A Cryptography Index Technology and Method to Measure Information Disclosure in the DAS Model

ZHAO WEI [1, 2], ZHAO DAN-FENG [1], GAO FENG [3], LIU GUO-HUA [1]
[1] College of Information Science and Engineering
Yanshan University
Qinhuangdao, Hebei, 066004
China
[2] Harbin Power Supply Bureau Information Centre
Harbin, Heilongjiang, 150001
China
[3] College of Information Science and Engineering
Fudan University
Shanghai, 200433
China
kis_2@163.com

Abstract: - Database-as-a-Service model is a new data management model which allow user to store their data at database service provider. In DAS model, data is stored in cryptograph form, so it will spend long time to query data. In order to improve cryptograph query efficiency, a cryptograph index strategy adapting to unequal-probability query is presented. Then, definitions of disclosure coefficient are presented focus on the problem of the information disclosure in cryptograph indices. Finally, the conclusion is analyzed and validated by experiments.

Key-Words: - Database-as-a-Service, Cryptography index, Query probability, Disclosure coefficient, Encrypted database system, Data cryptography

1 Introduction

As a consequence of the trend towards the database owner publishes its data through remote servers, sensitive data are no more under the control of the data owners and their confidentiality and integrity may be at risk[1].

We would like to point out the security issues in such models. Examples include: privacy-preservation issues [22 23], secure query execution [9], security in conjunction with access control requirements [24, 25, 26] and query execution assurance [27]. query execution assurance of [27] does not provide authentication: the server could pass the challenges and yet still return false query results.

In order to ensure the confidentiality of the outsourced data, cryptographic techniques are usually adopted [2]. By encrypting the outsourced data, clients are guaranteed that they alone can access the data [3].

However, encryption technology in database system has a serious negative effect on query performance. Many techniques have been developed aimed at efficiently querying encrypted database [4, 5] that associates indices with the outsourced data. Yet, inference attacks exploiting index information are existed [6].

The approach in [7] balances the trade off between efficiency requirements in query execution and protection requirements due to possible information disclosure. However, all methods require queries are equiprobable ones.

In this paper, in order to address the problem mentioned above, new cryptograph index technology is proposed which first ensures the security by limiting the number of partitioning buckets, and then improves the query efficiency by minimizing the wrong hit expectation using different query probabilities. Then, a function of disclosure coefficient is used to detect the degree of information disclosure.

The rest of the paper is organized as follows. After a brief discussion on related work in Section 2, section 3 introduces the basic knowledge about DAS model. We describe the bucket partition index algorithm based on query probability in Section 4. Section 5 presents disclosure coefficients and discusses how to measure information disclosure. Experimental results are shown in Section 6. Section 7 concludes.

2 Related Work

There are two approaches aiming at processing queries over encrypted data and protecting the data confidentiality for outsourced indexed data [7, 19].

In [19], the author proposed storing, together with the encrypted data, additional indexing information. [7] introduced a method to query a tuple-lever encrypted database but with a better security lever for the outsourced data.

However, as shown in [15], it can be exploited by the untrusted server to carry out inference and linking attacks. In a quite different scenario, when the data owners store their private data on a third party and allow other uses to access the outsourced data, the data privacy may also become important [16, 17, 19-21].

In [16], the authors introduced an approach to ensure the data privacy in PIR schemes and in [17] a solution to the outsourcing model was introduced. Yet they can not be applied to outsourced search trees.

Bucket partition techniques for executing queries on untrusted serves without loss of efficiency are proposed in [9-14]. Methods in [9, 10] support efficient evaluation on the remote server of both equality and range predication. However, it makes it awkward to manage the correspondence between interval and the actual values.

A related solution using smart cards for key management is proposed in [1]. Schemes in [11, 12] enhanced query hit rate, but it became a question that the limit of partition buckets number affected the query hit rate. [13, 14] presented index technologies which solved the question of low query hit rate. However large disclosure of information still exists and the queries in these methods were limited to equiprobable query.

3 DAS model

This section introduces the basic knowledge of DAS model.

3.1 Query process in DAS model

There are four main parts in DAS model.

- Data Owner: storing data in the database of the database-server-provider (DSP), this is used by User.
- User: submitting query application to Client through internet to access the data stored in DSP after authorized by Data Owner.
- Client: converting original query to cryptograph query, and submit it to Server through internet, meanwhile encryption and decryption manipulation are executed here.
- Server: managing encrypted database and executing query manipulation submitted by Client [18].

In traditional database, DBA has the keys, but in DAS model it is necessary to protect data from the database provider, so the data are encrypted at Client.

