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-level encrypted database but with a better security level 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 users 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 cannot be applied to outsourced search trees.

Bucket partition techniques for executing queries on untrusted servers 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 cryptographic 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](image)

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 $R'$ is stored in the server; the original relation $R$, table $\text{Tab\_index}$ related to attributes and indices and table $\text{Tab\_bucket}$ containing indices information are stored in the client.

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 $A$, $D=\{v_1, v_2, \ldots, v_N\}$.
3) $N$: the amount of values in $D$.
4) $v_i$: any arbitrary value in $D$, $1 \leq i \leq N$.
5) $f_i$: the amount of tuples whose attribute value is $v_i$ (frequency of value $v_i$), $1 \leq i \leq N$.
6) $M$: the number of partition buckets.
7) $B_{ij}$: 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 of relation $R'$ is returned, shown in table 2.

![Fig. 2 Index metadata of $R$](image-url)

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.

<table>
<thead>
<tr>
<th>Table 1. Relation $R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
</tr>
<tr>
<td>0101</td>
</tr>
<tr>
<td>0102</td>
</tr>
<tr>
<td>0201</td>
</tr>
<tr>
<td>0202</td>
</tr>
<tr>
<td>0301</td>
</tr>
<tr>
<td>0401</td>
</tr>
<tr>
<td>0402</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2. Encrypted relation $R'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>etuple</td>
</tr>
<tr>
<td>01000110</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Definition 1: Query probability \( p(q(v_i)) \). Repeat query \( n \) times on attribute \( A \) with a domain \( D=\{v_1, v_2, ..., v_m\} \), \( n(q(v_i)) \) 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, the statistical probabilities of \( v \) 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 \( B_1, B_2, ..., B_m \), \( q \) 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 \( q'(v_i) \). Cryptograph \( q' \) is corresponding with query \( q \) which result is \( q(v_i) \), while \( q'(v_i) \) is the result of \( q' \).

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 \( q' \) : SELECT * FROM table \( R^S \) WHERE depa'=4, and the cryptograph query result \( q'(02) \) contains four tuples with ids of 0201, 0202, 0401 and 0402.

Definition 4: Wrong hit tuple \( ET(q'(v_i)) \). Tuples containing the wrong result in \( q'(v_i) \), denoted by \( ET(q'(v_i)) \).

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

Definition 5: Unit wrong hit expectation \( E(B_j) \). Given partition buckets of attribute \( A \) as \( B_1, B_2, ..., B_m \), \( E(B_j) \) is the expectation of the number of wrong hit tuples in bucket \( j \),

\[
E(B_j) = \sum_{i=1}^{n} \left( \left( \frac{p(q(v_i))}{p(q(B_j))} \right) \times ET(q'(v_i)) \right)
\]

(1)

In Example 1, the unit wrong hit expectation of bucket \( B_1 \) is

\[
E(B_1) = \left\{ \frac{p(q(01))}{p(q(01)) + p(q(03))} \times ET(q'(01)) \right\} + \left\{ \frac{p(q(03))}{p(q(01)) + p(q(03))} \times ET(q'(03)) \right\} \times 0.23/ \left\{ 0.16 \times (0.23 + 0.16) \right\} = 1.41
\]

Definition 6: Collectivity wrong hit expectation \( E \). Given partition buckets of attribute \( A \) as \( B_1, B_2, ..., B_m \), \( E \) is the expectation of the number of wrong hit tuples in the hole domain,

\[
E = \sum_{j=1}^{m} \left( p(q(B_j)) \right) \times ET(B_j)
\]

(2)

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

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 assume the query probability on each attribute value is equal, viz. \( p(q(v_i)) = p(q(v_i)) \), \( i, j \in [1, N], i \neq j \). 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 \( v_i (1 \leq i \leq N) \), 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 \( n_i \), then the advantage of index can be measured by the formula below, and the less \( E \) is, the better of the index strategy.
\[
E = \sum_{j=1}^{M} \left( \sum_{i=1}^{n} p(q(v_i)) f_i^j - \sum_{i=1}^{n} p(q(v_i)) f_i^{j-1} \right).
\]

(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 \(E(B_i)\). 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 \(j^{th}\) attribute value is \(p(q(v_i))/p(q(B))\), \(1 \leq i \leq n\); the wrong hit number is

\[
|ET(q^i(v_i))| = f_i + f_{i+1} + \ldots + f_{i-l} + f_{ij}.
\]

(4)

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

\[
E(B_j) = \sum_{i=1}^{n} \left( \left( \frac{p(q(v_i))}{p(q(B_j))} \right) |ET(q^i(v_i))| \right)
= \sum_{i=1}^{n} f_i - \left( \sum_{i=1}^{n} p(q(v_i)) f_i / \left( \sum_{i=1}^{n} p(q(v_i)) \right) \right).
\]

(5)

Second, to compute collectivity wrong hit expectation.

\[
E = \sum_{j=1}^{M} p(q(B_j)) E(B_j)
= \sum_{j=1}^{M} \left[ \left( \sum_{i=1}^{n} \frac{p(q(B_j))}{p(q(v_i))} \left( \sum_{i=1}^{n} p(q(v_i)) f_i / \sum_{i=1}^{n} p(q(v_i)) \right) \right) \right]
= \sum_{j=1}^{M} \left( \sum_{i=1}^{n} p(q(v_i)) \sum_{i=1}^{n} f_i - \sum_{i=1}^{n} p(q(v_i)) f_i' \right).
\]

(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**

Crypograph 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: \(B=\{v_i\} / \text{ each bucket } /
\)

\(E / \text{ the wrong hit expectation under this partition } /
\)

