Information Sciences Letters
An International Journal

http://dx.doi.org/10.18576/isl/120101

Developing an Efficient Secure Query Processing Algorithm on Encrypted Databases using Data Compression

Ahmed I. Taloba, Mohamed A. Fouly and Taysir H. A. Soliman*

Information System Department, Faculty of Computers and Information, Assiut University, Assiut, Egypt

Received: 2 Mar. 2022, Revised: 25 Jun. 2022, Accepted: 28 Jun. 2022
Published online: 1 Jan. 2023

Abstract: Distributed computing includes putting aside the data utilizing outsider storage and being able to get to this information from a place at any time. Due to the advancement of distributed computing and databases, high critical data are put in databases. However, the information is saved in outsourced services like Database as a Service (DaaS), security issues are raised from both server and client-side. Also, query processing on the database by different clients through the time-consuming methods and shared resources environment may cause inefficient data processing and retrieval. Secure and efficient data regaining can be obtained with the help of an efficient data processing algorithm among different clients. This method proposes a well-organized through an Efficient Secure Query Processing Algorithm (ESQPA) for query processing efficiently by utilizing the concepts of data compression before sending the encrypted results from the server to clients. We have addressed security issues through securing the data at the server-side by an encrypted database using CryptDB. Encryption techniques have recently been proposed to present clients with confidentiality in terms of cloud storage. This method allows the queries to be processed using encrypted data without decryption. To analyze the performance of ESQPA, it is compared with the current query processing algorithm in CryptDB. Results have proven the efficiency of storage space is less and it saves up to 63% of its space.

Keywords: CryptDB, Data Compression, Distributed Database, Information Security, Cloud Computing.

1 Introduction

Digital and technological transformation in learning, medical field, and enterprises contributed greatly to the need for innovative methods of data processing like storage, analysis, and representation. Both cloud computing and the advantages of distributed database concepts is the best and optimal option in preserving and processing data from different places [1,2,3].

Since information security has become necessary and critical to maintaining the confidentiality, integrity, and availability of data. Hence, recently there has been a shift towards placing encrypted data in databases instead of storing data in plain text format [2,4,5].

To ensure confidentiality, we define the data encryption technique. The action of converting the original data through a sequence of mathematical operations to create different representations of data format is known as encryption. Encryption is viably used to conceal the information from everybody and even the approved people can see the scrambled information ciphertext. Essentially, the change of ciphertext to its unique plaintext is called decoding [6] as illustrated in Fig.1. There are two major techniques used in our algorithm encryption, and compression

Encryption: Comprises two general types of key-based encryption schemes:

1.1 Symmetric encryption

In Symmetric encryption, only one key is supported for both encrypting and decrypting the message. To encrypt the data first, give the key to the destination for decrypting the message. This type of encryption is utilized when the sender encrypts the data and, if the destination doesn't have a key to retrieve the original plain text, it sends the key and code text independently to the destination. When a key is received the information is decrypted. This

* Corresponding author e-mail: m.a.fouly@aun.edu.eg

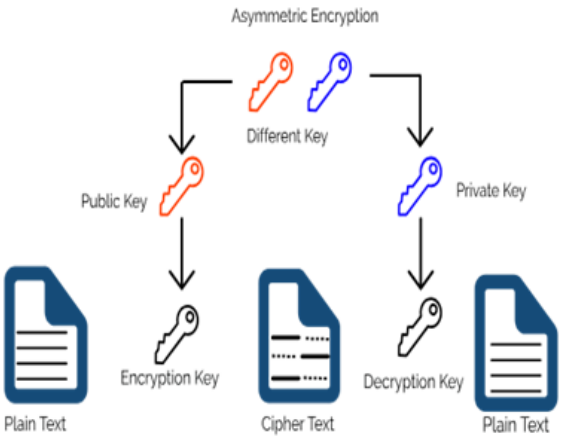

Fig. 1: Two Types of Encryptions [7]