A user poses the query to the Client. The Client translates it to two parts, one is query over encrypted data which is executed at Server side and returns a temp-result to Client, another is query over decrypted data which is used at the temp-result to filters the unsatisfied tuples and returns the final result to user, as Fig. 1 shows.

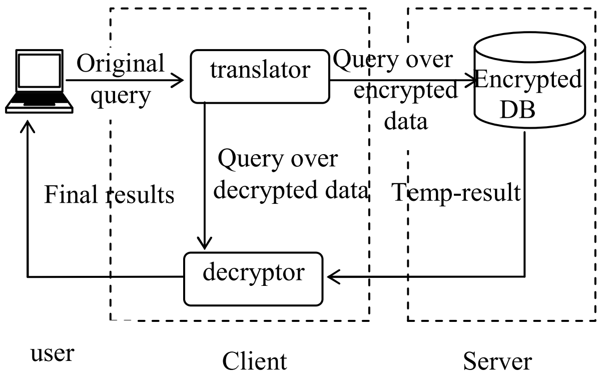


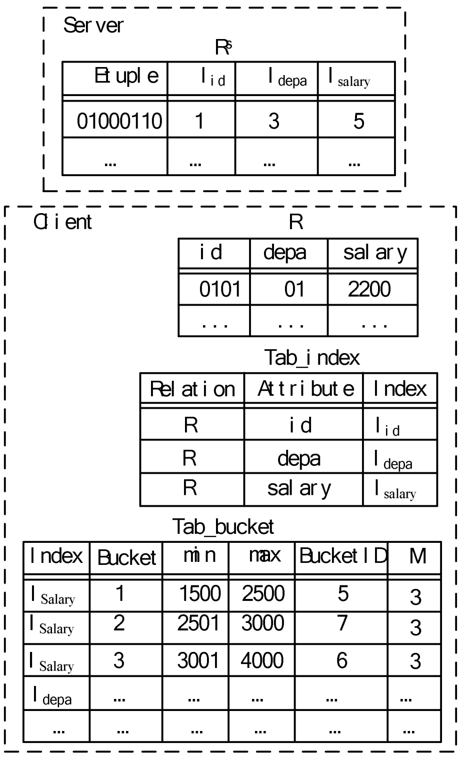
Fig. 1 The query process in DAS model

3.2 Metadata about index in DAS model

In order to efficiently access and manage database, it generally provides so-called meta data for data owner and server. Clients and servers interpret and execute SQL using such metadata. the principles of managing metadata is as follow:

- Data owners are entitled to manage all metadata.
- Ordinary users have no access to metadata.
- Anyone can not manipulate metadata on the server.

As a result, like other relations, metadata can be organized as relation structure and be manipulated using SQL, which simplifies maintains of data owners and makes it difficult for attracters to access the metadata. For example, an relation instance and part of its storage information is shown in Fig. 3 and the encrypted relation R^s is stored in the server; the original relation R, table Tab_index related to attributes and indices and table Tab_bucket containing indices information are stored in the client.


Server

R^s

Etuple	I_{id}	I_{depa}	I_{salary}
01000110	1	3	5
…	…	…	…

Client

R

id	depa	salary
0101	01	2200
. . .	. . .	. . .

Tab_index

Relation	Attribute	Index
R	id	I_{id}
R	depa	I_{depa}
R	salary	I_{salary}

Tab_bucket

Index	Bucket	min	max	Bucket ID	M
I_{Salary}	1	1500	2500	5	3
I_{Salary}	2	2501	3000	7	3
I_{Salary}	3	3001	4000	6	3
I_{depa}	…	…	…	…	…
…	…	…	…	…	…

Fig.2 Index metadata of R

4 A Cryptograph Index Based on Query Probability

The result set of cryptograph queries which are executed through bucket partition index is a superset of the final result, which means wrong hit tuples exist in the former result set. This result set is returned to a client site, which calculate and obtain correct results.

So the presence of the wrong hit tuples strongly affects query efficiency. The distribution of query probability influences the number of wrong hit tuples, so we create cryptograph index from the query probability in this paper. The index is created on nonnegative integer domain in this paper.

4.1 Definitions

There are several symbols used in following definitions.

1) A: an attribute.

2) D: domain of A, D={$v_1, v_2, \dots, v_N$}.

3) N: the amount of values in D.

4) v_i: any arbitrary value in D, 1 i N.

5) f_i: the amount of tuples whose attribute value is v_i (frequency of value v_i), 1 i N.

6) M: the number of partition buckets.

7) B_i: the i^{th} *partition* bucket.

8) Bid_i: the identifier of the i^{th} bucket, namely the index number [6].

