by comparing the re-builted buckets and old buckets. Order Preserving Encryption Scheme (OPES) [11] provides range queries over EDB, i.e. weak private range query, and needs statistic information about domains. SCONEDB [12] provides a kind of private query using a matrix based encryption. The main purpose of SCONEDB is to achieve k-nearest neighbor (kNN) query over encrypted vector databases. SCONEDB provides a querying approach without decrypting encrypted vectors on servers. Those approaches about EDB are able to hide plain values from queries. However, in those approaches, queries which request a same range are transformed to a same query for EDB. It means those approach cannot protect the frequency analysis attack so that attackers, who know the distribution of plain queries, are able to compute the plain queries from queries for EDBs. On the other hands, our IPP method adds perturbations to queries in addition to matrix based encryption. As a result, IPP method makes transformed queries associated with a same plain query varied, so that it ensure to protect the frequency analysis of attackers. Additionaly, IPP method does not need any statistical information about domains so that IPP method is also suitable to growing databases.

III. Inner product predicat method

In this paper, we propose IPP method which is a new private range query method i.e. it enables users to request tuples without exposing their queries to third parties including servers. On the QT scheme described section I, IPP method ensures the distributions of transformed queries and transformed Key attribute values are different from the distribution of plain queries and plain Key attribute values. The assumptions for IPP method are

- the database is constructed from Key and $Value$ attributes,
- Key attribute has l_K bit natural number domain $\mathbb{D}_K$,
- users request tuples of which Key attribute values are in closed ranges $[a, b] (a, b \in \mathbb{D}_K)$.

IPP method starts from a predicate query model. In this model, clients send predicate functions p as queries to servers. Database servers, on the other hand, search tuples of which Key attribute value k satisfies $p(k) =$ true and then return the tuples to the clients. Our IPP method uses a kind of functions $p : \mathbb{D}_K \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers, as the predicate functions and regards $p(k) \leq 0$ as true so that $p(k) > 0$ means false. For example, there is a simple predicate function representing a range query $k \in [a, b]$:

$$p_{a,b}(k) = (k - a)(k - b). \tag{1}$$

This function obviously takes values below 0 if and only if k is in the closed range $[a, b]$. IPP method adds perturbations to this basic predicate $p_{a,b}$ and also adds them to each Key attribute value k. In addition, IPP method encrypts predicate functions and Key attribute values in order to make the distributions of transformed queries and Key attribute values different from ones of plain queries and plain Key attribute values.

A. Addition of perturbations

Perturbations for predicate functions are added to the kind of basic predicate function $p_{a,b}$ defined by (1). The perturbations

d are l_d bit natural numbers, where l_d is a given security parameter. The predicate function with the perturbation d is

$$p_{a,b,d}(k) = (k-a)(k-b)(k+d).$$

From the assumptions, k, a, b, and d are all natural numbers. Therefore, the above predicate function $p_{a,b,d} \leq 0$ if and only if $a \leq k \leq b$. In other words, the predicate function will be true for only tuples of which Key attribute value is in $[a, b]$, thus clients are able to obtain correct tuples.

The perturbations for Key attribute values are real numbers δ which satisfy $|\delta| < 1/2$. The perturbation-added value $\tilde{k}_\delta$ of a Key attribute value k is defined by $\tilde{k}_\delta = k + \delta$. To handle $\tilde{k}_\delta$ correctly, the predicate functions $p_{a,b,d}$ are also arranged as below;

$$\tilde{p}_{a,b,d}(k) = (k-a+\frac{1}{2})(k-b-\frac{1}{2})(k+d). \qquad (2)$$

This function is calculated with $\tilde{k}_\delta$ as follows;

$$\tilde{p}_{a,b,d}(\tilde{k}_\delta) = (k+\delta-a+\frac{1}{2})(k+\delta-b-\frac{1}{2})(k+\delta+d).$$

We assumed k, a, b, and d are all natural naumbers and $|\delta| < 1/2$, so that the above predicate function become true i.e. $\tilde{p}_{a,b,d}(\tilde{k}_\delta) \leq 0$ if and only if $a \leq k \leq b$. It means query processing by perturbation-added predicate functions and perturbation-added Key attribute value will be calculated correctly.