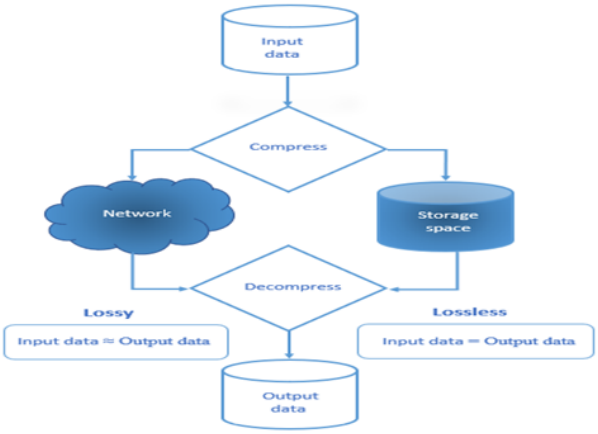

Fig. 2: Data Compression Techniques

scheme is simple and faster to actualize it yet has some disadvantages; for occurrence, when a hacker notices the key, it tends to be easier to decrypt messages. Single key encryption is commonly more straightforward for hackers. This means that the calculation used to encrypt the data is simple for the hackers to understand, and enables them to effectively translate the message.

1.2 Asymmetric encryption

This type of encryption uses two unique keys such as public and private keys. The First one is used for encryption and another one is used for decryption. The Public key can be effortlessly scattered to communicate with others since one must have the option to unscramble the message using the private key. To check the messages between the clients, the sender utilizes the public key to see the information and in the objective private key is utilized to unscramble the information. Here the private key should be secured because decryption only takes place at the destination point.

Moreover, a few data compression techniques are used to minimize the usage of bandwidth over a network and the storage space through reducing the size of the data, as shown in Fig.2 and [8]. Compression: There are two general categories for compression techniques:

1.3 Lossless Compression Technique

This technique is very efficient to transfer data over a limited bandwidth channel and also prevents the data from loss. The compression is carried out by representing the file containing a smaller number of bits, with no loss of information. This can be accomplished by different numerical and statistical tools like, entropy coding, which are also used to convert the compressed data back to the initial uncompressed data again. Traditional Lossless compression algorithms used in prior methods consist of Huffman Coding, Prediction by Partial Matching (PPM), Deflate and Abraham Lempel and Jacob Ziv (LZ77).

1.4 Lossy Compression Technique

The compression is done by minimizing the data size by eliminating pointless bits from the file. The output data obtained is not the same data before compression but it is the approximation of original data after decompression because of the removal of unwanted bits. Compression ratio is more in lossy compression techniques when contrasted to lossless compression techniques. Lossy techniques approximate the group of pixels into a solitary value which is frequently used in image and video compression schemes, by using change encoding or differential encoding methods. Joint Photographic Experts Group (JPEG) and MPEG Audio Layer-3 (MP3) are the common examples of lossy compression strategies.

We proposed an efficient secure query processing algorithm (ESQPA) on encrypted compressed databases that guarantees data confidentiality and efficiency through encryption and compression respectively. We design a strategy utilizing an ESQPA approach, which is efficient for reducing storage space by applying the LZ77 lossless compression algorithm upon the CryptDB server. Because it is a very simple technique that requires no earlier information on the source and appears to require no assumptions about the characteristics of the source. LZ77 exploits the fact that words and phrases within a text file are likely to be repeated. When there is a repetition, they can be encoded as a pointer to a previous occurrence, with the pointer followed by the number of characters to be matched [9]. We implement the ESQPA algorithm on the CryptDB system. Comparing our method with the existing server CryptDB, storage space is

less and it saves up to 63% of its space. Experimental results have proven the efficiency of the proposed method with respect to space complexity.

The paper is contributed as follows: in Section 2, we discuss the related work. Then, we explain the proposed ESQPA algorithm briefly in Section 3. In Section 4, we describe the implementation and evaluation of the algorithm. The conclusions of our work and future work are discussed in Section 5.

2 Related Work

In a few recent years, many algorithms and model systems were developed to execute query processing over the encrypted database. One of the most popular practical systems is CryptDB, which was developed by Popa [10, 11].There are three connected components in the system: database server, proxy server, and clients.

CryptDB encrypts various table columns by various encryption algorithms called onion layers encryption technique. The proxy server is placed between the clients and the main database server to encode the plain text query from the client to secure the data and the encrypted text is then sent to the database server. The server packs the information and gets the encoded response from the information base and sends it back to the intermediary server and the decoded structure is shipped off the clients, as illustrated in Fig.3.