9) n: the amount of different attribute values in a single bucket.

10) q: any query.

11) $q(v_i)$: result of query q, which contains v_i.

*Example*1: The domain of attribute depa in relation R is D={01,02, 03, 04}, $f_1=f_2=f_4=2, f_3=1$, as shown in table 1. a bucket partitioning with M=2 on attribute depa is executed given B_1={01,02}, Bid_1=3; B_2={03,04} and Bid_2=4, a index row $depa^s$ of relation R^s is returned, shown in table2.

Table1. Relation R

id	*depa*	*salary*
0101	01	2200
0102	01	2300
0201	02	3500
0202	02	2250
0301	03	2700
0401	04	2700
0402	04	2820

Table 2. Encrypted relation R^s

etuple	I_{id}	I_{depa}	I_{salary}

01000110	1	3	5
00111000	3	3	5
10010101	1	4	6
11000111	8	4	5
01110100	3	3	7
01011101	8	4	7
10110010	1	4	7

Definition 1: Query probability $p(q(v_i))$. Repeat query n times on attribute A with a domain $D=\{v_1, v_2,\ldots, v_n\}$. $n(q(v_i))$ is the occurrence number of $q(v_i)$ in these examinations. Along with the increasing of n, the frequency $n(q(v_i))/n$ gradually stabilizes around a certain value $p(q(v_i))$, where $p(q(v_i))$ is the query probability of query result $q(v_i)$.

Suppose there is only one kind of single condition equivalence query, which expressed by SQL is *q*: SELECT * FROM table *R* WHERE *depa*=v_i, the statistical probabilities of v_i *are* 0.23, 0.28, 0.16 and 0.33, then the corresponding query probabilities of each attribute values are $p(q(01))=0.23$, $p(q(02))= 0.28$, $p(q(03))=0.16$, $p(q(04))=0.33$.

Definition 2: Bucket query probability $p(q(B_i))$. Given partition buckets of attribute A as B_1, $B2,\ldots, B_M$, q is a query, $q(B_i)$ is query result of q which contain random value in B_i, $p(q(B_i))$ is the probability of q which query result is $q(B_i)$.

In *Example* 1, $01,03\in B_1$; $02,04\in B_2$, thereby $p(q(B_1)) = p(q(01)) + p(q(03)) = 0.39$, $p(q(B_2))= p(q(02))+ p(q(04))=0.61$.

Definition 3: Cryptograph query result $q^s(v_i)$. Cryptograph q^s is corresponding with query q which result is $q(v_i)$, while $q^s(v_i)$ is the result of q^s.

Given a query on relation *R as q:* SELECT * FROM table *R* WHERE *depa*=02, query result $q(02)$ is the two tuples with ids of 0201 and 0202. When query is executed on cryptograph relation R^S, it is rewritten as q^s : SELECT * FROM table R^S WHERE $depa^s$=4, and the cryptograph query result $q^s(02)$ contains four tuples with ids of 0201, 0202, 0401 and 0402.

Defination 4: Wrong hit tuple $ET(q^s(v_i))$. Tuples containing the wrong result in $q^s(v_i)$, denoted by $ET(q^s(v_i))$.

In $q^s(02)$, wrong hit tuples $ET(q^s(02))$ is the tuples with ids of 0401 and 0402, the number of them is $|ET(q^s(02))|=2$.

Defination 5: Unit wrong hit expectation $E(B_j)$. Given partition buckets of attribute A as B_1, $B_2,\ldots,B_j,\ldots,B_M$, $E(B_i)$ is the expectation of the number of wrong hit tuples in bucket *j*,

$$E(B_j)=\sum_{i=1}^{n}\Big(\big(p(q(v_i))/p(q(B_j))\big)|ET(q^s(v_i))|\Big) \quad (1)$$

In *Example* 1, the unit wrong hit expectation of bucket B_1 is

$E(B_1)=\{p(q(01))/[p(q(01))+p(q(03))]\}|ET(q^s(01))| +\{p(q(03))/[p(q(01))+p(q(03))]\}|ET(q^s(03))|=[0.23/(0.23+0.16)]*1+[0.16/(0.23+0.16)]*2=1.41$

Defination 6: Collectivity wrong hit expectation *E*. Given partition buckets of attribute A as $B_1, B_2,\ldots, B_M$, *E* is the expectation of the number of wrong hit tuples in the hole domain,

$$E=\sum_{j=1}^{M} p(q(B_j))E(B_j) \quad (2)$$

In *Example* 1, the collectivity wrong hit expectation of attribute depa is $E=p(q(B_1))E(B_1)+p(q(B_2))E(B_2)=0.39*1.41+0.61*2=1.7699$.

