Turkish Journal of Computer and Mathematics Education Vol.12 No.12 (2021), 4527-4534

Research Article

Processing Queries over Multidimensional Encrypted Data

Norah Saud Al-Musib[1], Hedi HAMDI[2], A. A. Abd El-Aziz[3]

[1]College of Computer and Information Sciences Jouf University Al-Jouf, Saudi Arabia
[2]College of Computer and Information Science Jouf University Al-Jouf, Saudi Arabia
[3]College of Computer and Information SciencesJouf UniversityAl-Jouf, Saudi Arabia Faculty of Graduate Studies for Statistical Research, Cairo University, Egypt
[1]401205535@ju.edu.sa,[2] hhamdi@ju.edu.sa, [3]aaeldamarany@ju.edu.sa

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 May 2021

Abstract: In new computing architectures, more and more data is outsourced to remote storage providers. When data outsourced, sensitive information becomes vulnerable to theft. Data owners want to make sure their data is safe from breaches by third parties, insider threats, and untrusted service providers. When queries are run over encrypted data, a certain level of secrecy is maintained. However, how to efficiently query encrypted multidimensional for data stored in an untrusted server environment residue a research challenge. In this paper, we propose an effective technique based on encryption by different keys in multidimensional mode to execute computations on data encrypted for multi principals. The proposed technique run queries directly without decrypting data as sensitive data are protected and insured .

Keywords: Data security, encrypted data, query processing, multi principals, CryptDB

1. Introduction

Users are more likely to outsource their encrypted data to cloud providers due to the explosion of cloud platforms [1]. There are two reasons for this occurrence: One is because cloud systems provide the capacity to accommodate large amounts of data and offer additional facilities (such as cloud computing), reducing the storage and computing pressures on customers [1]. The other is that cloud systems provide users with accessing interfaces that make it easy for them to access their results [1]. However, it has many problems users' personal information in their data would be revealed to cloud servers if they send their data directly to cloud services, resulting in privacy leakage[1].

The outsourcing issue will arise not only as a result of outside hackers but also as a result of an insider, who may be a business employee or a data server administrator [2]. An encryption scheme can be used to solve such a problem, it is a form of construction in which data is kept in a way that it cannot be accessed without the use of external hardware [2]. The issue with encryption is that there is no way to run queries on encrypted data without first decrypting it [13]. To query encrypted data, either they must be decrypted, potentially exposing sensitive information, or special techniques for querying encrypted data with minimal querying capacity and reliability must be used [3]. In recent years, a variety of software-based and hardware-based cryptographic solutions have been developed for performing SQL queries over encrypted data with decryption in advance [4].

The main goal of this paper is to propose an effective technique based on encrypted by different keys in multidimensional mode to execute computations on data encrypted for multi principals run queries directly without decrypting data as sensitive data are protected and insured.

This paper is structured as follows: in section 2, we discuss the related work. In section 3, CryptDB is overviewed. Section 4 displays the proposed algorithm with examples of query execution. In section 5, the evaluation and comparison between with the proposed algorithm and the recent algorithms are presented. In section 6 the conclusion and the Future work are shown.

2. Related Work

In this section, we will discuss some of the literature related to the proposed problem in terms of systems executing queries on encrypted data, proposed encryption algorithms for preserving data when queries on encrypted data in databases, and multi-principle limitations in cryptDB

A. Systems Execute Queries on Encrypted Data

Popa et al. [5] (CryptDB) is the first software that can run SQL queries over encrypted data. It is a functional software that ensures server security using database management systems. The software serves as a proxy to secure contact between the application server and database server, implementing the concept. Its key aim is to protect data security against those who have control and access to it [2]. CryptDB cannot run computations on encrypted values using different keys at DBMS [3]. However, our technique is based on encryption by different keys in multidimensional mode to execute computations on data encrypted for multi principals.