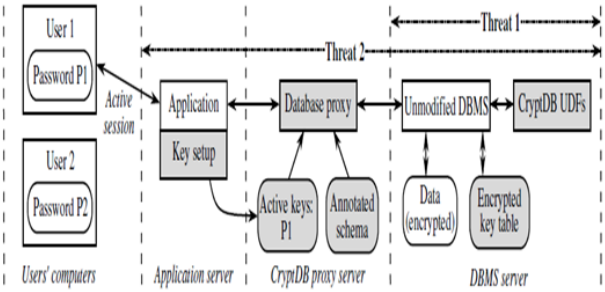

Fig. 3: CryptDB System Uses Onion Layers Encryption Techniques [10]

Ryan Su present [12] deals with secure database methods and encrypted database systems. CryptDB is an implementation that allows query processing over encrypted databases. It provides confidentiality for applications on those systems by tending two threats, DBMS Server Compromise and Adversary that gains complete control of application and DBMS servers.

Moreover, Ihsan H. Ak?n and B. Sunar also demonstrate in [13] that the data integrity is not sufficient to protect the databases, when query integrity and frequency attacks are considered. And propose a variety of practical countermeasures to mitigate attacks targeting the integrity of the CryptDB database.

The objective for Kevin Foltz and William R. Simpson [14] was to decide achievability for a full-scale usage on a real Oracle Enterprise Resource Planning (ERP) framework. This requires obliging additional highlights, for example, stored procedures, views and multi-client access controls. In addition, it shows that these extra functionalities can be basically actualized utilizing encrypted data, and they can be executed in a way that requires no code changes to the ERP application code.

Likewise, Hebah Nasereddin and Ali Darwesh in [15] also proposed object-oriented programming such as insert, update and delete objects on encrypted database system CryptDB. To insert new records, update and delete records, using Java language. This object can be called by the developer without the need to write the SQL query every time as it is built in the object.

In [16], K. Sood proposed a structure model by combining different procedures to perform the task of information security in the cloud. The model includes two phases namely the first phase manages with the process of transmitting and putting away information safely into the cloud and the second phase manages the recovery of information from the cloud and showing the generation of requests for information access.

To provide an overview of existing systems and approaches that can be used to handle encrypted data, E. Saleh, A. Alsa?deh and A. Kayed in [17] examine business utilization of such frameworks, and investigate the current advancements.

Xing bang Tian and et.al [18,19,20,21] use NoSQL databases for high scalability, availability, and high-efficiency storage and access. Security risk in these databases can be overcome by transparent middleware. In MongoDB Enterprise Advanced edition, it uses an open SSL library to encrypt pages and it improves the exhibition and it also uses two types of encryptions such as Order revealing encryption and homomorphic encryption. Also, JSON is used for the security plan. High-performance NoSQL Cassandra database is also used in health care services which provide data security during transmission. However, the performance may decrease owing to the assurance of confidentiality.

Encryption-then-Compression approach proposed by Manoj Kumar and Ankita Vaish [22] employ singular value decomposition for encrypting the images. Here loss in the compressed bit stream was overcome by Huffman coding. This technique also improves the image class and the compression exhibition is better when compared with the CTE method. To secure, storing, searching, and retrieving the encrypted data from the cloud.

Main idea in I. Demertzis, R. Talapatra and C. Papamanthou [23] is to utilize compression so as to reduce the size of the plaintext indexes before producing the encrypted searchable indices. Solution can use any existing Searchable Encryption (SE) scheme as a

black-box and any combination of lossless compression algorithms, without trading off security.

Present Yiwen Shao, Sa Wang and Yungang Bao, a backup and recovery system that could profoundly decrease the backup storage cost of encrypted databases. The key idea in [24] is to leverage the metadata information of encryption schemes and selectively backup one or more columns among semantically redundant columns.

Sultan Almakdi and Brajendra panda [25] proposed a model called BVM (bit vector-based model). This model uses QM (Query manager) for retrieving, encryption, and decryption means to decrease the quantity of recovery of encrypted data and minimizes up to 35 % of entire encrypted data.

W. Zhengy, F. Liy, Raluca Ada Popay, Ion Stoicay and R. Agarwal [26] introduced a first big data key-value store that reconciles encryption and compression. MiniCrypt mentions an experimental objective fact about data compression trends and provides a set of distributed systems techniques for recovering, updating, blending and parting encrypted packs while saving consistency and performance.