4.2 Cryptograph Index Method Based on Query Probability

The effect of query probability on partition strategy is often ignored in present cryptograph index technologies based on bucket partition. We presume the query probability on each attribute value is equal, viz. $p(q(v_i))= p(q(v_j)),(i,j\in[1,N], i\neq j)$. The result is some buckets would be accessed frequently, because attribute values with great query probability are stored in them. And all values in these buckets become temp results, which increase the cost of post process and affect the query efficiency. The query probability of attribute value is regarded as an important factor in the process of bucket partitioning in this paper and a method of creating cryptograph index based on query probability is proposed.

Theorem 1: Given attribute $v_i(1\le i\le N)$, corresponding query probability $p(q(v_i))$ and tuples number f_i, attribute *A* is partitioned into *M* buckets, labeled the number of different attribute values in each bucket as *n*, then the advantage of index can be measured by the formula below, and the less *E* is, the better of the index strategy,

$$E=\sum_{j=1}^{M}\left(\sum_{i=1}^{n}p(q(v_i))\sum_{i=1}^{n}f_i^j-\sum_{i=1}^{n}p(q(v_i))f_i^j\right). \quad (3)$$

Proof: Different index strategies cause different query wrong hit number, so query wrong hit number could measure the advantage of index.

First, to compute unit wrong hit expectation $E(B_j)$. For a bucket with n attribute values, bucket query probability is $p(q(B))=p(q(v_1))+p(q(v_2))+\ldots+p(q(v_n))$; the probability to query the i^{th} attribute value is $p(q(v_i))/p(q(B))$, $(1\le i\le n)$; the wrong hit number is

$$|ET(q^s(v_i))|=f_1+f_2+\ldots+f_{i-1}+f_{i+1}+\ldots+f_n.. \quad (4)$$

Then, for bucket $j(1\le j\le M)$, the unit wrong hit expectation is

$$E(B_j)=\sum_{i=1}^{n}\left(\left(p(q(v_i))/p(q(B_j))\right)|ET(q^s(v_i))|\right)$$
$$=\sum_{i=1}^{n}f_i-\left(\sum_{i=1}^{n}p(q(v_i))f_i\right)/\left(\sum_{i=1}^{n}p(q(v_i))\right). \quad (5)$$

Second, to compute collectivity wrong hit expectation.

$$E=\sum_{j=1}^{M}p(q(B_j))E(B_j)$$
$$=\sum_{j=1}^{M}\left[p(q(B_j))\left(\sum_{i=1}^{n}f_i-\left(\sum_{i=1}^{n}p(q(v_i))f_i\Big/\sum_{i=1}^{n}p(q(v_i))\right)\right)\right]$$
$$=\sum_{j=1}^{M}\left(\sum_{i=1}^{n}p(q(v_i))\sum_{i=1}^{n}f_i^j-\sum_{i=1}^{n}p(q(v_i))f_i^j\right) \quad (6)$$

So, collectivity wrong hit expectation is related to the strategy of bucket partition under the circumstance of the query probabilities and occurrence numbers are known. The bucket partition strategy is better by increasing value E. So formula (6) can be used to measure the performance of index strategy and be used as determining condition.

4.3 Cryptograph Index Algorithm Based on Query Probability

Cryptograph index algorithm aims at minimizing the collectivity wrong hit expectation E, and its algorithm idea is to choose the best bucket for each attribute value through minimizing E, thus the optimal bucket partition strategy is presented. Then distribute different bucket ids randomly to each bucket, which is also stored in cryptograph database as index values.