Liangmin Wang et al. [9] proposed the concept of SHAMC, a safe, multi-cloud database system. SHAMC can run SQL queries directly over encrypted data, significantly improving the cloud database's availability. It tackles the concerns of service outages and vendor lock-in. The evaluation's findings demonstrate that SHAMC can take the feature of multi-cloud to effectively conduct queries and can be implemented on the commercial cloud at a competitive cost. SHAMC supports most arithmetic operators and other query operators which do not manipulate values. In our proposed technique, we used cryptDB, because it is more secure according to the encryption level and advanced encryption can provide a higher level of data security.

Ansar Rafique et al. [11] implemented CryptDICE, The framework allows multiple types of search and aggregate queries to be performed over encrypted data for a wide variety of different NoSQL databases, with almost no alteration to the underlying database engine and with minor modifications using built-in client-side annotations. CryptDICE's lightweight service eliminates the management complexity of integrating User Defined Functions (UDF) to every underlying database functionality directly into the database engine. The results of the assessment confirm that CryptDICE dramatically reduces the time and effort for production to allow data encryption and facilitate multiple forms of immersive secure search queries. Also, efficiency enhancements, efficiency is stated to be moderate. while most server-side implementations in CryptDB run as user-defined functions (UDF).

B. Encryption Algorithms

In the following three encryption algorithms are explained to perform queries on encrypted data. In the proposed technique, depend on cryptDB the encryption algorithms

Sultan et al. [6] proposed a practical and efficient model for running SQL queries over encrypted data using a single randomized encryption algorithm, the Advanced Encryption Standard AES-CBC. Their architecture seeks to reduce the number of encrypted documents collected from the cloud to a limited number of applicants for the user's query. Their studies show that their model is capable of reducing decryption processes to under 30% of the total number of encrypted data in a table.

JIANCHAO et al [10] proposed a secure query scheme for data outsourcing on cloud platforms. Users' data is encrypted in their scheme focused on all possible requests to satisfy users' complex query demands. Also, to deal with the frequency attack that exists in deterministic encryption, a double AES encryption approach is used. A neighbor rows sharing system is built in their scheme to protect users' privacy as their data is modified. The cost in their scheme is lower than the cost in CryptDB, because the data is encrypted by ASHE in their scheme.

Sultan et al. [16] proposed a bit-based model to perform various relational algebra operators over encrypted DB in the cloud without decrypting the files, and they proposed a realistic model and succeeded in minimizing the range of the obtained encrypted information. The intention is to divide attributes into critical and non-critical attributes, with only critical attributes being encrypted. The table owner predefines the feasible partition domains on which the tuples will be encoded into bit vectors before encryption for each critical attribute. They keep the bit vectors in a separate column in the cloud's encrypted table. They used these bits to get just a fraction of encrypted documents that are candidates for a particular query.

C. Multi-Principle Limitations in CryptDB

Algorithms in Rasha Refaie et al. [7, 13] based on cryptDB, suggested two safe techniques for searching over encrypted data without decrypting all records. Their proposed algorithm overcomes the computational constraints of values encrypted for multi principles but they used one key for encryption and execute one type of conditions to run a query over encrypted data. we proposed atechnique based on encryption by different keys in multidimensional mode to become more effective.

Kevin Foltz et al. [8] build on previous work by adding an Oracle application. Via the CryptDB proxy, this function only involved converting raw SQL queries to the database. Their work demonstrates that external

functionalities such as stored procedures, views, and multi-user access controls can be enforced using encrypted data without requiring any implementation modifications to the ERP program code. These findings showed that an Oracle ERP could be run on encrypted files.

3. CryptDB

Encrypting data for storage on a server is a good option for protecting confidential data from disclosure where the data is not completely under the protection of the data owner. Though it is not arbitrary to run SQL queries over encrypted data, ciphertexts may sacrifice some of their original characteristics, such as length and format; plaintext comparison and arithmetical operations may no longer or only partly endorse ciphertexts [4].

In CryptDB [5], SQL uses a well-defined series of operators, most of which supports effectively over encrypted files. CryptDB uses a SQL-aware encryption technique to perform SQL queries over encrypted data and offers query-based encryption that can be adjusted to ensure that the least amount of information is revealed to satisfy the encryption algorithms. It protects against the application server, proxy, and DBMS server infrastructures being hacked randomly by preventing a curious DBA or other foreign intruders from knowing private data. CryptDB allows the DBMS server to run SQL queries on encrypted data just as if it were plaintext data, and it provides clients with a high level of transparency.

