Speed up Querying Encrypted Data on Outsourced Database

Kim Giau Ho
Le Quy Don Technical University
236 Hoang Quoc Viet street
Hanoi, Vietnam
(+84) 919 447 897
hkgiau@gmail.com

Ly Vu
Le Quy Don Technical University
236 Hoang Quoc Viet street
Hanoi, Vietnam
(+84) 978 366 574
lyvt@mta.edu.vn

Nam Hai Nguyen, Hieu Minh Nguyen
Academy of Cryptography Techniques
141 Chien Thang street
Hanoi, Vietnam
(+84) 989193571
hieuminhmta@gmail.com

ABSTRACT
The rapid development of cloud computing has appeared outsourced database and that is essential solution to reduce the cost for data owners (DOs). The user's data which is stored on the cloud will face many risks from attackers including service providers. To ensure the security of data, the DOs encrypt data before storing on the cloud. However, the encrypting before storing will increase the processing time to encode/decode records when querying to the database. Therefore, speeding up querying on the encrypted database is essential in an environment where data need to be encrypted before pushing to the cloud. In this paper, we propose a new method to improve the speed of querying on the encrypted database using parallel computing. The experimental results prove the effectiveness of our proposed method.

CCS Concepts
• **Security and privacy→Management and querying of encrypted data**

Keywords
Cloud computing; outsourced database; querying encrypted data; parallel processing

1. INTRODUCTION
Recent days, to manage data, almost companies choose to store data under database form. This storing method helps the DOs to access and share data to others easily. There are two ways to store data that are in-house data and on-line data. In the in-house storing method, the DOs manage data on their own server without sharing on the Internet. So that, the DOs need to have servers, operating systems, database management systems, and employees. Moreover, increasing of storing and processing data requires the DOs to pay higher cost for upgrading hardware, software, and training employees. In the on-line storing method, the DOs push data to a server on the Internet. In this case, the DOs can access data, share data everywhere on the Internet but they have to pay expense for hiring servers with more security methods.

ICMLSC '17, January 13-16, 2017, Ho Chi Minh City, Viet Nam

DOI: http://dx.doi.org/10.1145/3036290.3036299

With the development speed of cloud computing, companies have more solutions to manage data. One of them is Outsource Database Service – ODBS [1]. In this service, a Database Service Provider (DSP) will manage and maintain executions on data of the companies. DOs mine data through the methods of DSP. Therefore, DOs will reduce expense in investing resource and employees to manage and maintain database.

In many cases, the content of data includes sensitive information leading the DOs do not want to allow to access by unauthorized users. Otherwise, there are many attackers who try to get or destroy database illegally. Therefore, the DOs need to have a specific strategy to protect their own data.

To ensure the confidentiality of data, many researchers contribute data encryption solutions before pushing to outsourced database environment. However, data encryption methods increase the computation complex of accessing data effecting to database query performance. Therefore, choosing an encryption model to ensure the confidentiality and computation performance of the outsourced database needs to be considered. As shown in the Figure 1, the outsourced database model includes three components which are the data owners (client side), the database service provider (server), and the queriers (client side) [1]. In this model, the data owners create and modify database and outsource its database to the server and the users query to database on the server.

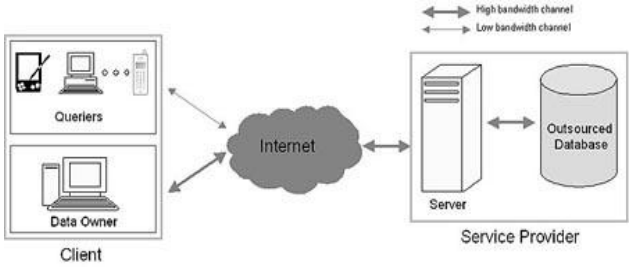

Figure 1. The model of outsource database [1].

To ensure computation performance of the outsourced database, we propose a parallel computation solution to speed up querying data on encrypted data. Q.Zhang et al. [2] proposes the Qscheduler tool to parallel query on database system. However, this method performs on plain text data and executes concurrent queries to the database by parallel computation. Ying-Fu Huang et al. [3] presents a parallel query method on plain text data where they perform the intersection, *JOIN*, *SORT*, *GROUP*, etc. operators of data tables. The work of Samaraddhi Shastri [4]

displays a speeding up method of querying on encrypted data. However, in their work, the parallel processing is applied for binary searching. To do that, they need sorting encrypted data before querying by users.