Similarly, Meng Zhang and et.al [27] also proposed a new encrypted key-value storage structure for compression which is executed on a NoSQL Cassandra database. Similar to the previous technique it also improves the system performance and furthermore builds the throughput. It upholds key-valve query and range query.

3 Proposed Work

Our algorithm (ESQPA) is applied to the CryptDB system below in Fig.4. There are three connected components in the system: database server, proxy server, and clients. CryptDB encrypts table columns by various encryption algorithms called onion layers encryption techniques. The proxy server takes the plain text query from the client which is situated between the client and the database server. Once the original text is received it encrypts the text and these encrypted queries are then sent to the information base server. Through comparing the proposed architecture with Fig.3, it is found that before storing the data into the database, the server compresses the information and decompresses it before processing a query to fetch the encrypted information from the database and send it to the proxy server. In this way the query results are decrypted and sent it to the clients. Data compression is implemented with the LZ77 Lossless compression algorithm.

The first Lempel-Ziv compression algorithm used for sequential data compression is a Lempel-Ziv 77 (LZ77) algorithm. A portion of the previously encoded sequence is described as a dictionary. The encoder scrutinizes the input sequence using a sliding window. The window comprises of two parts namely. A search buffer holds the portion of the recently encoded sequence, and a look-ahead buffer that contains the byte sequence from the coding position to the end of the input stream.

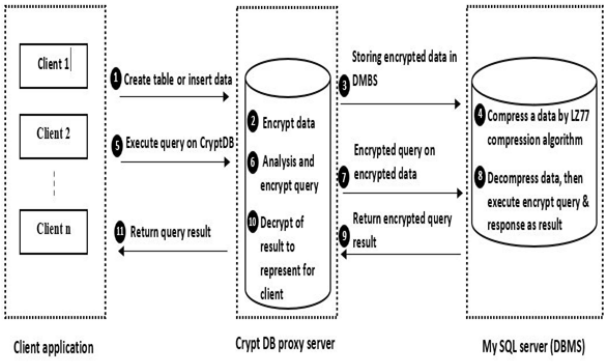

Fig. 4: Our Proposed ESQPA Algorithm as Flow Diagram

Algorithm 1: LZ77 Compression Algorithm [9]. Pseudocode of lossless compression Lempel-Ziv 77. **Input**: sequence of bytes (S = b1,b2, ..., bn). **Output**: a triple $< o, l, c >$, where 'o' meant for an offset to the match, 'l' constitutes the length of the match, and 'c' specifies the next symbol to be encoded.

1.Divide a sequence of bytes into search and look-ahead buffer.
2.While (look-Ahead Buffer is not empty):
3.Get a pointer (o, l) to longest match
4.if ($l > 0$) then Output $< o$, longest match l, $c >$, Shift the window by (l+1) positions along.
else
Output $< 0, 0$, first symbol c in the look-ahead buffer$>$, Shift the window by 1 character along.
5.Go to step 3 if look-Ahead Buffer is not empty

4 Implementation and Evaluation

In this section, the proposed ESQPA data server is compared with two similar database servers like the MySQL and the CryptDB server. For the experiments, the ESQPA was implemented by C++ programming language, on an Intel R Core (TM) i7-3770 3.4 GHz processor, and memory size of 32 GB.

ESQPA is tested on several random large-scale datasets and three real datasets, as shown in tables [1, 2].

Algorithm 2: Pseudocode of creation database and tables. Creation Database Tables DB= (T1, T2,..., Tn): database and tables creation in CryptDB server algorithm. **Input**:DB name, Table T = (Ri ,Cj). **Output**: Database and Tables are created.

Client application

1.Enter database name db_name, then write command "create database db_name if not exist"
2.Enter table name t_name and table properties columns Cj, then write command "create table t_name (Cj)"
3.Insert the data for each column Cj as rows Ri format in the CryptDB proxy server.
CryptDB Proxy server
4.Takes plain text data inserted from the client and changes it into encrypted form.
5.Forward encrypted data to the DBMS server.
DBMS server
6.Compress encrypted data by executing the LZ77 algorithm, then storing compressed encrypted data.

Algorithm 3: Pseudocode of ESQPA algorithm. Algorithm ESQPA (Q).
An Efficient and Secure Query Processing on encrypted CryptDB. **Input**:query Q as a selection command. **Output**:response from CryptDB server to client application as a query result.