CryptDB has limitations on the server-side. It cannot perform computations on encrypted data for multi principals, because it is encrypted with various keys [12]. Traditionally, the computation is performed after the data has been decrypted. It may be practical for some computations, but it may be too expensive for others, such as large-scale aggregates [12].

The design of CryptDB is the foundation for our proposed algorithm. The architecture of CryptDB is divided into two parts: the first is a database proxy, which is a trusted server that holds the database schema as well as a hidden master key [13]. It functions as a middle layer that encrypts and decrypts all data and modifies certain query operators while maintaining the query's semantics [13]. The second part is that CryptDB's unmodified DBMS will query an encrypted database as if it were plaintext [13]. The proxy controls all traffic to and from the database and encrypts all data sent to the DBMS using hidden keys [13,14,15]. CryptDB allows users to connect encryption keys to their passwords. Only the user's password will decrypt his authenticated data object in CryptDB. As a result, if all servers are hacked, an attacker will be unable to decode the data of any non-logged-in person, as well as a database administrator, will be denied access to decrypted data [12].

If the program has the authority of all principals in question, CryptDB has difficulty running computations on values encrypted using separate keys at the DBMS [12]. Traditionally, the calculation is done after the data has been decrypted [13]. It might be feasible for certain computations, but it is prohibitively costly for large-scale aggregate computations [13].

Data in CryptDB is encrypted in a layered, onion-like fashion [5]. These onions are made up of several layers, each of which is encrypted using a different algorithm [5]. The most protected layer of an onion is its exterior layer [5]. In each onion, a value has just one current layer [5]. For queries that provide contrast, the ORD onion is used to modify order leakage [5]. Order-preserving encryption (OPE) is a way of encrypting data while keeping its order [5]. As a result, if a column is encrypted with OPE, the server will run comparison queries and range queries.

In our algorithm, we assume that different keys are used to encrypt certain columns on multidimensional mode. The issue is that when the same value is encrypted with different keys, the effect is different values. As a result, range queries and queries involving contrast on a column containing values encrypted by several keys and the query has different conditions are not possible on the server. To use CryptDB to run this query, the query would be encrypted and forwarded to the DBMS registry to be executed. Since the column is encrypted with different keys, the server is unable to verify the data.

To solve the CryptDB's limitations, we are enhancing the proposed algorithm in [13] to execute the query directly in encrypted form without the need for decrypting all the data. The enhanced algorithm aims to overcome this problem by encrypting each query with different keys in multidimensional mode and with different conditions as Fig. 1. Therefore, the proposed algorithm deals with multidimensional mode and running each query on the corresponding encrypted data with the same keys and divided according to WHERE conditions, which is composed from "AND" and "OR" Boolean expression.

4. Proposed Algorithm

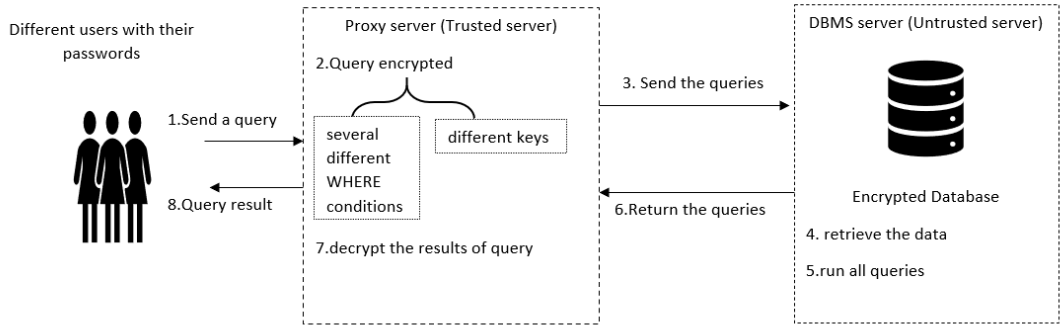

Fig. 1. The Proposed Algorithm Architecture.