In this paper, we present the parallel computation method on querying encrypted data. Because the query responds from an encrypted data are cipher texts. So that, to get plain text responds, the server of the DO has to decrypt one by one database record for clients. If the number of records is huge, the processing time to get query responds is long. Our proposed method aims to reduce processing time significantly to decrypt records when querying a huge number of records.

The rest of this paper is constructed as follows. Section 2 presents some related works of the encrypted data query problem; our proposed work will be showed in Section 3; Section 4 presents experimental results and Section 5 concludes our paper.

2. RELATED WORKS

There are many published works related to the approaches of encrypted data query in which it can divide in three approaches. Firstly, using metadata or structure of tree (like XML) stores additional information to support querying on encrypted data [5, 6]. Secondly, using mediate servers translates query sentences [7, 8]. Thirdly, querying directly is executed on encrypted data but this method only supports some query forms [9, 12].

The work of Hacigumus et al. [5] proposes a solution to perform query on the encrypted database where a program named as query translator is used to translate query from plain text query of users to cipher text query. In their work [5], the data is encrypted before storing on the server of SP. Moreover, data has metadata which are indexes allowing performing query on data at the server of SP without decrypting data. The DO uses metadata to translate the query of clients into appropriate query to perform on the server. Clients will receive results when the server of SP returns query. Based on the metadata, queries divide into two components which are queries at the server side of encrypted data and queries at the client side. Queries at the client side perform on a client machine through a DO's machine where the DO filters query results and returns to clients. However, the drawback of this method [5] is that the high expense for storage and re-calculating after updating database. Moreover, the combination of queries such as SUM, COUNT, AVERAGE, etc. cannot perform.

R. Brinkman et al. [6] present a searching method on encrypted data by comparing the encrypted XML data tags and outsourced data. In their work, the data is divided into blocks and these blocks are encrypted before pushing to server. The DO stores encryption information to decrypt in processing of respond queries. In the searching process, the needed data sequences are encrypted to compare with tokens and determine the location of that data sequence. In the getting data process, the encryption responds are decrypted based on the information which is stored by the DO. This method is only performed searching on query sentences that match the XML data tags without processing data content in the tags.

In the work of A. Popa et al. [7], they display an encryption model and a method to perform query on an encryption database using the mediate server named as CryptDB proxy. The CryptDB proxy stores private keys, database diagrams, and current encryption layers of each column. The database server stores an incognito diagram in which table names and column names are substituted by identification names. Data is encrypted by clients and the CryptDB uses some supported tables. The CryptDB provides some user defined functions to servers that allow the servers calculating some specific operators on encrypted data. The advantage of the CryptDB is allowing the servers calculating on encrypted data without decrypting it and resulting by encrypted data. The CryptDB is an mediate system which provides high performance encryption for online applications on database management systems. It can perform the huge number of Structure Query Language (SQL) queries on encrypted data without any modification inside the database management system servers and results are decrypted by trusty clients. It can work with almost SQL databases with low expense due to the ability of changing encryption models automatically based on the number of queries to choose and ensure enough executions on database quickly.

The CryptDB can address almost but not entire SQL query types on the encrypted database and guarantee the private property of data. Moreover, the processing time of CryptDB is low. However, the CryptDB needs a strong trusty proxy so that attackers can access to the CryptDB relying on a weak proxy. In that case, attackers can decrypt database records to archive entire data information. Therefore, this model depends widely on the secure level of the proxy server.

3. METHODOLOGY

3.1 Storage Model of Encryption Database

The DO encrypts data before storing on an outsourced database. To execute calculations in querying, the DO server need to know the structure of database. Therefore, the DO server stores data tags in the XML file to match the structure of encrypted data. Instead of storing the plain text of table names, field names, the DO uses a hash function to encrypt these values into cipher texts. In this case, attackers or SP cannot have the structure of the database of the DO. The model is shown in Figure 2 where DOs encrypt data and create indexes before pushing to outsource database.

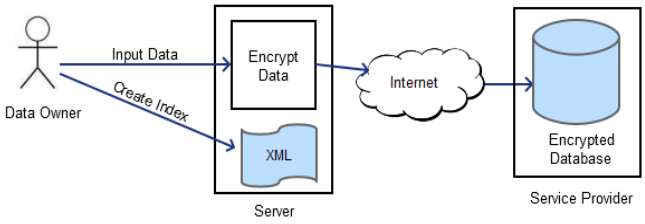

Figure 2. The model of encrypted database storage.