Client application

1.Write a selection command query (Q) in plain text format to execute an encrypted database.
CryptDB Proxy server
2.Takes a plain text data query from the client and changes it into encrypted form.
3.Forward encrypted query to the DBMS server.
DBMS server
4.Decompresses encrypted stored data by executing the LZ77 decoding.
5.Execution of an encrypted query on encrypted data, then return the encrypted result to a proxy server.
CryptDB Proxy server
6.Similar to encryption in step 2, Decrypts the query result and sends it to the client.

4.1 Random Dataset Description

I have used the [28] tool to generate random data, as illustrated in Fig.5. It does support many field data types like integer, float, double, varchar, date, text and binary long object. In addition to supporting foreign keys constraints.

4.2 India News Headlines (INH) Dataset Description

This news dataset is real constant historical file of notable events in the Indian subcontinent from start-2001 to end-2020, recorded progressively by the columnists of India [29]. It contains roughly 3.4 million events distributed by Times of India. A majority of the data is focusing on Indian local news including national, city level and diversion. Dataset contains 3.4 million records and three columns defined as follow:

1.Publish Date: Date of the article being published online in yyyy-MM-dd format.
2.Headline_Category: Category of the headline, ASCII, dot delimited, lowercase values.
3.Headline_Text: Text of the Headline in English, only ASCII characters.

4.3 PubMed Abstracts Dataset Description

This dataset is real scraped data from the National Library of Medicine [30]. Dataset contains 13.2 thousand records and 17 columns. The columns correspond to some topics, and the records correspond to the data from the pages with articles. To be specific, abstracts where the features of the topic are indicated and the essence of the articles. There are main topics such as deep learning, covid 19, virtual reality, human connectome, brain machine interfaces, electroactive polymers, PEDOT electrodes, and neuroprosthetics.

4.4 Amazon Review Polarity Dataset Description

This dataset is a real Amazon review from 6,643,669 users on 2,441,053 products, from the Stanford Network Analysis Project (SNAP) [31]. Dataset contains about 4 million records (reviews) and three columns defined as follow:

1.Polarity: Integer value refers to sentiment opinion such that 1 for negative and 2 for positive.
2.Title: Text value refers to review heading.
3.Body: Text value refers to the review body.

Table 1: Information about random datasets

Dataset	Number of columns	Number of records
1M	8	1 million
2M	8	2 million
4M	8	4 million
8M	8	8 million
16M	8	16 million

```
mysql> describe 8M;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| tcolid | int(11)      | NO   | PRI | NULL    |       |
| tcol01 | text         | YES  |     | NULL    |       |
| tcol02 | blob         | YES  |     | NULL    |       |
| tcol03 | text         | YES  |     | NULL    |       |
| tcol04 | float(8,3)   | YES  |     | NULL    |       |
| tcol05 | varchar(255) | YES  |     | NULL    |       |
| tcol06 | text         | YES  |     | NULL    |       |
| tcol07 | date         | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
```

Fig. 5: Table structure for eight millions records

Table 2: Information about real datasets

Dataset	Number of columns	Number of records
PubMed Abstracts	17	13.2 thousand
India News Headlines	3	3.4 million
Amazon Reviews Polarity	3	4 million

Table 3: Numerical results for storage space of MySQL, CryptDB, and our algorithm ESQPA on random datasets

Dataset	MySQL server	CryptDB server	ESQPA server	Com. Ratio
1M	0.27 G	0.74 G	0.33 G	55%
2M	0.53 G	1.5 G	0.63 G	58%
4M	1.1 G	3.0 G	1.3 G	56%
8M	2.1 G	6.1 G	2.3 G	62%
16M	4.3 G	12.0 G	4.8 G	60%

Table 4: Numerical results for storage space of MySQL, CryptDB, and our algorithm ESQPA on real dataset

Dataset	MySQL server	CryptDB server	ESQPA server	Com. Ratio
PubMed Abstracts	0.21 G	0.58 G	0.24 G	59%
India News Headlines	0.78 G	2.21 G	0.92 G	58%
Amazon Reviews Polarity	1.83 G	5.86 G	2.17 G	63%

Tables [3,4] and figures [6,7] show an illustrative execution for the ESQPA algorithm. The performance of the proposed algorithm is evaluated and compared with those existing in MySQL and CryptDB servers in storage space term measured in a gigabyte.

