ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE
International Journal of Advanced Research in Computer and Communication Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 5, Special Issue 1, Feburary 2016

Encryption with Ordered Bucketization and Dynamic Query Forms

Shameema Thasneem[1], Pragisha K[2]

PG Scholar, Department of Computer Science, LBS College of Engineering, Kasaragod, India [1]

Assistant Professor, Department of Computer Science, LBS College of Engineering, Kasaragod, India[2]

Abstract: The current problem in Database As a Service (DAS) is how to maximize the efficiency of retrieving encrypted data from remote servers without compromising security. Bucketization is a privacy preserving method for executing SQL queries over encrypted data. In Ordered Bucketization (OB), plaintext-space is divided into p disjoint buckets that are numbered from 1 to p which are based on the order of the ranges that they cover. Here we propose an Encryption with Ordered Bucketization (EOB) to support range query efficiently while preserving high-level of security. How to let non-expert users make use of the relational database is a challenging topic. Traditional predefined query forms are not able to satisfy various queries from users on those databases. Here we also propose a novel database query form interface that is DQF which is able to dynamically generate query forms.

Keywords: Query form, user interaction, query form generation, bucketization, security.

I. INTRODUCTION

QUERY form is one of the best used user interfaces for querying databases. With the rapid development of web databases and scientific databases, the modern databases become very large and complicated. In natural sciences, the databases have hundreds of entities for chemical or biological data resources. Many web databases typically have hundreds or even thousands of structured web entities. So, it is very difficult to design automatic static query forms to satisfy various database queries on those complex databases.

Many database management and development tools provide several mechanisms to let users create customized queries on databases. The customized query creation is totally depends on users' manual editing. If the user is not familiar with the database schema in advance, those hundreds or thousands of data attributes would confuse him/her.

Data is a valuable asset in modern enterprise and the need to handle large amounts and types of data is essential. Client queries on unencrypted data return exact matches, requiring minimal processing on the client machine, as all access functions are handled on the server. When the remote data is encrypted, however, the client must often take steps to determine which of the data returned satisfy query criteria and filter those that do not. These unwanted records, often referred to as false positives. It leads to the consumption of substantially higher network bandwidth, and incurs a penalty at client in terms of computational overhead for unnecessary decryptions.

Encryption preserves privacy, but often at the expense of efficiency, and balancing the trade off between the two becomes paramount. The major objective is to minimize client side processing (e.g., by reducing false positives), consigning responsibility to the server when possible, while balancing data privacy concerns that arise when the remote server is untrusted. Bucketization reduce complications for the client that arise when retrieving encrypted data from a remote server. It partitions encrypted attributes into buckets, thereby disguising which records are requested. For example, if a client program needs to retrieve the data in the range between 100,000 and 200,000, it first calculates the numbers of buckets whose union is the smallest set which covers the queried range.

The client program sends the bucket numbers to the database server. The database server can search all the encrypted data whose bucket number is one of the received numbers. The server can then sends the data back to the client program. The client program decrypt it and can obtain the correct query result by filtering out the data that are not in the range. So this approach can be very efficient compared to the case where the client receives all the cipher text from the server and decrypts all data items to obtain the correct query result.

Therefore, this method can be very useful when users cannot store their data without encryption such as in a cloud computing environment.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

nCORETech

LBS College of Engineering, Kasaragod

Vol. 5, Special Issue 1, Feburary 2016

II. RELATED WORK

How the non-expert users use the relational database is a challenging topic. A lot of research works focus on database interfaces which help users to query the relational database without SQL. Query Form is one of the most preferred database querying interfaces. At present, these have been utilized in most real-world business or scientific information systems. Current works mainly focus on how to generate the query forms.

Customized Query Form: For the developers**,** visual interfaces are provided to create or customize query forms. The problem is that these are not provided for the end-users. An end-user may be unfamiliar with the database. If the database schema is very large, it is difficult to find appropriate database entities and to create desired query forms for him.

Automatic Static Query Form: Recently, automatic approaches are proposed to generate the database query forms without user participation. By using data driven method, first finding a set of data attributes, which are most likely queried based on the database schema and data instances. Then, based on the selected attributes, the query forms are generated. Another method is workload-driven method. Here, clustering algorithms are applied on historical queries to find the representative queries. The query forms are then generated based on those queries. One problem is that, if the database schema is large and complex, user queries could be quite diverse. So, even if we generate lots of query forms in advance, there are still user queries that can't be satisfied by any of query forms. Another problem is that, when we generate a large number of query forms, how the users find an appropriate and desired query form would be challenging.