A XML file is a document structured by tags. By using tags, the XML file can create a database structure that allows to access directly or generate a searching tree. The XML file is supported in almost programming languages with parser packets such as SAX, DOM, and SRAX. The structure of the XML file is a text form leading the XML file having small size and low processing time. For one database query, the DO server only relies on the field names, table names in the XML file instead of using a hash function to calculate field names table names. So that, the processing time become lower.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tables>
    <Employee name="079711ea 16f37fe4 22587534 46c08153 51870043">
        <EmployeeID>64cdfdca c10cb6e6 09f0f808 e30a9a71 df67cc36</EmployeeID>
        <Name>709a2322 0f2c3d64 d1e1d6d1 8c4d5280 f8d82fca</Name>
        <Birthday>1d87df58 b83b360a d4033d3c df062a7b 87359cd6</Birthday>
        <Sex>e301dd60 62f7e9a7 9975fe8e 2d0ba916 94c4dbc3</Sex>
        <Address>d70f93df 5e8f9b55 be44fbee e9d20397 2e3383d4</Address>
    </Employee>
</tables>
```

Figure 3. Structure of XML file.

This ability of the XML file is proved in Figure 3. The Figure 3 presents the structure of a table in the XML file named as *Employee* (*EmployeeID*, *Name*, *Birthday*, *Sex*, *Address*). In this case, the Employee table is structured as 1-1 mapping shown in Table 1. Therefore, the DO stores the table name as: "*079711ea 16f37fe4 22587534 46c08153 51870043*" instead of storing the name as *Employee*. This storage information is similar for fields in the table.

Table 1. Structure of Employee table in XML file

Employee	079711ea 16f37fe4 22587534 46c08153 51870043
EmployeeID	64cdfdca c10cb6e6 09f0f808 e30a9a71 df67cc36
Name	709a2322 0f2c3d64 d1e1d6d1 8c4d5280 f8d82fca
Birthday	1d87df58 b83b360a d4033d3c df062a7b 87359cd6
Sex	e301dd60 62f7e9a7 9975fe8e 2d0ba916 94c4dbc3
Address	d70f93df 5e8f9b55 be44fbee e9d20397 2e3383d4

3.2 Query on Encryption Database

If a client wants to query to a database, the client will send a request to the database server and the server replies a respond as a plain text. The simple approach in querying encrypted database is downloading entire data into the DO's server then decrypting one by one record to execute query. However, this method spends huge storage resource and processing time. The other approach is querying related data then decrypting records to calculate these queries. Querying on the encrypted database has to match the tables of a XML file to guarantee the result of queries correctly.

Some kinds of queries as *SELECT*, *INSERT*, *UPDATE*, and *DELETE* can perform directly on the encrypted data while combination queries need to modify before executing on the encrypted data. In this paper, we perform queries as *INSERT*, *UPDATE*, *DELETE*, *SELECT* combines with *SUM*, *AVERAGE*, *MAX*, *MIN*, and so on. For example, a client wants to update a new address of an employee who has the code as "01". The query form on plain text data and cipher text data are presented in equation 1 and equation 2:

$$\text{UPDATE Employee SET Address = <new data>}\quad \text{WHERE EmployeeID= '01'} \tag{1}$$

$$\text{UPDATE t SET f5 = Enc(<new data>, key)}\quad \text{WHERE f1= Enc('01', key)} \tag{2}$$

where *t* is accessed data table in the XML file corresponding to the "*Employee*" tag, f_1 corresponding to the "*EmployeeId*" tag; f_5 corresponding to the "*Address*" tag; *Enc(data, key)* is a encrypted function with the determined key. The Figure 4 describes the process of querying to the encrypted database of a user based on XML tags.

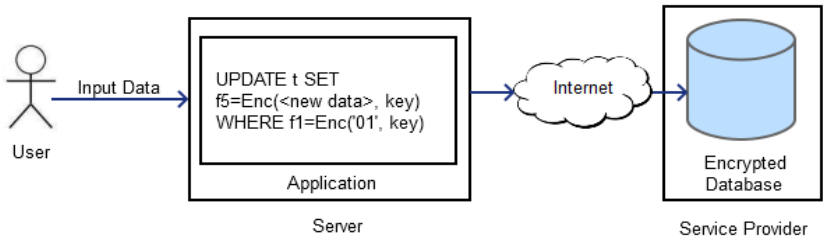

Figure 4. The model of querying data on outsourced encrypted database.

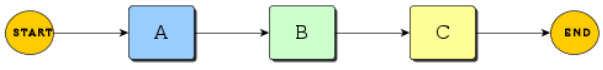

Figure 5. Sequent processing model.

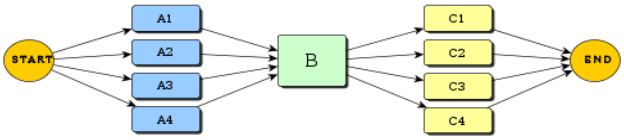

Figure 6. Parallel processing model.

3.3 Our Proposed Model

3.3.1 Parallel processing