Input: $D=(V,P,F)$, $m=M$

Output: $B_i=\{v_j\}$ /* each bucket */

E /* the wrong hit expectation under this partition */

BID /*bucket id, index value */

Probability Cryptograph Index Algorithm (*PCI*):

```
For ($k$=1; $k$<=$m$; $k$++)
    $B_i=\{v_k\}$
For ($k=m+1$; $k$<=$n$; $k$++ )
{
  if $v_k<maxB_1$
    {
      $v_k\in B_1$;
      $maxB_1\in$ JudgB(1,$maxB_1$) ;
    }
    if $v_k=minB_m$
    {
       $v_k\in B_m$;
       $minB_m\in$ JudgB($m$,$minB_m$) ;
    }
    if ($maxB_1<v_k<minB_m$)
    {
      for($l$=2; $l$<=$m$-1; $l$++)
      {
        if ($minB_l<v_k<=maxB_l$)
        {
          $v_k\in B_l$;
          $maxB_l\in$ JudgB($l$, $maxB_l$);
          $minB_l\in$ JudgB($l-1$, $minB_l$);
        }
        if($v_k<minB_{l+1}$ && $v_k>maxB_l$)
```

```
            $v_i \in$ JudgB($l$,$v_i$);
          }
        }
    }
    Create a set of random number $S=\{s_1, s_2,\ldots,s_M\}$;
    For ( $i$=1, $i$<= $M_2$, $i$++)
      {
        $Bid_i=s_i$;
      }
    Return $B_i=\{v_j\}$;
    BID={ $Bid_1, Bid_2,\ldots, Bid_M$ };
    END
Function of judging two neighboring buckets
    JudgB($u$,$v_i$)
/* to judge which is the best bucket for $v_i$ the $u^{th}$ or $u+1^{th}$ */
  {
      $E_u$;
/* compute E according to formula (6) if $v_k$ is inserted into $B_u$, */
      $E_{u+1}$;
/* compute E according to formula (6) if $v_k$ is inserted into $B_{u+1}$, */
      min($E_u$,$E_{u+1}$);
      Return $B_v$;
  }
```

4.4 Algorithm Analysis

There are three kinds of cases for each attribute value.

1) To be inserted into bucket 1, if its query probability is less than the max in this bucket.
2) To be inserted into bucket M, if its query probability is greater than the min in this bucket.
3) To be inserted into the rest (M-2) buckets, other cases.

When there is case 1), judge if the one with maximal probability in bucket 1 should be moved to the right bucket. When there is case 2), judge if the one with minimal probability in bucket M should be moved to the left bucket. When there is case 3), judge if the one with minimal probability in bucket M should be moved to the left bucket and so do the max in the same time.

This algorithm consists of two loopsone part is a sort for the first M values, and another aims at judging buckets for the other N-M values. There are returns after judgment functions, and the executed time is a known finite integer, so it is not an endless loop, and this algorithm is terminable.

The set of attribute values has n elements, and there are M partition buckets in all. The two loops are independence, the first is executed M times and its time complexity is $O(M)$; there are three cases in the second one, the best is case 1) or 2) occur in each second step partition, and the judgment function is executed (n-M_2) times; the worst is case 3) occur in each time, and the executed time is 2(n-M_2), for the time complexities are $O(n)$ in any cases. So the time complexity of the algorithm is $O(n)$.

5 A Method of Measuring Information Disclosure Using Disclosure Coefficient

Intruders can dope out some information according to multi-data result when researching by index in cryptograph database, so creating index would causes latent information disclosure to cryptograph database. Several disclosure coefficients are proposed here to measure the information disclosure.

Defination 7: bucket disclosure coefficient λ_{B_i}: given an index I with M buckets, when the query result is the tuples according with all the values in a certain bucket B_i, the probability of guessing right for attracters is called the bucket disclosure coefficient of B_i, labeled as $\lambda_{B_i} = 1\Big/\sum_{k=1}^{n_i} f_k$, where n_i is the number of different values in B_i.

Definition 8: index disclosure coefficient λ_I: given an index I with M buckets, when the query result is the tuples according with all the values in a random bucket B_i, the probability of guessing right for attracters is called the index disclosure coefficient, labeled as

$$\begin{aligned}\lambda_{R^s} &= \frac{1}{M}\sum_{vi=1}^{M}\lambda_{B_i} \\ &= \frac{1}{M}\sum_{i=1}^{M}\frac{1}{\sum_{k=1}^{n_i} f_k}\ .\end{aligned} \tag{7}$$

Defination 9: relation disclosure coefficient: given relation R^s with k indices, where $k \ge 1, k \in R$, when the query result is random tuples in relation R^s, the probability of guessing right for attracters is called the index disclosure coefficient, labeled as λ_{R^s}.

In *Example* 1, indices I_{id}, I_{depa} and $I_{salaryI}$ in table2 are corresponding to attributes *id*, *depa* and *salary* respectively.

Index I_{id} contains three storage buckets B_1={0101,0201,0402};B_2={0102,0301};B_3={0202,0401}. If the temp result is all the values in B_1, the probability for attracters to guess right is 1/3, that is the bucket disclosure coefficient for B_1 is $\lambda_{B_1} = 1/3$, and $\lambda_{B_2} = 1/2$, $\lambda_{B_3} = 1/2$ in the same way.

The buckets numbers of indices I_{id} I_{depa} and I_{salary} are 3, 2 and 3, accronding to definition 8 $\lambda_I = \frac{1}{M}\sum_{i=1}^{M}\lambda_{B_i}$, $\lambda_{I_{depa}} = 4/9$, $\lambda_{I_{depa}} = 7/24$, $\lambda_{I_{salary}} = 5/9$, as shown in Table 3.