B. Encryption of queries and key attributes

IPP method uses an encryption scheme for plain Key attribute values k and querying ranges $[a, b]$ to prevent attackers from obtaining them from perturbation-added Key attribute values $\tilde{k}_\delta$ and predicate functions $\tilde{p}_{a,b,d}$. The encryption scheme is based on matrix operations and allows servers to handle the encrypted queries without decrypting them. In order to apply this encryption scheme, we first express predicate functions and Key attribute values as vectors.

Considering the perturbation-added predicate defined by (2), we define the predicate vectors $\tilde{\mathbf{p}}_{a,b,d}$ and key vectors $\tilde{\mathbf{k}}_\delta$ as the followings;

$$\tilde{\mathbf{p}}_{a,b,d} = \begin{pmatrix} 1 \\ -(a+b-d) \\ (a-\frac{1}{2})(b+\frac{1}{2}) - (a+b)d \\ (a-\frac{1}{2})(b+\frac{1}{2})d \end{pmatrix}, \qquad (3)$$

$$\tilde{\mathbf{k}}_\delta = (\tilde{k}_\delta^3, \tilde{k}_\delta^2, \tilde{k}_\delta^1, 1)^t = \big((k+\delta)^3, (k+\delta)^2, (k+\delta), 1\big)^t \quad (4)$$

Using those vectors, the query processing on servers is to find tuples of which the key vector satisfies $<\tilde{\mathbf{p}}_{a,b,d}, \tilde{\mathbf{k}}_\delta> \leq 0$. This inner product is calculated by

$$<\tilde{\mathbf{p}}_{a,b,d}, \tilde{\mathbf{k}}_\delta> = (k+\delta-a+\frac{1}{2})(k+\delta-b-\frac{1}{2})(k+\delta+d). \quad (5)$$

Thus, the condition $<\tilde{\mathbf{p}}_{a,b,d}, \tilde{\mathbf{k}}_\delta> \leq 0$ is same as $(k+\delta-a+1/2)(k+\delta-b-1/2)(k+\delta+d) \leq 0$. Because $|\delta| < 1/2$, this condition is true if the key attribute value k is in the closed range $[a, b]$.

In the encryption scheme used IPP method, the encryption key for predicate vectors is a four-dimensional regular-integer matrix M i.e. M has an inversed matrix M^{-1} and every elements are integer. The encryption key for key vectors is $D_M = |\det(M)|M^{-1}$, where $\det(M)$ means the determinant of M. Because of the Cramer's formula, this matrix D_M is also four-dimensional regular-integer matrix. Using these matricies, we define the encryption procedure for predicate vactors $\mathrm{EP}_M(\mathbf{p})$ and for key vectors $\mathrm{EK}_{D_M}(\mathbf{k})$ as followings;

$$\mathrm{EP}_M(\mathbf{p}) = r_p M^t \mathbf{p}, \ \mathrm{EK}_{D_M}(\mathbf{k}) = r_k D_M \mathbf{k},$$

where r_p and r_k are random natural numbers and they are set different values each encryption time.

The query processing for those encrypted vectors is same as for plain vectors and it is to find tuples of which the encrypted key vector $\mathrm{EK}_{D_M}(k)$ satisfies $<\mathrm{EP}_M(\tilde{\mathbf{p}}_{a,b,d}), \mathrm{EK}_{D_M}(\tilde{\mathbf{k}}_\delta)> \leq 0$. This inner product is computed as

$$<\mathrm{EP}_M(\tilde{\mathbf{p}}_{a,b,d}), \mathrm{EK}_{D_M}(\tilde{\mathbf{k}}_\delta)> = r_p r_k |\det(M)| <\tilde{\mathbf{p}}_{a,b,d}, \tilde{\mathbf{k}}_\delta>.$$

Because $r_p r_k |\det(M)| > 0$, the above inner product will be below 0 if and only if $a \leq k \leq b$ as we discussed about (5).Thus, query processing for encrypted query vectors and key vectors is handled in the way of anticipation without decrypting them.