Query Refinement: Query refinement is a technique used by most information retrieval systems. It recommends new terms related to the query according to the navigation path of the user. But for the database query form, a database query is not just a set of terms. It is a structured relational query,.

Dynamic Faceted Search: It is a type of search engines where relevant facets are provided for the users. If we are only considering the Selection components in a query,then these engines will be similar to our dynamic query forms. But a database query form has other important components like Projection component. Projection components cannot be ignored.

Active Feature Probing: It is the technique for automatically generating questions to provide appropriate recommendations to users in database search. It actually focuses on generating the appropriate questions to ask the user and DQF aims to select appropriate query components.

Database systems play a critical role in modern enterprise. Efficient data processing is essential not only for business and academic enterprise, but also for end–consumers. The Database As a Service (DAS) model focuses on research on querying encrypted data. The primary technological challenge in DAS concerns data confidentiality. In the DAS model, user data is hosted at the service provider. As most organizations and individuals consider their data a vital asset, the service provider must support appropriate security measures to guard confidentiality of the owner's data. In designing mechanisms, a key consideration to ensure confidentiality of databases is that of trust, i.e., how much trust an organization or individual places in the service provider . Symmetric & asymmetric are the two types of cryptography to provide data confidentiality. Now let us focus on managing encrypted data. Consider the database consisting of the following two relations:

TEACHER (tid, tname, salary, addr, did)
DEPT (did, dname, dch)

Fields in the TEACHER table refer to a teacher's id, name of teachers, salary, address and the department id for which the teacher works. Fields in the DEPT table refer to the department id, name of the department, and the department chair name.

In a DAS model, these tables are stored at the service provider. Assuming the service provider is untrusted, all instances of these relations are encrypted for storage on the server. Assume that data is encrypted at the tuple level. That is, respective rows of each table are encrypted as a single unit. In this case, the relational representation consists of a set of encrypted records. Based on the given relation, a tuple (a) and its encrypted counterpart (b) may look like:

a) {123456, Alice, 10000, 55 Address, MATH134}
b){U2FsdGVkX1/65ysG6FrR1seDAawYTIbhuVORbv1=}

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE
International Journal of Advanced Research in Computer and Communication Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 5, Special Issue 1, Feburary 2016

For instance, Alice may wish to to evaluate "total salary for employees who work for Bob". In SQL, this query is expressed in the following way:

```
SELECT SUM(T.salary) FROM TEACHER as T, DEPT as D
WHERE T.did = D.did AND D.dch = ''Bob''
```

So the query requests the encrypted form of the TEACHER and DEPT tables from the server. The client can then decrypt the tables and execute the query.

III. PROPOSED WORK

A. Dynamic Query Form(DQF)

DQF is a novel database query form interface, which is able to generate query forms dynamically. The soul of DQF is to capture preference of the user and rank query form components that assist him/her to make decisions. The query form generation is an iterative process and it is guided by the user. The system automatically generates ranking lists of form components at each iteration and then the user can add the desired form components into the query form. The ranking of form components is based on the captured user preference. Then the user can fill the query form and submit it to view the query result at each iteration. In this manner, a query form could be dynamically developed until the user is satisfied with the query results. Fig 1 shows the flowchart of dynamic query form.

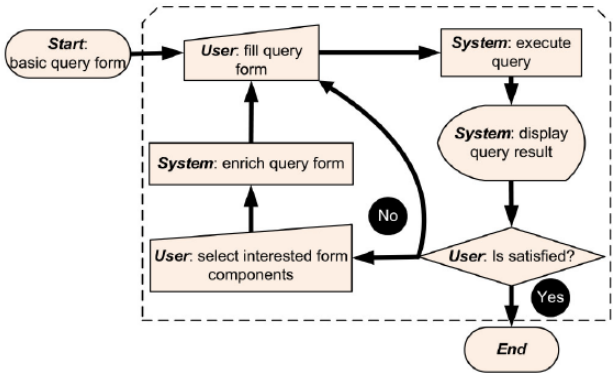

Fig.1. Flowchart of dynamic query form

B. Encryption with Ordered Bucketization(EOB)

The major goal of bucketization is to minimize the processing at the client side (e.g., by reducing false positives), consigning responsibility to the server when possible, while balancing data privacy concerns that occur when the remote server is untrusted. Encryption preserves privacy, but often at the expense of efficiency, and balancing the tradeoff between the two becomes paramount. Bucketization will reduce complications for the client that occure when retrieving encrypted data from a remote server. It is one of the technique for executing SQL queries over encrypted data on a DAS server. Actually the encrypted records are divided into buckets, where each bucket has an ID & a range that are defined by the minimum and maximum values in the bucket. The client contains indexing information about the range of each bucket on the server. Client queries can be then mapped to the set of buckets that contain any value that satisfying the conditions of the query. Fig. 2 shows the results of bucketization when the plaintext-space is [0,100]. In Fig. 2, the whole plaintext-space is divided into five buckets.