For a multi-index relation, the relation disclosure coefficient can't be decided simply by any index disclosure coefficient or their algebraic operation. Supposing tuple in R^s is the fundamental unit, the disclosure of each tuple is influenced by all the corresponding index disclosure coefficients together. Because each index returns a superset, and the final result is in the intersection of these supersets. The elements number in this interest decides the probability of attractors' right guesses.

Table 3. Disclosure coefficient values

Index	I_{id}	I_{depa}	I_{salary}
λ_{B_i}	1/3	1/3	1/3
	1/2	1/3	1/3
	1/3	1/4	1
	1/2	1/4	1/3
	1/2	1/3	1/3
	1/2	1/4	1/3
	1/3	1/4	1/3
λ_I	4/9	7/24	5/9
λ_{R^s}		2/3	
λ_{R^s}		2/3	

Supposing tuple T_v corresponds toB_i in I_j, $F_i^j = \sum_{u=1}^{n_i^j} f_u$ is the number of attribute values in B_i including repeated ones, that is the returned number of values after querying this tuple through I_j. And then the returned number of values searched through k indices is $F \in \left[1, \min(F_i^j)\right]$, supposing F is satisfied hypodispersion in this range, then the probability of guessing this tuple right is

$$P_{T_v} = \frac{2}{\min(F_i^j)+1}\ . \tag{8}$$

So, the formula of relation disclosure coefficient is

$$\begin{aligned}\lambda_{R^s} &= \frac{1}{N}\sum_{v=1}^{N}P_{T_V} \\ &= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{2}{\min\left(F_i^j\right)+1}\right)\end{aligned} \tag{9}$$

According to formula (9), the relation disclosure coefficient of R^s in the example above is

$$\begin{aligned}\lambda_{R^s} &= \frac{1}{7}\left(\frac{2}{3+1}+\frac{2}{2+1}+\frac{2}{1+1}\cdots+\frac{2}{3+1}\right) \\ &= \frac{2}{3} \approx 0.6667\end{aligned} \tag{10}$$

If there are two indices I_{depa} and I_{salary} in R^s, the relation disclosure coefficient is

$$\begin{aligned}\lambda'_{R^s} &= \frac{1}{7}\left(\frac{2}{3+1}+\frac{2}{3+1}+1+\cdots+\frac{2}{3+1}\right) \\ &= \frac{4}{7} \approx 0.5714\end{aligned} \tag{11}$$

If there is only one index I_{depa} in R^s, the relation disclosure coefficient is

$$\lambda''_{R^s} = \frac{1}{7}\left(\frac{2}{3+1}+\frac{2}{3+1}+\frac{2}{4+1}\cdots+\frac{2}{4+1}\right)$$
$$=\frac{31}{70}\approx 0.4429 \qquad (12)$$

There are several rules can be summarized from the formula (9) and the gotten data.

- Given a certain relation, the value of index disclosure coefficient raised while the buckets number increased, vice versa.
- The value of relation disclosure coefficient rises while the number of indices increased, and the information security decreased.
- The value of formula (9)is decreased with the increasing of F_i^j , when the relation is certain, F_i^j increased with the reducing of corresponding buckets number M. So cut down M can enhance the relation security.

6 Experimental Results and Evaluation

We have conduct two sets of experiments on parallel corpus of ‘Dream of the Red Chamber’ to show the validity and the effectiveness of our proposed method.

The experiments are conducted on two Intel-based personal computers with P4 2.4G Hz processors with 1 GB RAM. One serves as the server and another is the client. We choose SQL Server 2000 as database and Windows XP as the operating system.

This section details three sets of experiments. The first set of experiments validates the efficiency through wrong hit rate under a constant buckets number and variable record numbers.

Fig.3 represents the query wrong hit rate as a function of records number for different partitioning schemes, given that the amount of bucket M is 10.

The result shows that the proposed the method proposed in this paper has lower wrong hit rates than equi-depth under a certain partition buckets number, no matter how many the records number is.