C. Integeration of vector elements

Each element of the predicate vectors and key vectors defined by (3) and (4), respectively, is not an integer but a real number. Therefore, servers compute the inner product as operations for floating-point numbers and those operations have rounding errors. As a result, the responses of queries may have errors. In order to proof no results have errors, IPP method makes the elements of predicate vectors and key vectors integer and ensures the computation of servers has no errors.

IPP method uses a linear approximate equation to make elements of vectors integer. This approximation presumes

$$f(k+\delta) \simeq f(k) + \delta \frac{d}{dk} f(k) \qquad (6)$$

for any polynomials $f(x)$ in a condition δ/k is small enough. In this section, we also proof the approximation does not bring any errors to the computations of the inner product.

The greatest common divisor of the denominators of the plain predicate vectors defined by (3) is 4. Therefore, to make the elements of predicate vectors integer, multiplying 4 to all elements is enough and the integer predicate vector $\hat{\mathbf{p}}_{a,b,d}$ is defined as follows;

$$\hat{\mathbf{p}}_{a,b,d} = 4\tilde{\mathbf{p}}_{a,b,d} = \begin{pmatrix} 4 \\ -4(a+b-d) \\ (2a-1)(2b+1) - 4(a+b)d \\ (2a-1)(2b+1)d \end{pmatrix}. \qquad (7)$$

Since we assumed a, b, and d are integer, all elements of $\hat{\mathbf{p}}_{a,b,d}$ are integer.

To make the elements of key vectors integer, we assume δ/k is small enough in the plain key vector defined by (4). In this assumption, we are able to apply the linear approximate equation (6) to each element of key vectors;

$$\tilde{\mathbf{k}}_\delta \simeq (k^3 + 3\delta k^2, k^2 + 2\delta k, k + \delta, 1)^t.$$

Let us ϕ be a natural number and $\delta = 1/\phi$, where ϕ is big enough to satisfy $|\delta| = |1/\phi| < 1/2$. Since k is a natural number, the greatest common divisor of the denominators of the approximated key vector is ϕ. Therefore, we define the integer key vector $\hat{\mathbf{k}}_{1/\phi}$ by multiplying ϕ to the all elements;

$$\begin{aligned}\hat{\mathbf{k}}_{1/\phi} &= \phi(k^3 + 3\frac{1}{\phi}k^2, k^2 + 2\frac{1}{\phi}k, k + \frac{1}{\phi}, 1)^t \\ &= (\phi k^3 + 3k^2, \phi k^2 + 2k, \phi k + 1, \phi)^t. \quad (8)\end{aligned}$$

Since k and ϕ are integer, the all elements of $\hat{\mathbf{k}}_{1/\phi}$ are integer.

IPP method uses a linear approximate equation to make the elements of key vector integer, therefore the integer key vectors have errors. We proof those errors do not bring any effects for the computation on servers. Let us think a function $g(k) = <\hat{\mathbf{k}}_{1/\phi}, \hat{\mathbf{p}}_{a,b,d}>$ and the following four conditions;

$$g(a) \leq 0,\ g(b) \leq 0,\ g(a-1) > 0,\ g(b+1) > 0.$$

If the above four conditions are satisfied, servers will handle queries expectingly. Indeed, the four conditions are satisfied with the assumptions; $0 < a \leq b$, $0 < d$, and $0 < \phi$. Therefore, the linear approximate equation does not bring any negative effects for the query processing of servers.

D. Distributions of vector elements

The property of IPP method is to make the distributions of transformed queries and Key attribute values different from the distributions of plain queries and Key attribute values. In this section, we argue the good way of choosing perturbations.