Parallel processing is a computation process where multiple tasks are executed at the same time. There are two kinds of parallel processing models as divided data and divided tasks. The divided data model is applied to problems which have to deal with large data and each part of data can executed separately, then the result is the summarize of each part. This model is simple but it requires no dependency between data blocks. The divided tasks model is used to divided entire process into separate tasks where each task has ability to execute independently. The design of this model is more complicated than previous because we may have to re-design sequence algorithm to suit with the requirement which has multi tasks performing separately. The result is the synthetic of all tasks.

Parallel processing is used mainly in high performance computations. The recent years, parallel computation is considered in almost problems which need high performance. The Figure 5 and Figure 6 show the sequence and parallel computation model correspondingly. In these Figures, the computation process

has three tasks named as *A*, *B*, *C*. In the Figure 5, each task is executed sequentially which means that after finishing the task *A* then performing the task *B* and after finishing the task *B* then performing the task *C*. Therefore, the processing time of entire process is the sum of processing time of the task *A*, *B*, and *C*. While in the Figure 6, the task *A* and *C* are divided into four sub-tasks named as A_1, A_2, A_3, A_4 and C_1, C_2, C_3, C_4 correspondingly. Therefore, the processing time of entire process approximates processing time of the task *B* with one by four processing time of the task *A* and *C*. Therefore, using parallel computation helps reducing processing time significantly. The parallel computers are divided based on supporting ability of its hardware. For example, multi-cores computers and multi-processes computers which have a multi-tasks processing component in a single machine while grid computing uses multi-computers to process multi-tasks.

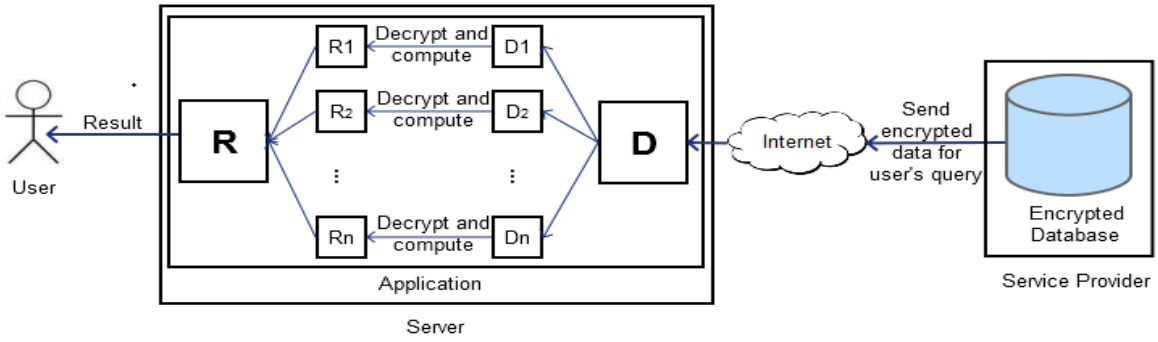

Figure 7. Parallel decryption on encrypted database.

3.3.2 Our proposed methodology

When a client queries to the encrypted data of an outsourced database, the system will map to table names in the XML file to re-write the query sentence then decrypting result records. This decryption process is shown in Figure 7. Decrypting records execute with the huge number of records leading high processing time. To reduce processing time, the DO can allow performing parallel decrypting records on CPU or GPU.

For example, when a client wants to display the *Employee* table or process on feedback data, the client has to decrypt one by one record. As we presented above, if there are the huge number of records, the decryption time is extremely large. Therefore, in this paper, we divide the set of feedback data *D* into subsets and the decryption is executed on each subset concurrently shown in Figure 7 where the feedback data *D* is divided into D_1, D_2, D_3,…, D_n depending on the number of processes in our case $n = 2$. The data blocks D_i is decrypted by parallel computation to get raw data named as R_1, R_2, …, R_n. This computation can be applied on the combination of queries on SQL such as SUM, AVERAGE, COUNT, etc.

The parallel algorithm for decrypting data is shown in the table 2 with *SELECT* command. The execution is on the equation 3.

$$SELECT\ f_1, f_2, \dots, f_n\ FROM\ t \quad (3)$$

where t is the data in the XML file corresponding to Employee tag; f_1, f_2, …, f_n corresponding to data fields. The respond queries are in the set *D* which includes the encrypted records. In the case of the size of the set *D* is huge, the processing time is the time for consequent decryption of records. In the algorithm, the set *D* is divided into *n* subsets for parallel decryption on each subset. Therefore, the decryption time is max(TD_1, TD_2, …, TD_n) where TD_j is the decryption time of the D_j set. The Section 4 shows that using our parallel processing method on the encrypted database will reduce significantly processing time comparing with consequently decrypting each encrypted record.

Table 2. Algorithm of query to encrypted database by parallel processing