Tables [5,6] and figures [8-10] show an illustrative run time measured in seconds for the ESQPA algorithm. The execution time of the proposed algorithm is evaluated and compared with those existing in MySQL and CryptDB servers.

As illustrated in figures[6,7], storage space at ESQPA is much less than CryptDB server storage by more than 55% saving. While the run time at ESQPA is a little higher than CryptDB as illustrated in figures [8-10]. This is due to the data compression and decompression operations that take place after encrypting the data. The sequence of operations is the key factor in a very small increase in run time. Depending on [32,33] the performance analysis of ESQPA is calculated from the product of complexity for order preserving encryption O

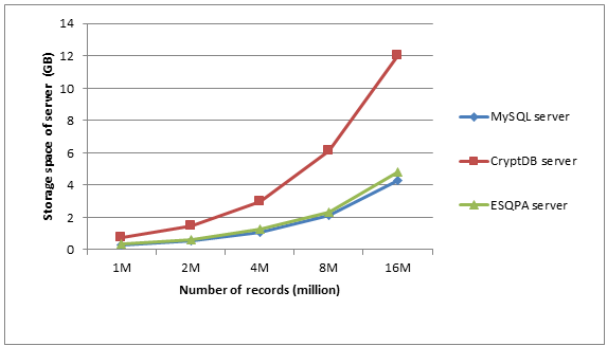

Fig. 6: Relation between storage space and number of records for these three database servers

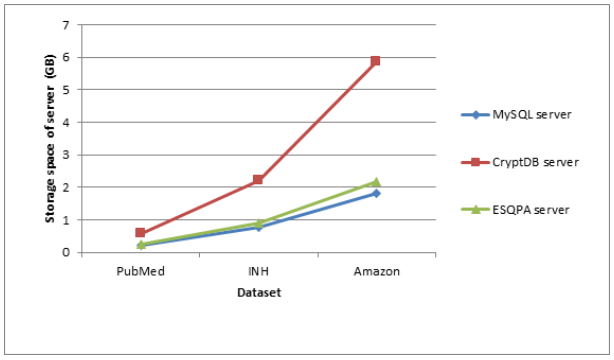

Fig. 7: Relation between storage space and real datasets for these three database servers

(n log n) by complexity for LZ77 compression O (m). Thus, time complexity for ESQPA is O (m*n log n). In general, the advantage of saving server storage space allows acceptance of a small amount of time to increase. Finally, the experimental results show that our algorithm is better and optimal for saving space. In addition to having the advantages of being intuitive, extremely less storage space specifically when applied to the encrypted database management system.

Table 5: Execution time for random datasets in MySQL, CryptDB, and ESQPA server.

Dataset	MySQL server	CryptDB server	ESQPA server
1M	167 s	1653 s	1773 s
2M	328 s	3281 s	3525 s
4M	666 s	6644 s	7156 s
8M	1352 s	12801 s	13773 s
16M	2716 s	24518 s	26446 s

Table 6: Execution time for real datasets in MySQL, CryptDB, and ESQPA server.

Dataset	MySQL server	CryptDB server	ESQPA server
PubMed Abstracts	148 s	1532 s	1728 s
India News Headlines	417 s	4225 s	4460 s
Amazon Reviews Polarity	1237 s	12682 s	13526 s

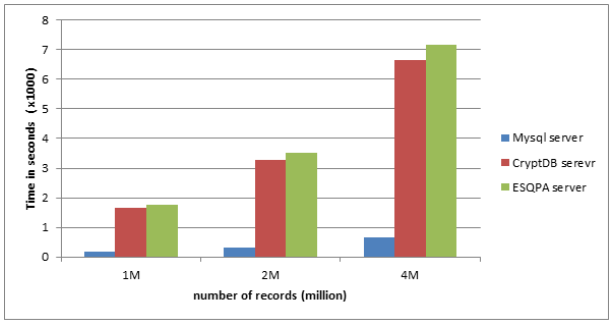

Fig. 8: Relation between time and number of records for these three database servers