Algorithm 1. Execute queries with composed condition

Input: user query
Output: query result

1. input query with different conditions
2. The proxy checks the query columns that were used in the search conditions.
3. **If** it is not on column
4. The proxy intercepts and **encrypt** the query
5. **send** the encrypted query to the DBMS server to retrieve data from Table2
6. **send** query results to the proxy
7. **decrypt** the results
8. **send** the result to the user
9. **Else if** authorized users
10. The proxy **encrypt** the query by using **different keys** and **different conditions**
11. **send** the encrypted query to the DBMS server to **run** all query on the corresponding data which encrypted by the same key and that contents the same conditions
12. **send** the results to the proxy.
13. **decrypt** results
14. **send** the result to the user

To explain the proposed algorithm, we have consider two tables; the first table called S**earch_Table** (Table1) is saved at the proxy server and the second table called **Encrypted_Table** (Table2) is saved at the DBMS server. **Encrypted _Table** is the encrypted form for **Search _Table** which has columns containing values encrypted by different keys for the same record.

Table1: Search _Table

ID	NAME	MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
1	Hind	500	0553621920	1
2	Muhammad	1000	0549823476	1
3	Maram	250	0563289471	2
4	Rahaf	600	0532968547	3
5	Sami	200	0521369710	3
6	Even	1000	0522399526	4

Table2 Encrypted _Table

ID	NAME	MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
1	Encrypted	Encrypted (K1)	Encrypted (K4)	1
2	Encrypted	Encrypted (K1)	Encrypted (K4)	1
3	Encrypted	Encrypted (K2)	Encrypted (K5)	2

4	Encrypted	Encrypted (K2)	Encrypted (K5)	3
5	Encrypted	Encrypted (K3)	Encrypted (K6)	3
6	Encrypted	Encrypted (K3)	Encrypted (K6)	4

When an authenticated user needs to search any data, the proxy will encrypt the query using different keys and several different "AND" or "OR" conditions. The proxy sends these queries to the DBMS server, which executes each query on the corresponding data, which is encrypted with the same keys and that matches the required condition. These queries will return the data from Table2 that the user is searching based on different conditions in multidimensional as in Fig. 1. This method doesn't need to decrypt all of the values in the encrypted column; instead, it only decrypts the values that the user wants in the query.

The proposed algorithms searching operation can be explained through the following examples:

Examples1:

SELECT NAME, MOBILE BILL, MOBILE NUMBER
FROM Table2
WHERE MOBILE BILL ≤ 500 AND MOBILE NUMBER = 0549823476;

The query execution steps for this example are as follows:

1. The proxy will encrypt this query using (K1, K4) to (Q1 and Q2), (K2, K5) to (Q3), and (K3, K6) to (Q4) based on WHERE conditionals using AND operation.
2. The proxy sends these four queries to the DBMS.
3. The DBMS runs each encrypted query on encrypted data, which is encrypted by the same keys, and the required condition is met based on different conditions in multidimensional to return exactly these records, which the user wants.
4. Then DBMS server returns the result (Table 3,4,5,6) and sends it to the proxy server.
5. The proxy decrypts these results and sends it to the user (Table7).

Q1:
SELECT NAME, MOBILE BILL, MOBILE NUMBER
FROM Table2
WHERE MOBILE BILL ≤ 500;

Table3: Result of Q1

NAME	MOBILE BILL	MOBILE NUMBER
Encrypted	Encrypted (K1)	Encrypted (K4)

Q2:
SELECT NAME, MOBILE BILL, MOBILE NUMBER
FROM Table2
WHERE MOBILE NUMBER = 0549823476;

Table4: Result of Q2

NAME	MOBILE BILL	MOBILE NUMBER
Encrypted	Encrypted (K1)	Encrypted (K4)

Q3:
SELECT NAME, MOBILE BILL, MOBILE NUMBER
FROM Table2
WHERE MOBILE BILL ≤ 500;