```
Dataset D
//Main function
D ← Execute(SELECT f1, f2, …, fn FROM t)
D = D1 ∪ D2 ∪ ... ∪ Dn
Thread T1 = new Thread(F1)
Thread T2 = new Thread(F2)
…
Thread Tn = new Thread(Fn)
R ← Join(T1,T2,…,Tn)
Return R
//Function F1
Foreach record r in D1
        r'← Decrypt(r)
R1 ← Calculate in r'
Return R1
//Function F2
Foreach record r in D2
        r'← Decrypt(r)
R2 ← Calculate in r'
Return R2
…
//Function Fn
Foreach record r in Dn
        r'← Decrypt(r)
Rn ← Calculate in r'
Return Rn
```

4. EXPERIMENTS

To measure our parallel model on querying an encrypted database, we test on the computer environment as Corei3-1.5GHz, 4GB memory. The querying data program is executed by sequent algorithm and parallel algorithm. The number of threads is two. We implement on the C#, Visual studio 2013 program language and the database management SQL. The encryption algorithm is used as the AES. The record has five data fields (as described in section 3.1). For each scenario, we execute ten times and plot processing time on mili-second (ms).

The results show that when the number of database records is small as 282 records, the processing time of sequent and parallel decryption has a little different as in Figure 8 where the processing time for sequent decryption is about 1.4 times of parallel decryption as shown in the Figure 8. However, when the database becomes larger as about 5140 records and 10120 records as shown in the Figure 9 and Figure 10 respectively, the processing time sequent decryption are up to 1.7 and 1.8 times of parallel decryption correspondingly. Therefore, the parallel processing decryption is meaningful with the huge number of encrypted records. When using outsource database to store data becomes more popular, the server has to deal with thousands of queries at the same time. So that, processing time for encrypted database is the one of strong factors that effectives to the performance of accessing database from clients. These experimental results prove that using parallel processing on querying to encrypted database is an optimal approach to address the performance of outsources database system.

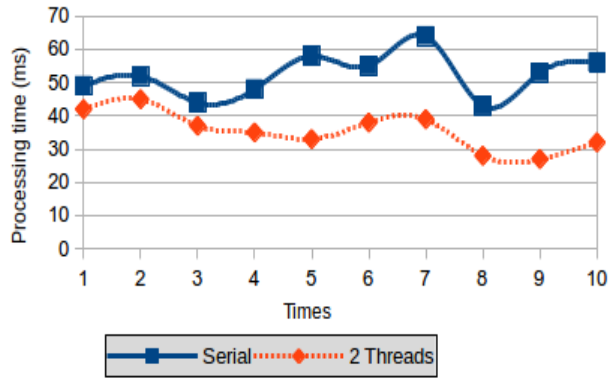

Figure 8. Result of querying on 282 encrypted records.

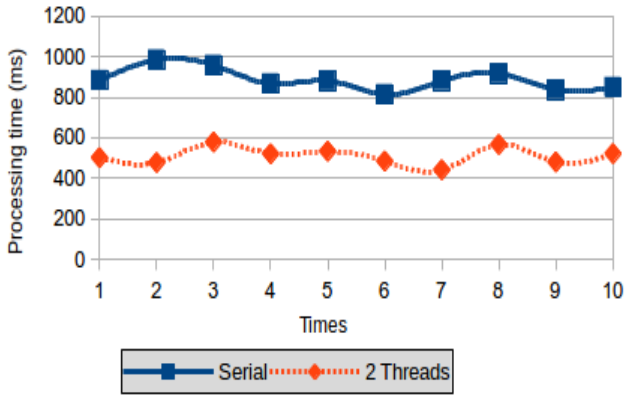

Figure 9. Result of querying on 5140 encrypted records.

Beside of relying on the ability of the computer, using the number of threads is considered carefully. Because if we use many threads, we have to pay much more processing time to divide data before executing and conclude data after executing. In some cases, the delay time is large even though it makes the processing time of parallel computation becoming bigger than sequent computation.

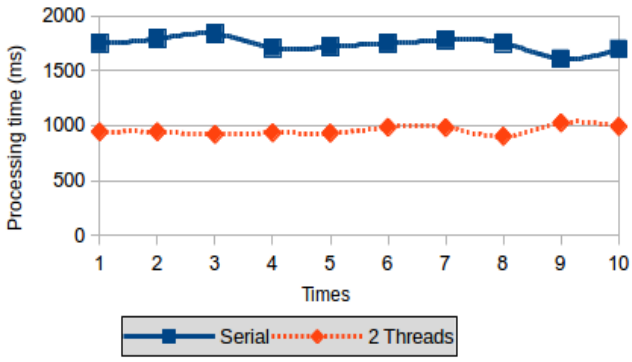

Figure 10. Result of querying on 10120 encrypted records.

5. CONCLUSION

The security of database is ensured by encrypting data before pushing it on an outsourced database server. However, encrypting data makes querying time on that data becoming longer due to decrypting that data before using it. Therefore, protecting data from hackers means losing more time to query data. Therefore, our paper proposes a method to reduce query time on the encrypted database by parallel decryption on the responding data queries. The experimental results show that our method is meaningful in reducing processing time in querying encrypted database.