Considering the definition (7), the distributions of the elements of the transformed predicate vector $\mathrm{EP}_M(\hat{\mathbf{p}}_{a,b,d})$ are dependent on abd. If we choose perturbations d from a uniform random number generator R_u, the distribution of abd is dependent on the distributions of a and b. It means the distribution of transformed queries is dependent on the ones of plain queries. Instead of R_u, we think to use an alternative random number generator

$$R_d(a, b) = \lfloor \frac{R_u}{ab} \rfloor. \quad (9)$$

Choosing perturbation from this generator, abd is simplified by

$$abR_d(a, b) = ab\lfloor \frac{R_u}{ab} \rfloor \simeq R_u.$$

It means we can presume that the distribution of abd is dependent on R_u and not dependent on the querying range a and b. In other words, the distribution of queries is according to a uniform random distribution.

Let us think about the transformed key vectors $\mathrm{EK}_{D_M}(\hat{\mathbf{k}}_{1/\phi})$ and (8). We can presume that each element of the key vectors is dependent on ϕk^3. Therefore, if we choose the perturbations ϕ from the uniform random generator R_u, the distribution of ϕk^3 is dependent on the distribution of key attribute values k strongly. To avoid the fact, we use an alternative random generator

$$R_{\phi,\alpha}(k) = \lfloor \frac{R_u}{k^\alpha} \rfloor. \quad (10)$$

Choosing perturbations ϕ from the above generator, $R_{\phi,\alpha}(k)k^3$ is computed as

$$R_{\phi,\alpha}(k)k^3 = \lfloor \frac{R_u}{k^\alpha} \rfloor k^3 \simeq k^{3-\alpha} R_u.$$

The parameter α controls how much the distribution of the transformed key vectors is dependent on the distribution of key attribute values k. For example, when $\alpha = 3$, we can presume that the distribution of key vectors is not dependent on the distribution of key attribute values and according to a uniform random distribution.

IV. Scheme of the IPP method

In this section, we summarize the scheme of IPP method. Let us think to apply IPP method to a database server which satisfies the assumptions of IPP method. At the first step, administrators of this database generate an encryption key pair (M, D_M). M is an encryption key for predicate vectors and a four-dimensional regular-integer matrix. We assume it consists of 16 l_m bit random integers. D_M is an encryption key for key vectors and calculated from $D_M = |\det(M)|M^{-1}$. Because of the Cramer's formula, each element of D_M is a cofactor of M and a determinant of three-dimensional matrix. It means each element is a $3l_m$ bit integer. The administrators share the key pair to all trusted clients of users. When an user adds a new tuple (k, v) which consists of Key attribute value k and $Value$ attribute value v to the database, the trusted client of the user transforms the Key attribute value k by the algorithm $\mathrm{TK}_{D_M,R_\phi}(k)$:

```
Require: $D_M$ is a encryption key, $R_\phi$ is a random generator
  Generate a $l_\phi$ bit random natural number $\phi$ by $R_\phi(k)$
  Compute the key vector $\hat{\mathbf{k}}_{1/\phi}$ defined by (8)
  Generate a $l_{rk}$ bit random natural number $r_k$
  Encrypt the key vector $\mathrm{EK}_{D_M}(\hat{\mathbf{k}}_{1/\phi})$ (= $r_k D_M^t \hat{\mathbf{p}}_{a,b,d}$)
  return $\mathrm{EK}_{D_M}(\hat{\mathbf{k}}_{1/\phi})$
```

Then, the client makes secure tuple $(\mathrm{TK}_{D_M,R_\phi}(k), v)$ and sends it to the database server.

When an user asks tuples of which Key attribute values are in a closed range $[a, b]$, the trusted client of the user transforms the querying range to a secure query by the algorithm $\mathrm{TQ}_{M,R_d}(a, b)$:

```
Require: $M$ is a encryption key, $R_d$ is a random generator
  Generate a $l_d$ bit random natural number $d$ by $R_d(a, b)$
  Compute the predicate vector $\hat{\mathbf{p}}_{a,b,d}$ defined by (7).
  Generate a $l_{rp}$ bit random natural number $r_p$
  Encrypt the predicate vector $\mathrm{EP}_M(\hat{\mathbf{p}}_{a,b,d})$ (= $r_p M^t \hat{\mathbf{p}}_{a,b,d}$)
  return $\mathrm{EP}_M(\hat{\mathbf{p}}_{a,b,d})$
```

The client sends the translated and secure query $\mathrm{TQ}_{M,R_d}(a, b)$ to the server.