Table5: Result of Q3

NAME	MOBILE BILL	MOBILE NUMBER

Encrypted	Encrypted (K2)	Encrypted (K5)

Q4:
SELECT NAME, MOBILE BILL, MOBILE NUMBER
FROM Table2
WHERE MOBILE BILL ≤ 500;

Table 6: Result of Q4

NAME	MOBILE BILL	MOBILE NUMBER
Encrypted	Encrypted (K3)	Encrypted (K6)

All the above tables are decrypted by the proxy server in Table 7:

Table 7: Result from the Proxy Server

NAME	MOBILE BILL	MOBILE NUMBER
Hind	500	0553621920
Muhammad	1000	0549823476
Maram	250	0563289471
Sami	200	0521369710

Examples2:

SELECT MOBILE BILL, MOBILE NUMBER, BILL -NUMBER
FROM Table2
WHERE MOBILE BILL ≥ 600 AND MOBILE NUMBER = 0521369710 OR MOBILE BILL = 250

The query execution steps for this example are as the previous example, the proxy will encrypt this query using (K1, K4) to (Q1), (K2, K5) to (Q2 and Q3), and (K3, K6) to (Q4 and Q5) based on WHERE conditionals using "AND" or "OR" operations and different keys in multidimensional, then implementation of the rest of the processes as the previous example.

Q1:
SELECT MOBILE BILL, MOBILE NUMBER, BILL -NUMBER
FROM Table2
WHERE MOBILE BILL ≥ 600;

Table 8: Result of Q1

MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
Encrypted (K1)	Encrypted (K4)	1

Q2:
SELECT MOBILE BILL, MOBILE NUMBER, BILL -NUMBER
FROM Table2
WHERE MOBILE BILL = 250;

Table 9: Result of Q2

MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
Encrypted (K2)	Encrypted (K5)	2

Q3:
SELECT MOBILE BILL, MOBILE NUMBER, BILL -NUMBER
FROM Table2
WHERE MOBILE BILL ≥ 600;

Table 10: Result of Q3

MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
Encrypted (K2)	Encrypted (K5)	3

Q4:

```
SELECT MOBILE BILL, MOBILE NUMBER, BILL -NUMBER
FROM Table2
WHERE MOBILE NUMBER = 0521369710;
```

Table 11: Result of Q4

MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
Encrypted (K3)	Encrypted (K6)	3

Q5:

```
SELECT MOBILE BILL, MOBILE NUMBER, BILL -NUMBER
FROM Table2
WHERE MOBILE BILL ≥ 600;
```

Table 12: Result of Q1

MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
Encrypted (K3)	Encrypted (K6)	4

Tablen13: Result from the Proxy Server

MOBILE BILL	MOBILE NUMBER	BILL - NUMBER
Muhammad	1000	1
Maram	250	2
Rahaf	600	3
Sami	200	3
Even	1000	4

5. Evaluation and Comparison with the Developed Algorithm

The result reached through the examples that have been explained above show that the proposed technique doesn't need to decrypt all of the values in the encrypted column; instead, it only returns the values that the user wants in the query. If the condition contains more or less clauses, the same procedure is followed. Unlike the algorithm in [14], it generally executes the query using one key for encryption and cannot implement more than one condition in the same query on the encrypted data. Our proposed algorithm is more secure where the data is encrypted with different keys instead of one key for all data in order for the data to become more protected so that if the key used to encrypt certain data is discovered or known, the other data cannot be known, because it is encrypted with other different key This is why it is difficult to know the data when hacking or knowing one encryption key. Moreover, it is more effective and comprehensive, because when the user requests some data, he can get data from different columns and rows in the same table in multi-dimensional mode, unlike other techniques.

6. Conclusion

In this paper, a secure algorithm for searching over encrypted data was proposed. Our proposed algorithm would effectively eradicate the shortcomings of computations on values encrypted for multi principals with different conditions in multidimensional encryption. The architecture of CryptDB is the foundation for our proposed methodology. In the future work, we enhance the proposed algorithm to include the JOIN clause and different types of queries.