However, to improve the speed of query, we need more solutions to support in which the encryption algorithm is one of factors that can delay computation processes on an encrypted database. So that, the selection of encryption algorithms needs to appropriate with the requirement of processing time for decrypting records. One of solutions is using a symmetry encryption algorithm due to the rapid of encryption and decryption. However, this solution needs more cost for calculating on encrypted database. Because a server has to decrypt records to calculate on the plain text of records, then encrypt records to respond results to clients. If using a homomorphism algorithm is better for calculating on encrypted data but increasing processing time on encryption and decryption data. Therefore, the method of combination many encryption algorithms to separate query cases corresponding to encrypted data are developing in our feature works. As described above, our feature work focuses on the solution of parallel processing for the combination of some algorithms to classify queries corresponding to each encryption algorithm. For example, text data is encrypted by the AES algorithm or numeric data is encrypted by homomorphism algorithms, etc. where homomorphism algorithms can reduce time processing for querying.

6. REFERENCES

[1] http://sprout.ics.uci.edu/past_projects/odb/.

[2] Q. Zhang, S. Li, and J. Xu, "QScheduler: A Tool for Parallel Query Processing in Database Systems," in *Engineering of Complex Computer Systems (ICECCS), 2014 19th International Conference on*, 2014, pp. 73-76.

[3] Y.-F. Huang and W.-C. Chen, "Parallel Query on the In-Memory Database in a CUDA Platform," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2015, pp. 236-243.

[4] S. Shastri, R. Kresman, and J. K. Lee, "An Improved Algorithm for Querying Encrypted Data in the Cloud," in *Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on*, 2015, pp. 653-656.

[5] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, 2002, pp. 216-227.

[6] R. Brinkman, L. Feng, J. Doumen, P. H. Hartel, and W. Jonker, "Efficient tree search in encrypted data," *Information System Security Journal*, vol. vol.13, pp. 14-21, 2004.

[7] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Processing queries on an encrypted database," *Communications of the ACM*, vol. 55, pp. 103-111, 2012.

[8] B. H. Chen, P. Cheung, P. Y. Cheung, and Y.-K. Kwok, "CypherDB: A Novel Architecture for Outsourcing Secure Database Processing," p. 1, 2015.

[9] Z.-F. Wang and A.-G. Tang, "Implementation of encrypted data for outsourced database," in *Computational Intelligence and Natural Computing Proceedings (CINC), 2010 Second International Conference on*, 2010, pp. 150-153.

[10] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Advances in Cryptology–CRYPTO 2012*, ed: Springer, 2012, pp. 850-867.

[11] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in *Proceedings of the VLDB Endowment*, 2013, pp. 289-300.

[12] K. Mallaiah and S. Ramachandram, "Applicability of Homomorphic Encryption and CryptDB in Social and Business Applications: Securing Data Stored on the Third Party Servers while Processing through Applications," *International Journal of Computer Applications*, vol. 100, 2014.