We estimate the necessary memory size of transformed key attribute values and queries, and compare them to ones of

TABLE I
COMPARISON OF NECESSARY MEMORY SIZE.

	Plain	Transformed
Key attribute values	l_K	$12l_K + 4(l_\phi + 3l_m + l_{rk})$
Queries	$2l_K$	$8l_K + 4(l_d + l_m + l_{rp})$

plain key attribute values and queries. Let us think about transformed key vectors $r_k D_M^t \hat{\mathbf{p}}_{a,b,d}$, first. r_k is a l_{rk} bit integer and each element of encryption key D_M is a $3l_m$ bit integer. The biggest element of $\hat{\mathbf{p}}_{a,b,d}$ is $\phi k^3 + 3k^2$ and it is a $l_\phi + 3l_K$ bit integer. Considering them, each element of transformed key vector needs $l_{rk} + 3l_m + l_\phi + 3l_K$ bits and totally translated key vector needs four times of them i.e. $12l_K + 4(l_\phi + 3l_m + l_{rk})$ bits. Next, let us think about transformed predicate vectors $r_p M^t \hat{\mathbf{p}}_{a,b,d}$. The perturbation d is a l_d bit integer and r_p is a l_{rp} bit integer. Each element of encryption key M is assumed as l_m bit integer. The biggest element of $\hat{\mathbf{p}}_{a,b,d}$ is $(2a-1)(2b+1)d$ and it is a $2l_K + l_d$ bit integer. Thus, each elements of transformed vector needs $2l_K + l_d + l_m + l_{rp}$ bit and totally transformed vector $r_p M^t \hat{\mathbf{p}}_{a,b,d}$ requires $8l_K + 4(l_d + l_m + l_{rp})$ bits. Table I shows the comparison of necessary memory size between plain values and transformed values.

V. EXPERIMENTAL EVALUATIONS

In this section, we present two sets of experiments we have conducted to evaluate: (i) the query processing time is according to $O(n)$ with the number of tuples n; (ii) the correlations between distributions of plain queries and attribute values and distributions of transformed queries and attribute values, respectively is small enough. All programs are implemented in Python (2.6.4). Experiments were performed on one 2.66GHz processor virtual machine with 512MB running on Virtual Box. We chose parameters of IPP method as $l_K = l_\phi = l_m = l_{rk} = l_{rp} = 32$.

We first present an evaluation of query processing times. We constructed six databases which had different sizes of tuples and made a client request random one million queries to each database. We wanted to reduce effects of communication so that we run the database servers and the client on a same computer. Fig. 2 shows the query processing times. The vertical axis of the left(right) figure is described in the normal (logarithm) scale, respectively. These figures shows if the number of tuples becomes twice, the query processing time also become twice, i.e. the query processing time is according to $O(n)$ with the number of tuples n.

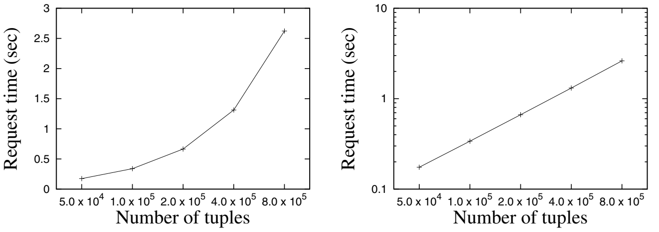

Fig. 2. Query processing times.

We next present an evaluation about the correlations between distributions of plain queries and ones of transformed queries. We made two sets of queries and each set contained one thousand queries which requested $[a, a+100] (a : 1, 2, \cdots, 1000)$. One of the two sets used a uniform random generator R_u to the perturbations d and the other set used R_d defined by (9). Fig. 3 shows the correlation diagram between the left sides a of the plain queries and the first elements of predicate vectors. We omit to describe the results about the other elements of predicate vectors but they were similar to the one about the first elements. Using R_u, smaller a leaded smaller values in the predicate vectors so that the correlation was big. On the other hands, bigger a leaded wide-range values and the correlation was small. Using R_d, the first element values of predicate vectors were distributed in wide range without depending the plain values. It means the correlation is small. Indeed, from Table II, which shows the correlations, R_d had smaller correlations than R_u.