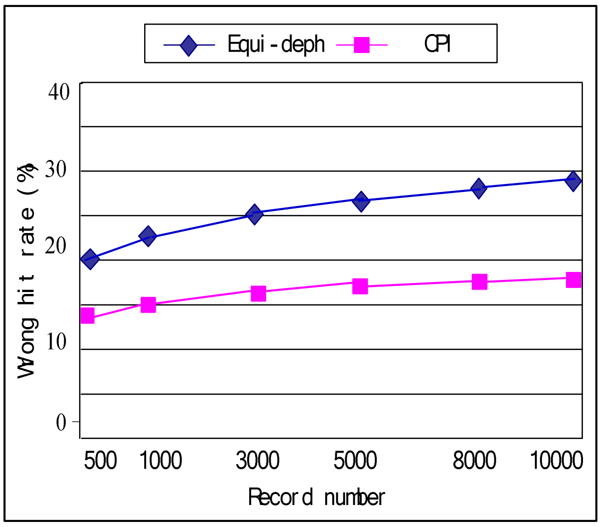


Fig. 3 The numbers of records and wrong hit rates

We second perform a set of experiments designed to cryptograph index on attributes in relation *Tab_wordfrequency* with 5000 records in it, and test the relationship between the *M* (number of partitioning bucket) and index security, which is indicated by index disclosure coefficient $\lambda_I = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{F_i}$ shown in Fig. 4.

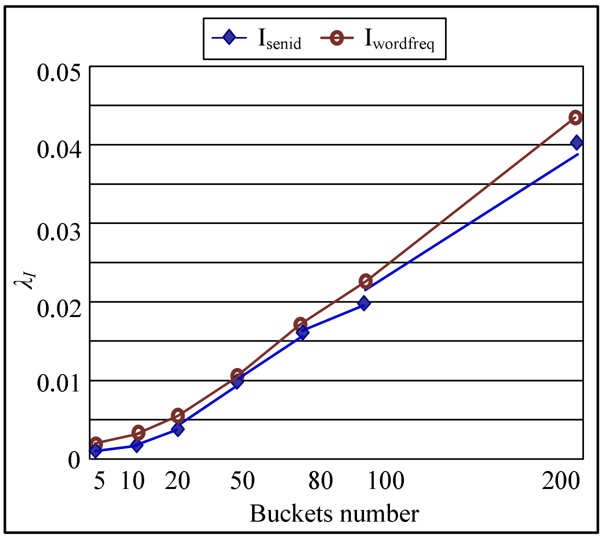


Fig. 4 Relation of λ_I and M

In the second set of experimens, we create indices on the attributes *senid*, *paraid* and *wordfreq* of the same relation and test the relationship between M and relation disclosure coefficient according to $\lambda_{R^s} = \frac{1}{N}\sum_{v=1}^{N} P_{T_v}$. The result shown in Fig.5 indicates that increase of number of indices reduces the relation security.

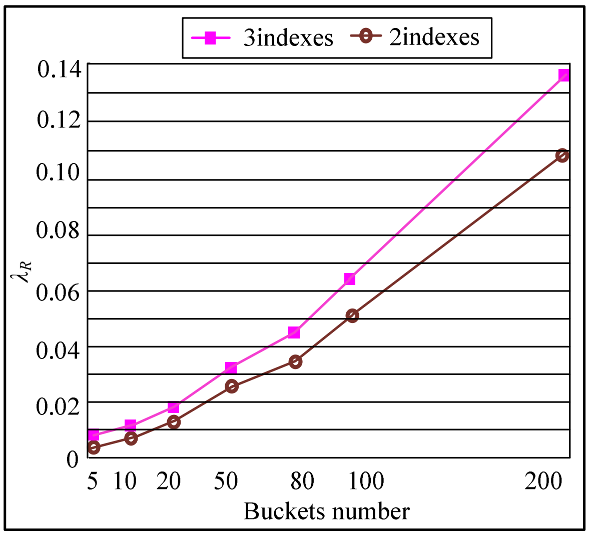


Fig. 5 The relation of λ_{R^S} , M and indices numbers

7 Conclusions

Low efficiency of encrypted database system performance in DAS model is caused by both low query hit rate and huge cost of post processing. By introducing query probability our proposed cryptograph index technology can address these two problems effectively and can be further applicable to non-equiprobable query. Three formulas of disclosure coefficient are proposed to determine the degree of information disclosure. Experiments show that our method can improve the query efficiency effectively, and index security is validated through disclosure coefficient.

References:

[1] L. Bouganim and P.Pucheral. Chip-secured data access: Confidential data on untrusted servers. In Proc. Of the 28th International Conference on Very Large Data Bases, Hong Kong, China, August 2002.131-142,

[2] Davida G, D Wells, J Kam. A database encryption system with subkeys. ACM Transactions on Database Systems, 1981:312-328.

[3] D.X.Song, D.Wagner, A. Perring. Practical Techniques for Searches on Encrypted Data. In IEEE Symposium on Security and Privacy, 2000.

[4] Damiani, E., S. De Capitani di Vimercati, M. Finetti, S. Paraboschi, P. Samarati and S. Jajodia, Implementation of a storage mechanism for untrusted DBMSs, in: Proc. Of the Second International IEEE Security in Storage Workshop, Washington DC, USA, 2003.