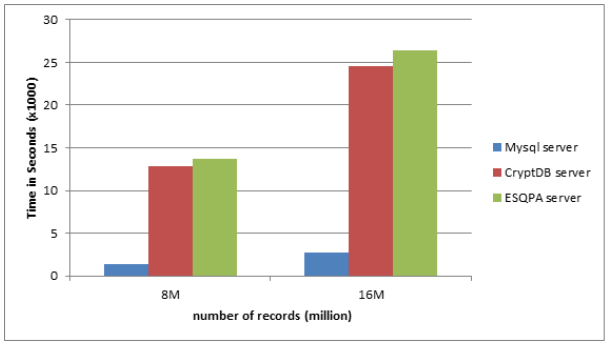

Fig. 9: Relation between time and number of records for these three database servers

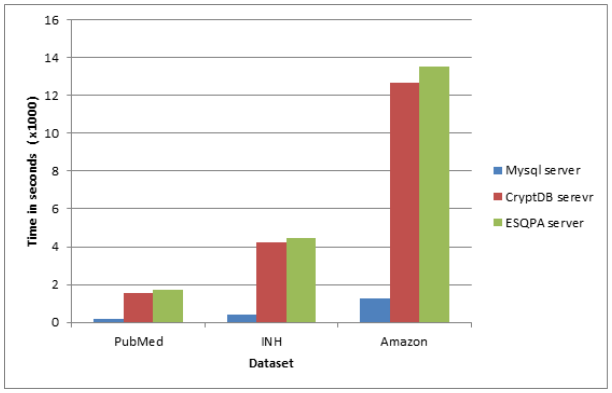

Fig. 10: Relation between time and real datasets for these three database servers

5 Conclusion

The main purpose of this paper is used to minimize space consumption and to maximize the secure and efficient retrieval of data. For this approach ESQPA algorithm is established to reduce the time and space complexity. Comparing the ESQPA method with the existing servers such as MySQL and CryptDB, storage space is less and it saves up to 63% of its space. In the case of execution time, it is a little bit higher than the prior methods because of compression and decompression. However, our proposed method performs well in saving space. For future work, we plan to improve query processing efficiency on encrypted databases by reducing the overall processing time using advanced algorithms and unconventional methods.

Conflict of Interest

All authors declare that there is no conflict regarding the publication of this paper.

Authors contributions

All authors actively equally contributed to the preparation, authorship and reviewed of the manuscript .

References

[1] Curino, Carlo et al., Relational Cloud: A Database-as-a-Service for the Cloud.,*5th Biennial Conference on Innovative Data Systems Research*,CIDR 2011, January 9-12, (2011).

[2] Amjad F. Alsirhani , Combining Multiple Encryption Algorithms and A Distributed System to Improve Database Security in Cloud Computing., , (2014) .

[3] E. S. A. Ahmed,R. A.Saeed, A Survey of Big Data Cloud Computing Security, *International Journal of Computer Science and Software Engineering (IJCSSE)*, Volume 3, Issue 1, (2014).

[4] L. Ferretti, M. Colajanni, and M. Marchetti, Supporting Security and Consistency for Cloud Database, *Proc. Fourth Int?l Symp. Cyberspace Safety and Security*, Dec, (2012).

[5] J. Vyas,P. modi, Providing Confidentiality and Integrity on Data Stored in Cloud Storage by Hash and Meta-data Approach. , *International Journal of Advance Research in Engineering, Science & Technology e-ISSN: 2393-9877*, Volume 4, Issue 5, May (2017).

[6] A.P.A.G.Deshmukh,R.Qureshi, Transparent Data Encryption- Solution for Security of Database Contents, *(IJACSA)*, Vol. 2, No.3, March (2011).

[7] http://www.stonefly.com/blog/data-encryption-essential-for-data-storage.

[8] A. Gupta,V.i Khanduja,A. Bansal , Modern Lossless Compression Techniques: Review, Comparison and Analysis, *ICECCT.2017.8117850*, (2017).

[9] S. M. Choudhary, A. S. Patel and S. J. Parmar, Study of LZ77 and LZ78 Data Compression Techniques, *International Journal of Engineering Science and Innovative Technology (IJESIT)*, Volume 4, Issue 3, May (2015).
[10] Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H, CryptDB: protecting confidentiality with encrypted query processing, *: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*,ACM, New York (2011) 85-100.
[11] A. Kumar, M. Hussain, Secure Query Processing Over Encrypted Database Through CryptDB, *Springer Nature Singapore Pte Ltd*, (2018).
[12] R. Su, Secure Database Techniques in Encrypted Database Systems, June (2018) .
[13] I. H. Ak?n and B. Sunar, On the Difficulty of Securing Web Applications using CryptDB, *IEEE International Conference on Big Data and Cloud Computing (BdCloud)*, (2014)
[14] K. Foltz and W. R. Simpson, Extending CryptDB to Operate an ERP System on Encrypted Data, *Proceedings of the 20th International Conference on Enterprise Information Systems*,pages 103-110 (ICEIS) (2018).
[15] , Hebah H. O. Nasereddin and Ali Jawdat Darwesh , An Object Oriented Programming on Encrypted Database System (CryptDB) *Talent Development & Excellence*, Vol.12, No.1, 5140 - 5146, (2020).
[16] Sandeep K.Sood, A combined approach to ensure data security in cloud computing, *Journal of Network and Computer Applications* , 35, 1831?1838, (2012).
[17] E. Saleh, A. Alsa?deh, Ch. Meinel and A. Kayed, Processing Over Encrypted Data: Between Theory and Practice., *SIGMOD Record*,(Vol. 45, No. 3) September (2016).
[18] X. Tian,B. Huang,M. Wu, A Transparent Middleware for Encrypting Data in MongoDB, *IEEE Workshop on Electronics, Computer and Applications*, (2014)
[19] M.W. Grim,A.T. Wiersma, F. Turkmen, Security and Performance Analysis of Encrypted NoSQL Databases, ,February 12, (2017).
[20] M. Ahmadian, F. Plochan, Z. Roessler, and D. C. Marinescu, SecureNoSQL: An approach for secure search of encrypted nosql databases in the public cloud, *International Journal of Information Management*, vol. 37, no. 2, pp. 63-74, (2017).
[21] S. Saha,T. Parbat,S. Neogy, Designing a Secure Data Retrieval Strategy Using NoSQL Database, *Springer International Publishing, ICDCIT*, LNCS 10109, pp. 235?238, (2017).
[22] M. Kumar, A. Vaish, An efficient encryption-then-compression technique for encrypted images using SVD, *Digital SignalProcessing*, 81?89, (2016).
[23] I. Demertzis, R. Talapatra and Ch. Papamanthou, Efficient Searchable Encryption Through Compression, *Proceedings of the VLDB Endowment*, Vol. 11, No. 11, (2018).
[24] Y. Shao, Sa Wang and Y. Bao.: CryptZip, Squeezing out the Redundancy in Homomorphically Encrypted Backup Data, *APSys ?18*, August 27?28, Jeju Island, Republic of Korea, (2018).
[25] S. Almakdi,B. Panda, Secure and Efficient Query Processing Technique for Encrypted Databases in Cloud, *2nd International Conference on Data Intelligence and Security (ICDIS)*, (2019)
[26] W. Zhengy, F. Liy, R. A. Popay, Ion Stoicay and R. Agarwal, MiniCrypt: Reconciling Encryption and Compression for Big Data Stores, . *EuroSys ?17* , April 23-26, Belgrade, Serbia, (2017).
[27] M. Zhang,S. Qi,M. Miao,F. Zhang, Enabling Compressed Encryption for Cloud Based Big Data Stores, *Springer Nature Switzerland,CANS 2019*, LNCS 11829, pp. 270?287, (2019).
[28] https://github.com/Percona-Lab/mysql_random_data_load
[29] https://www.kaggle.com/therohk/india-headlines-news-dataset
[30] https://www.kaggle.com/bonhart/pubmed-abstracts
[31] Zhang, Xiang, Junbo Zhao, and Yann LeCun, Character-level convolutional networks for text classification, *arXiv preprint arXiv*, 1509.01626 (2015).
[32] T. Bell, Better OPM/L Text Compression, *IEEE Transactions on Communications*, vol. 34, no. 12, doi: 10.1109/TCOM.1986.1096485 , December 1986, 1176-1182
[33] F. Kerschbaum, A. Schröpfer, Optimal Average-Complexity Ideal-Security Order-Preserving Encryption, *CCS?14*, November 3?7, Scottsdale, Arizona, USA, (2014).