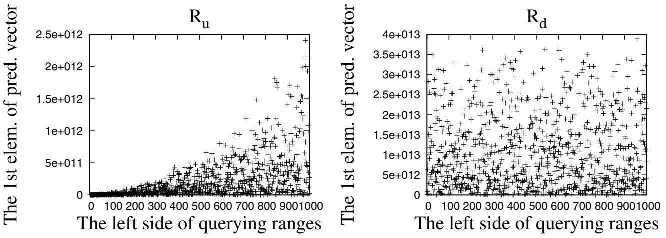

Fig. 3. Correlation diagram between predicate vectors and querying range.

TABLE II
CORRERATIONS BETWEEN PREDICATE VECTORS AND PLAIN QUERIES.

	1st elem.	2nd elem.	3rd elem.	4th elem.	norm
R_u	0.537092	0.536883	0.536790	0.536896	0.536907
R_d	0.014679	0.006242	0.002593	0.006774	0.006856

We finally present an evaluation about the correlation between distributions of the plain Key attribute values and ones of key vectors. We made four sets of tuples. Each set contained one thousand tuples of which Key attribute values were 1 to 1 000. One set used uniform random ganerator R_u to make the perturbations ϕ, and the others used $R_{\phi,\alpha}$ defined by (10). Fig. 4 shows the correlation between the plain Key attribute values k and the first elements of the key vectors associating with k. As same as about queries, we omit to describe the results about the other elements of key vectors but they were similar to the one about the first elements. Table III shows the correlation for all elements of key vectors. Using R_u and $R_{\phi,2}$, the first elements of the key vectors had some kind of correlation that smaller k leaded transformed values of which the absolute values were small and bigger k comparatively leaded transformed values of which the absolute values were bigger. Indeed, in Table III, the absolte values of both correlations were not small. Note that, the reason why the correlations have negative values is because the encryption key D_M had some negative values. Using $R_{\phi,3}$ and $R_{\phi,4}$, the transformed values distributed almost uniformly.

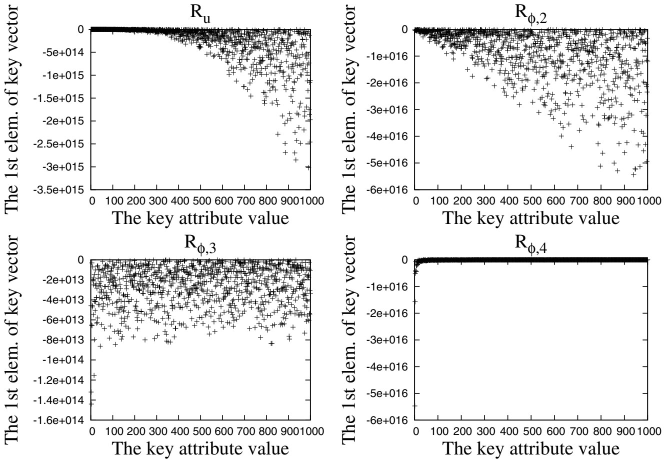

Fig. 4. Correlation diagram between key vectors and key attribute values.

TABLE III
CORRELATIONS BETWEEN KEY VECTORS AND PLAIN ATTRIBUTE VALUES.

	1st elem.	2nd elem.	3rd elem.	4th elem.	norm
R_u	-0.739942	0.740338	0.739848	-0.739575	0.739876
$R_{\phi,2}$	-0.654244	0.641542	0.657164	-0.665624	0.656257
$R_{\phi,3}$	0.229862	-0.320270	-0.143313	-0.131806	-0.181528
$R_{\phi,4}$	0.152775	-0.113652	-0.281633	-0.070735	-0.166220

The transformed values were not depend on the plain attribute values. Table III also shows the correlations was small enough.