Consider the range query [30,70]. In the first case, the client program sends all the bucket numbers containing at least one plaintext in [30,70], which are 1,3 & 4 in fig 2. The database server finds all the encrypted records to which bucket numbers 1,3, or 4, are attached ,and returns them to the client program. It check what bucket numbers include the data that are not in the range where the user want to retrieve the data. The user then decrypts all the data on those bucket numbers & filters out the data that is not in the range. The data of bucket numbers 1 & 3 may have false positives. The other buckets do not have false positives because their scopes are in the range.

In the second case, the client program sends the minimum bucket number & maximum bucket number containing the plaintext in [30,70]. In fig 2, only 2 & 4 need to be informed to the server. All the encrypted records whose bucket numbers are in the range [2,4] are returned to the client program. In the client side, same procedure is necessary as the first case. The difference is that the bucket numbers that might have false positives are only the maximum & minimum bucket numbers. Here the numbers are 2 & 4.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

nCORETech

LBS College of Engineering, Kasaragod

Vol. 5, Special Issue 1, Feburary 2016

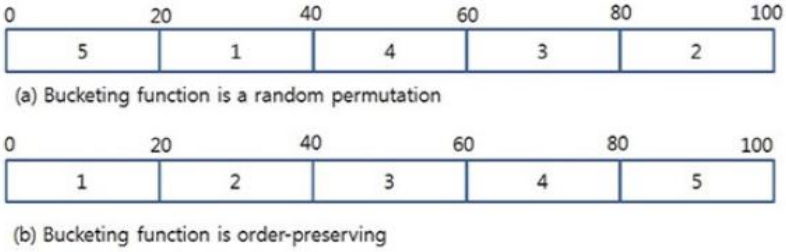

Fig. 2.An example of bucketization

IV. DISCUSSION

Data confidentiality is the primary technological challenge in DAS concern. Today, most organizations and individuals consider their data a vital asset. Therefore , the service provider must support appropriate security measures to guard confidentiality of the individual's data. Symmetric & asymmetric are the two types of cryptography to provide data confidentiality. AES is one of the symmetric algorithm which is extremely secure and is faster in both hardware and softwares. The approach Encryption with Ordered Bucketization can be very efficient compared to the case where the client receives all the ciphertexts from the server and decrypts all data items to get the correct query result. So, this method can be very helpful when users cannot store their data without encryption such as in a cloud computing environment.

Query forms are one of the best used database querying interface. In Active Feature Probing, it focuses on finding the appropriate questions to ask the user and the DQF aims to select appropriate query components. Customized Query Forms are provided for the professional developers who are known with their databases, not for end-users. In Automatic Static Query Form even if we generate many query forms in advance, there are still user queries that can't be satisfied by any of query forms. Another problem is that, when we generate a large number of query forms, how the users find an appropriate and desired query form would be challenging. So all these problems can be solved by using our dynamic query forms.

V. CONCLUSION

This paper proposed an Encryption with Ordered Bucketization scheme to support range query efficiently while preserving high-level of security compared to the existing methods. Also we proposed a dynamic query form generation approach which helps users to generate query forms dynamically.

REFERENCES

[1]. Younho Lee, Member, IEEE "Secure Ordered Bucketization",IEEE Transactions on Depend- able and Secure
[2]. Computing, vol. 11, no. 3.
[3]. Liang Tang, Tao Li, Yexi Jiang, and Zhiyuan Chen,Members, IEEE," Dynamic Query Forms for Database
[4]. Queries", IEEE, Transactions on Knowledge and Data Engineering, vol. 26, no. 9.
[5]. Ray Kresman, Advisor, Jong Kwan Lee,Tong Sun, "Bucketization technique for encrypted database".
[6]. Thorsten Joachims and Filip Radlinski, Cornell University, "Search Engines that Learn from Implicit Feedback".
[7]. M. Jayapandian and H. V. Jagadish, "Automating the design and construction of query forms," IEEE Trans.
[8]. Knowl. Data Eng., vol. 21, no. 10.
[9]. O. Goldreich, Foundation of Cryptography. New York, NY, USA: Cambridge Univ. Press, 2000.
[10]. C. E Shannon, "A mathematical theory of communication," Bell Syst. Tech. J., vol. 27.