[5] H Hacigumus, B Iyer, S Mehrotra. Providing database as a service, in: Proc. of 18th International Conference on Data Engineering, San Jose, California, USA, 2002.

[6] Zhu Qin, Yu ShouJian, Le Jiazin. Research on Security Mechanisms of Outsourced Database [J]. *Computer Science*. 2007,152-156

[7] E.Damiani, S. De Capitani Vimercati, S. Jajodia, S. Pareaboschi, and P. Samarati. Banlencing Confidentiality and Efficiency in Untrusted Relational DBMSs. InCCS'03. ACM Press.

[8] Ma Shaobu, Hu Lei, Xu Deqi. A Dynamically Secure In- dex on Encrypted Database [J]. *Computer Engineering*, 2005, 31(6): 132-133.

[9] H Hacigumus, B Iyer, S Mehrotra. Executing SQL over Encrypted Data in Database Service Provider Model. *ACMSIGMOD*. New York, 2002.

[10] H Hacigumus, B Iyer, S Mehrotra. Efficient Execution of Aggregation Queries over Encrypted Relational Databases. *In Proc. of the 9th International Conference on Database Systems for Advanced Applications*, Jeju Island, Korea, March 2004.

[11] Bijit Hore,Sharad Mehrotra,Gene Tsudik. A Privacy-Index for Range Queries. *Proceedings of the 30th VLDB conference,Toront*o, Canada, 2004.223-235.

[12] Yu Han, Zhao Liang, Xu Weijun. Research on a New Me- thod for Database Encryption and Cipher Index [J]. *Acta Electronica Sinica*, 2005, 12(3): 23-25.

[13] Wang Di, Liu Guohua, Yu Xingbing. Cryptograph Index Technology based on Multi-buckets Partition [J]. *Application of Electronic Technique*, 2007.3:141-144.

[14] Wang Di, Liu Guo-hua, Yu Xing-bing. Cryptograph Index Technology Based on Strategy of Optimal Bucket Partitioning. Journal of Chinese Computer Syestems[J].2008 Vol.29 No.4 P.649-652.

[15] T.K. Dang. Extreme Security Protocols for Outsourcing Database Services. Proc. of the 6th International Conference on Information Integration and Web-based Applications and Services - iiWAS 2004, Jakarta, Indonesia, 2004. 497-506

[16] Yael Gertner, Yuval Ishai,Eyal Kushilevitz. Protecting data privacy in private information retrieval schemes. Journal of Computer and System Sciences. 2000, 6(3), p592- 629.

[17] W. Du and M. J. Atallah. Protocols for secure remote database access with approximate matching. In Proc. of the First Workshop on Security and Privacy in E-Commerce, Nov. 2000.

[18] Zhao Dan-feng, Jin Shun-fu, Liu Guo-hua, and Gao Feng. A cryptograph index technology based on query probability in DAS model [J]. Journal of Yanshan University, 2008, 32(6):477-482.

[19] Kapil A. Gwalani, and Omar Elkeelany. Design and Evaluation of FPGA Based Hardware Accelerator for Elliptic Curve Cryptography Scalar Multiplication. *WSEAS TRANSACTIONS on COMPUTERS,* Issue 6, Volume 8, June 2009

[20] Ching-Sheng Hsu, Shu-Fen Tu, and Young-Chang Hou. A Document Protection Scheme using Innocuous Messages as Camouflage. *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS*, Issue 5, Volume 6, May 2009.

[21] Jan Hajny, Tomas Pelka, and Vaclav Zeman. Flexible Authentication Framework with Bound Authentication and Authorization, *WSEAS TRANSACTIONS on COMMUNICATIONS*. Issue 1, Volume 8, January 2009.

[22] R. Agrawal and R. Srikant. Privacy-preserving data mining. In Proc. of ACM Management of Data (SIGMOD) 2000. 439-450.

[23] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In Proc. of ACM Symposium on Principles of Database Systems (PODS), 2003. 211-222.

[24] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In Proc. of Very Large Data Bases (VLDB). 2003. 898-909.

[25] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy.Extending query rewriting techniques for _ne-grained access control. In Proc. of ACM Management of Data (SIGMOD). 2004. 551-562.

[26] L. Bouganim, F. D. Ngoc, P. Pucheral, and L. Wu.Chip-secured data access: Reconciling access rights with data encryption. In Proc. of Very Large Data Bases (VLDB), 2003. 1133-1136.

[27] R. Sion. Query execution assurance for outsourced databases. In Proc. of Very Large Data Bases (VLDB). 2005. 601-612,