VI. CONCLUSION

In this paper, we propose a new private query approach, IPP method, which supports range queries and allows tuples have a same Key attribute value. IPP method is based on a query transform by trusted query (QT) scheme in which trusted clients of users transform queries and Key attribute values to secure queries and Key attribute values. On this QT scheme, IPP method provides a transformation algorithms which adds perturbations to queries and Key attribute values and encrypts them, so that it ensures the distributions of transformed queries and Key attribute values to be different from the ones of plain them. Finally, IPP method protects the frequency analysis of attackers. IPP method provide the such kind of private range query but it require more memorizing cost and computational cost to servers than plain requesting approaches. The memorizing costs for each query and each Key attribute value are shown in table I. They needs four times and 12 times as large as costs of plain them, respectively. The computational cost belongs to, as same as basic PIR approach, $O(n)$ with the number of tuples n.

From the experimental evaluations, we confirmed the computational cost on servers belongs to $O(n)$ with the number of tuples n. We also confirmed that the distributions of transformed queries and Key attribute values and the ones of plain them have almost no correlation if the perturbations are according to the alternative random distributions which we describe in section III-D. The fact means the distributions of transformed queries and Key attribute values are different from the ones of plain them.

We mainly argue about the Key attribute but $Value$ attribute, in this paper. That is because we think to protect $Value$ attribute values, which is not used to query process, we are able to use existing encryption schemes. There is an open problem that attackers may guess the plain queries and Key attribute values by gathering and analyzing results of queries. However, in general, each result of queries consists from many tuples so that gathering the results needs much more memorizing space. We think that it is also necessary to argue about effectiveness of attacks for the results of querying.

REFERENCES

[1] G. I. Davida, D. L. Wells, and J. B. Kam, "A Database Encryption System with Subkeys," *ACM Transactions on Database Systems*, vol. 6, no. 2, pp. 312–328, Jun. 1981.

[2] H. Hacıgümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," in *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data*. Madison, WI, USA: ACM, 2002, pp. 216–227.

[3] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private Information Retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–982, 1998.

[4] R. Ostrovsky and W. E. Skeith, III, "A Survey of Single-Database PIR : Techniques and Applications," in *Proc. of the 10th International Conference on Practice and Theory in Public-key Cryptography*. Beijing, China: Springer-Verlag, 2007, pp. 393–411.

[5] E. Kushilevitz and R. Ostrovsky, "Replication is not needed: Single Database, Computationally-Private Information Retrieval," in *Proc. of the 38th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Press, 1997, pp. 364–373.

[6] S. Wang, D. Agrawal, and A. E. Abbadi, "Generalizing PIR for Practical Private Retrieval of Public Data," in *Proc. of the 24th Annual IFIP WG 11.3 working Conference on Data and Applications Security and Privacy*. Rome, Italy: Springer-Verlag, 2010, pp. 1–16.

[7] L. Sweeney, "k-anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 1–14, 2002.

[8] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private Queries in Location Based Services : Anonymizers are not Necessary Categories and Subject Descriptors," in *Proc. of the 2008 ACM SIGMOD International Conference on Management of Data*, no. ii. Vancouver, Canada: ACM, 2008, pp. 121–132.

[9] B. Hore, S. Mehrotra, and G. Tsudik, "A Privacy-Preserving Index for Range Queries," in *Proceedings of the 30th International Conference on Very Large Data Bases*. Toronto, Canada: VLDB Endowment, 2004, pp. 720–731.

[10] R. C.-W. Wong, A. W.-C. Fu, J. Liu, K. Wang, and Y. Xu, "Global Privacy Guarantee in Serial Data Publishing," in *Proc. of the 26th International Conference on Data Engineering*, Long Beach, CA, USA: IEEE Press, 2010, pp. 956-959.

[11] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order Preserving Encryption for Numeric Data," in *Proc. of the 2004 ACM SIGMOD International Conference on Management of Data*. Paris, France: ACM, 2004, p. 563–574.

[12] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN Computation on Encrypted Databases Categories and Subject Descriptors," in *Proc. of the 35th SIGMOD International Conference on Management of Data*. Providence, RI, USA: ACM, 2009, pp. 139–152.