References

1. Tang, J., Fu, S. and Xu, M., 2019. An Effective Encrypted Scheme Over Outsourcing Data for Query on Cloud Platform. IEEE Access, 7, pp.66242-66250.
2. Maisura, M., 2018. ANALYZING CLIENT-SIDE ENCRYPTION IMPLEMENTED IN CRYPTDB. Cyberspace: Jurnal Pendidikan Teknologi Informasi, 2(1), p.69.
3. Almarwani, M., Konev, B. and Lisitsa, A., 2019. Flexible Access Control and Confidentiality over Encrypted Data for Document-based Database. In: 5th International Conference on Information Systems Security and Privacy. [online] ResearchGate, pp.606-614. Available at: <https://www.researchgate.net/publication/331775415_Flexible_Access_Control_and_Confidentiality_over_Encrypted_Data_for_Document-based_Database> [Accessed 26 March 2021].
4. Zhu, T., 2017. Executing SQL Queries over Encrypted Data: A Survey. International Journal of Computer Techniques, 4(3), pp.23-33.
5. Popa, R., Redfield, C., Zeldovich, N. and Balakrishnan, H., 2012. CryptDB: Processing Queries on an Encrypted Database. Communications of the ACM, 55(9), pp.103-111.
6. Almakdi, S. and Panda, B., 2019. A Secure Model to Execute Queries Over Encrypted Databases in the Cloud. In: 2019 IEEE International Conference on Smart Cloud (SmartCloud) Smart Cloud (SmartCloud). Tokyo, Japan: IEEE, pp.31-36.
7. Refaie, R., Abd El-Aziz, A., Hamza, N., Mahmood, M. and Hefny, H., 2015. A secure algorithm for executing queries over encrypted data. In: Third World Conference on Complex Systems (WCCS). Egypt: IEEE, pp.1-6
8. Foltz, K. and Simpson, W., 2018. Extending CryptDB to Operate an ERP System on Encrypted Data. In: Proceedings of the 20th International Conference on Enterprise Information Systems(ICEIS). Funchal, Madeira, Portugal: SciTePress, pp.103-110.
9. Wang, L., Yang, Z. and Song, X., 2020. SHAMC: A Secure and highly available database system in multi-cloud environment. Future Generation Computer Systems, 105, pp.873-883.
10. Tang, J., Fu, S. and Xu, M., 2019. An Effective Encrypted Scheme Over Outsourcing Data for Query on Cloud Platform. IEEE Access, 7, pp.66242-66250.
11. Rafique, A., Van Landuyt, D., Heydari Beni, E., Lagaisse, B. and Joosen, W., 2021. CryptDICE: Distributed data protection system for secure cloud data storage and computation. Information Systems, 96, p.101671.
12. Refaie, R., Abd El-Aziz, A., Hamza, N., Mahmood, M. and Hefny, H., 2014. A Survey on Executing Query on Encrypted Database. In: The International Conference on Intelligent Information Technologies (ICIIT 2014). Chennai: researchGate, pp.1-6.
13. Refaie, R., Abd El-Aziz, A., Hamza, N., Mahmood, M. and Hefny, H., 2015. A new efficient algorithm for executing queries over encrypted data. In: International Conference on Computing, Communication and Security (ICCCS). Egypt: IEEE, pp.1-4.
14. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. " Review of \CryptDB: Protecting Confidentiality with Encrypted Query Processing". May 16, 2012 SOSP, pages 85100, Cascais, Portugal, Oct. 2011.
15. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. " CryptDB: A Practical Encrypted Relational DBMS". Available:http: //lib4shared.com/doc-file/cryptdb-a-practical-encrypted-relationaldbms
16. Almakdi, S. and Panda, B., 2019. Designing a Bit-Based Model to Accelerate Query Processing Over Encrypted Databases in Cloud. [online] DeepAI. Available at: <https://deepai.org/publication/designing-a-bit-based-model-to-accelerate-query-processing-over-encrypted-databases-in-cloud> [Accessed 17 April 2021].