\(BID / \text{bucket id, index value } /
\)

Probability Cryptograph Index Algorithm (PCI):

For \((k=1; k<=m; k++)\)

\(B_i=\{v_k\}\)

For \((k=m+1; k<=n; k++)\)

\{ if \(v_k<maxB_i\)

\{ \)

\(v_k\notin B_i; \)

\(maxB_i, minB_i, \text{JudgB}(1, maxB_i); \)

\} \)

if \(v_k=minB_m\)

\{ \)

\(v_k\notin B_m; \)

\(minB_m, \text{JudgB}(m, minB_m); \)

\} \)

if \((maxB_i<v_k<minB_m)\)

\{ \)

for \((l=2; l<=m-1; l++)\)

\{ \)

if \((minB_l<v_k<maxB_l)\)

\{ \)

\(v_k\notin B_l; \)

\(maxB_l, \text{JudgB}(l, maxB_l); \)

\(minB_l, \text{JudgB}(l-1, minB_l); \)

\} \)

if \((v_k< minB_{i-1} && v_k>maxB_i)\)
Create a set of random number $S = \{s_1, s_2, \ldots, s_M\}$

For ($i=1, i<= M_i, i++$)

$Bid_i=s_i$

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 $B_u$, */

$\min(E_u,E_{u+1})$;

Return $B_i$;

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 minimal probability in bucket $M$ should be moved to the left bucket. When there is case 2), 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 loops: one 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 $(n-M)$ times; the worst is case 3) occur in each time, and the executed time is $2(n-M)$, 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.

**Definition 7**: bucket disclosure coefficient $\lambda_{B_i}$: 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 $\lambda_{B_i} = \frac{1}{\sum_i n_i}$, where $n_i$ is the number of different values in $B_i$.

**Definition 8**: index disclosure coefficient $\lambda_I$: 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
\[ \lambda_{R^t} = \frac{1}{M} \sum_{i=1}^{M} \lambda_{B_i} \]
\[ = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{\sum_{k=1}^{n_i} f_k} \]  
\[ (7) \]

Definition 9: relation disclosure coefficient: if given relation \( R^t \) with \( k \geq 1, k \in R \), when the query result is random tuples in relation \( R^t \), the probability of guessing right for attractors is called the index disclosure coefficient, labeled as \( \lambda_{R^t} \).

In Example 1, indices \( I_{id}, I_{depa} \) and \( I_{salary} \) in Table 2 are corresponding to the 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 result is all the values in \( B_1 \), the probability for attractors 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 \( 3, 2, 3 \), according to definition
\[ \lambda_i = \frac{1}{N} \sum_{v=1}^{N} \lambda_{B_{jv}} \]  
\[ \lambda_{I_{id}} = 4/9 \]  
\[ \lambda_{I_{depa}} = 7/24 \]  
\[ \lambda_{I_{salary}} = 5/9 \]  

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 \( R^t \) 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>
<thead>
<tr>
<th>Index</th>
<th>( I_{id} )</th>
<th>( I_{depa} )</th>
<th>( I_{salary} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_{B_1} )</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>( \lambda_{B_2} )</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>( \lambda_{B_3} )</td>
<td>1/3</td>
<td>1/4</td>
<td>1</td>
</tr>
<tr>
<td>( \lambda_{I_{id}} )</td>
<td>1/2</td>
<td>1/4</td>
<td>1/3</td>
</tr>
<tr>
<td>( \lambda_{I_{depa}} )</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>( \lambda_{I_{salary}} )</td>
<td>1/2</td>
<td>1/4</td>
<td>1/3</td>
</tr>
</tbody>
</table>

Table 3. Disclosure coefficient values

Supposing tuple \( T_v \) corresponds to \( B_i \) in \( I_t \),
\[ F_i^{(j)} = \sum_{u=1}^{u_j} f_u \]  
\[ \lambda_{I_{id}} = 4/9 \]  
\[ \lambda_{I_{depa}} = 7/24 \]  
\[ \lambda_{I_{salary}} = 5/9 \]  

According to formula (9), the relation disclosure coefficient of \( R^t \) in the example above is
\[ \lambda_{R^t} = \frac{1}{N} \sum_{i=1}^{N} (\frac{2}{\min(F_i^{(j)}) + 1}) \]  
\[ (9) \]

If there are two indices \( I_{depa} \) and \( I_{salary} \) in \( R^t \), the relation disclosure coefficient is
\[ \lambda_{R^t} = \frac{1}{7} \left( \frac{2}{3 + 1} + \frac{2}{2 + 1} + \frac{2}{1 + 1} + \cdots + \frac{2}{3 + 1} \right) \]  
\[ = \frac{2}{3} \approx 0.6667 \]  
\[ (10) \]

If there is only one index \( I_{depa} \) in \( R^t \), the relation disclosure coefficient is
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 $F_{ij}$, when the relation is certain, $F_{ij}$ 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](image)

In the second set of experiments, we create indices on the attributes senid, paraid and wordfreq of the same relation and test the relationship between $M$ (number of partitioning bucket) and index security, which is indicated by index disclosure coefficient $\lambda_i = \frac{1}{M} \sum_{i=1}^{N} I_i M F_i$ shown in Fig. 4.

![Fig. 4 Relation of $\lambda_i$ and M](image)

In the second set of experiments, we create indices on the attributes senid, paraid and wordfreq of the same relation and test the relationship between $M$ (number of partitioning bucket) and index security, which is indicated by index disclosure coefficient $\lambda_i = \frac{1}{M} \sum_{i=1}^{N} I_i M F_i$ shown in Fig. 4. The result shown in Fig.5 indicates that increase of number of indices reduces the relation security.
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:


