Developing an Efficient Secure Query Processing Algorithm for Unstructured Data on Encrypted Databases

Mohamed A. Fouly
Information Systems Dept.
Assiut University
Assiut, Egypt
m.a.fouly@aun.edu.eg

Taysir Hassan A. Soliman
Information Systems Dept.
Assiut University
Assiut, Egypt
Taysirhs@aun.edu.eg

Ahmed I. Taloba
Information Systems Dept.
Assiut University
Assiut, Egypt
Taloba@aun.edu.eg

***Abstract*— A few years ago, information size increased unexpectedly and a data explosion happened. In this world of growing information, a change in database generation may also be required. Historically, we used a structured query language that works best with structured data. Now, we want to work with unstructured data as well as with structured data. The solution is to use not only SQL (NoSQL) database, this means not only structured query language. Recently, NoSQL databases are widely used in many organizations. Moreover, the data is kept in external services like Database as a Service (DaaS), where server-side and client-side security concerns are created. Additionally, the database's query processing by several clients using complicated techniques and a shared resource environment may lead to ineffective data processing and retrieval. An effective data processing technique among several customers can be used to retrieve data in a secure and effective manner. In this paper, we present an Efficient Secure Query Processing Algorithm for Unstructured Data (ESQPA_U) for efficient query processing by applying data compression techniques before transferring the encrypted results from the server to clients. We have solved security concerns by using CryptDB to encrypt a database on the server to protect the data. Encryption methods have recently been suggested to give customers secrecy in cloud storage. The queries can be processed using encrypted data using this technique without having to first decrypt it. In order to evaluate ESQPA_U performance, it is contrasted with CryptDB existing query processing method. According to results, storage space is more effective and can save up to 57% of its original space.**

Keywords*— *Cloud Computing, CryptDB, Data Compression, Information Security, MongoDB, and NoSQL Databases

I. INTRODUCTION

The increasing number of applications used in our daily lives, such as transportation, social media, government services in many developed countries, and artificial intelligence in smart cities applications. All of the above produced a huge amount of large data size, which needs modern and advanced technologies to store data. NoSQL is a current-generation database designed to provide scalability for routing big data. Non-relational databases store records in unstructured models that are different from structure tables and the database language is not SQL [1]. Because of the advanced efficiency of big data solutions in data management. Processing and storing huge data via traditional technology is very difficult. NoSQL has pushed developers over the past decade to start favoring big data databases, such as Apache, Oracle, and NoSQL databases like Cassandra and MongoDB. These databases aim to overcome the limitations of relational databases. Nowadays, large organizations need to change traditional databases to NoSQL due to unlimited scalability, excessive overall performance, data distribution, and availability. It is a high-quality task for organizations to convert existing databases to NoSQL databases [2,3].

We can define four different categories of NoSQL databases, as illustrated in Fig.1: (one) Key/value, which includes data that is saved and reachable through a completely unique key that references a value (e.g., DynamoDB, Riak, Redis, and so on.), (two) Column, just like the key/value pairs, but the key includes a combination of column, row and a hint of time used to reference groups of columns (e.g., Cassandra, BigTable, Hadoop/HBase), (three) document, includes data are saved in collections that encapsulate all of the data following a general layout like CSV, XML or JSON (e.g., MongoDB, CouchDB) and (four) graph, the graph principle is implemented and expanding among more than one computer systems (e.g., Neo4J and GraphBase) [4].

Fig.1. Four Types of NoSQL Databases [5]

In a related context, the term "NoSQL database" was originally used to describe a free, non-relational database that did not include a SQL interface in 1998. NoSQL databases have grown in popularity among corporations and organizations in recent years because of their efficiency and effectiveness in processing large volumes of dynamic, heterogeneous, and frequently unstructured data as shown in Fig.2.

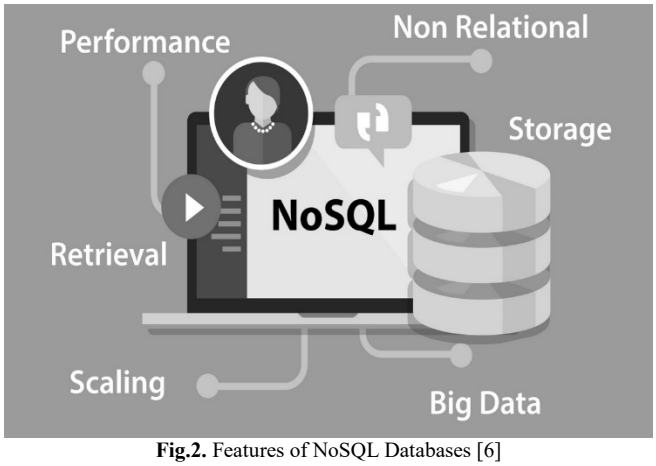

Fig.2. Features of NoSQL Databases [6]

Particularly, NoSQL databases have the following main advantages: (one) fast data management and storing operations; (two) low response time; (three) simple growth; and (four) inexpensive administration and maintenance costs. Because of the capabilities stated above, NoSQL databases better satisfy the needs and requirements of various businesses [7]. In this work, we present the ESQPA_U, an efficient secure query processing method for unstructured data on encrypted NoSQL databases that ensures data security and efficiency through the respective use of encryption and compression. Additionally, we develop an ESQPA_U technique that effectively uses the LZ77 lossless compression algorithm on the CryptDB server to reduce storage space. Because it is a fairly straightforward technique that requires no prior knowledge about the source and doesn't seem to rely on any assumptions on the source's characteristics. The LZ77 algorithm takes use of the likelihood of repetition in text files. Repetition can be represented as a pointer to a prior occurrence, with the number of characters that need to match coming after the pointer [8]. Finally, we put the ESQPA_U algorithm into operation on the CryptDB platform. When compared to the currently used server CryptDB, our technique uses up to 57% less storage space. The effectiveness of the suggested strategy in terms of space complexity has been demonstrated by experimental data. We explain the related work in Section II of the paper, which is our contribution to the effort. The proposed ESQPA_U method is then discussed in Section III. We describe the experimental results in Section IV. In Section V, conclusion and future work are discussed.

II. Related Work

A. Secure Unstructured Data (NoSQL Databases)

Many methods and model systems have recently been created to handle query processing over NoSQL databases. Tian, Xing bang and et al. [9–12] NoSQL databases are a good option if you need exceptional flexibility, reliability, high-performance storage, and retrieval. The protection risk in these databases can be decreased by employing obvious middleware. The NoSQL database's more advanced version in order to encrypt pages and enhance the display, MongoDB uses two different kinds of encryption, including order revealing encryption and homomorphic encryption. The security strategy further uses JSON. Good performance across the board The NoSQL Cassandra database is additionally utilized in healthcare systems that offer information security throughout transmission. However, the confidentiality guarantee may also result in decreased efficiency.

Zahid, Anam and et.al [13] proposed an estimation criterion comprising various protection capabilities for the evaluation of sharded NoSQL databases. Also, they present a particular view of the security capabilities provided through NoSQL databases and analyze results with an appreciation of the proposed estimation criteria.

M. Zhang and et.al [14] on the NoSQL Cassandra database, a brand-new compressed encrypted data storage format has been suggested. It boosts throughput while simultaneously enhancing system performance, much like the previous strategy. It supports range and key-valve queries.

Also, W. Zhengy, F. Liy, Raluca Ada Popay, Ion Stoicay, and R. Agarwal [15] first big data key-value store that combines compression and encryption was introduced. MiniCrypt additionally provides an experimental objective fact about data compression tendencies in addition to a collection of distributed database methodologies for retrieving, changing, combining, and splitting encrypted packs while maintaining consistency and efficiency.

B. Secure Structured Data

In one of the studies, CryptDB created by Popa, is the most well-known practical encrypted system [16, 17]. The system is made up of three interrelated components: clients, a proxy server, and a database server. With the help of multiple encryption algorithms and a method known as "onion layers", CryptDB encrypted different database columns. The proxy server sits between the users and the primary database server to protect the data. It encrypts the users' original-text queries before sending the encrypted text to the database server. The server sends the decoded structure to the intermediary server after packing the data, receiving the encoded answer from the database, and sending it back to the users, as shown in Fig.3.

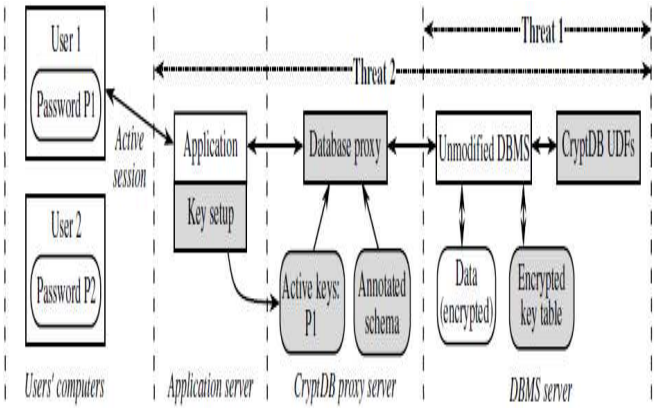

Fig.3. CryptDB System Employs Techniques for Encryption Using Onion Layers [16]

A.I. Taloba, M.A. Fouly, and T. Soliman in [18] before transferring It was recommended an Efficient Secure Query Processing Algorithm (ESQPA) for request handling effectively so that users may receive encrypted results from the server. They have solved security concerns by utilizing CryptDB to encrypt a database on the server to protect the data. Encryption methods have recently been suggested to give customers secrecy in cloud storage. This method does not require decryption of the encrypted data before processing the queries. They evaluated ESQPA's performance in comparison to CryptDB's existing query processing algorithm. According to the results, storage space efficiency is higher and can save up to 63% of the available space.

Likewise, H. Naser-Eddin and A. Darwesh in [19] It was also suggested to use object-oriented programming to create, modify, and destroy objects on the CryptDB encrypted database system. They employed the Java programming language. The SQL query is incorporated into this object, so the developer can call it without having to write it each time.

C. Encryption Then Compression (ETC) Method

The main idea of Demertzis, Talapatra, and Papamanthou [20] was to create the encrypted searchable indices first, then compress the original text indexes to make them smaller. The technique uses any current Searchable Encryption technique as a black box and any number of lossless data compression techniques without compromising security.

Finally, ETC approach was created in a study by M. Kumar and A. Vaish [21]. Utilizing the decomposition of singular values, the images are encrypted. Huffman coding was applied in this case to make up for the loss of the compressed data. Compared to the Compression-Then-Encryption (CTE) method, ETC improved the compression exhibition and the picture class.

III. PROPOSED METHOD

In this section, the proposed work ESQPA_U is applied to the CryptDB architecture, as shown in Fig.4. Clients, MongoDB as a document-based NoSQL database, CryptDB proxy server, and MySQL server are the four interconnected parts of the system. use several encryption algorithms, often known as "onion layer" encryption approaches, to encrypt table columns. The middle server receives the user's original text request and routes it between the user and the MySQL database server. After receiving and encrypting the original text, the encoded requests are subsequently sent to the MySQL database server.

A. Lempel-Ziv 77(LZ77) Data Compression Technique [22]

Lempel-Ziv 77 refers to the initial Lempel-Ziv compression technique for sequence data. The previously encoded sequence contains a portion that resembles a dictionary. The encoder uses a sliding window to thoroughly review the input sequence. The window is divided into two sections. The encoded sequence is kept in a search buffer, and the stream sequence from the coding site to the end of the input stream is recorded in a look-up table.

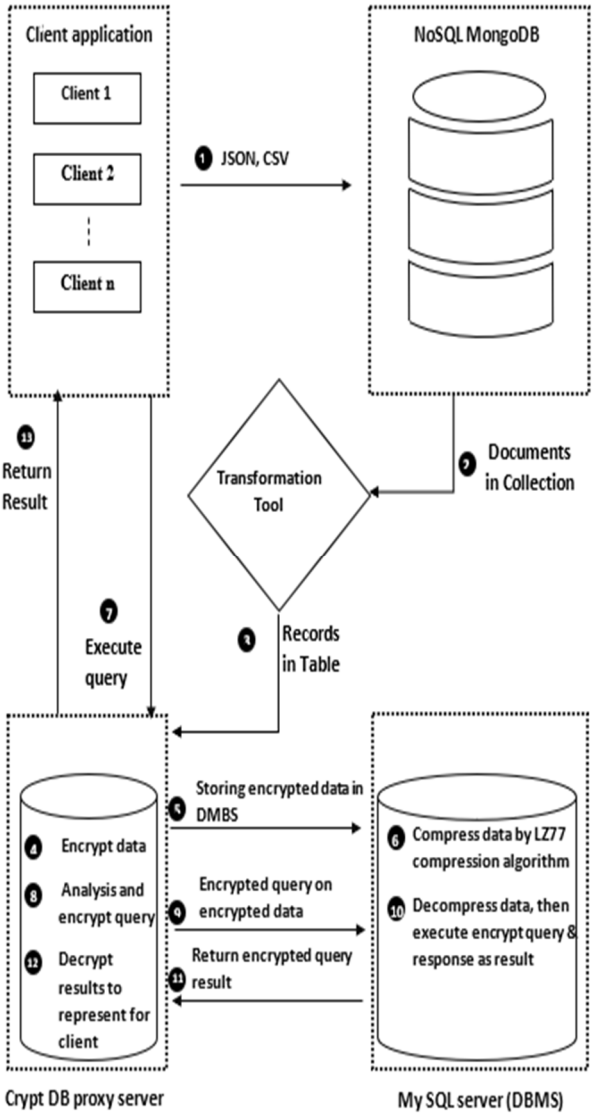

Fig.4. Proposed ESQPA_U Algorithm as Flow Diagram

LZ77 Compression Algorithm. Pseudocode of lossless compression LZ77. Input: stream of byte (S = b1b2... bn). Output: a triple with the letters "o," "l," and "c," where "o" denotes the offset to the match, "l," the length of the match, and "c," the next character to be encoded.

step1: Make a search and look-ahead buffer out of a series of bytes.
step2: While (look-Ahead Buffer is not empty)
step3: Obtain a pointer to the longest match (o, l).
step4: if (l > 0) then Output (o, longest match l, c), move the window by (l+1) positions along.
step5: else Output (0, 0, first symbol c in the look-ahead buffer), move the window by 1 character along.
step6: Go to step3 if the buffer is not empty.

B. Creation Database and Collections

Algorithm1. Pseudocode of creation database and collections. Creation_Database_Collection DB= set of collections (C_1, C_2, …, C_i): database and collection creation in MongoDB server. Input: DB name, Collection C = set of documents (D_1, D_2, …, D_j). Output: Database and Documents in Collection are Created, then convert to records in a table.

User application
step1: Enter database name *db_name*.
step2: Enter collection name *c_name*.
step3: Insert the data for each document D_j as JSON or CSV format in the NoSQL MongoDB server.

NoSQL MongoDB
step1: Store unstructured data, then convert data into documents in the collection.
step2: Export documents into the transformation tool in order to transform each document D_j in collection C_i to record in a table.

CryptDB Proxy Server
step4: Use a transformation tool to convert client-entered plain-text data into encrypted data.
step5: Send data that is encrypted to the DBMS server.

DBMS Server
step6: Apply the LZ77 algorithm to compress encrypted data, then store the reduced encoded data.

C. ESQPA_U Algorithm

Algorithm2. Pseudocode of ESQPA_U algorithm. Algorithm ESQPA_U (Q): An Efficient and Secure Query Processing for Unstructured Data on Encrypted CryptDB System. Input: query Q as a selection command. Output: as a query result, a response from the CryptDB server to the user application.

User application
step1: To execute an encrypted database, create a plain text selection command query (Q).

CryptDB Proxy Server
step2: Encrypts a data query that was submitted by the user in plain text.
step3: Transmits a secure request to the DBMS server.

DBMS Server
step4: Executes the LZ77 decoding to decompress encrypted stored data.
step5: A middle server receives the findings of an encoded query that was executed on cipher text.

CryptDB Proxy Server
step6: Delivers the client the decrypted query result after decrypting it.

IV. EXPERIMENTAL RESULTS

In this section, two similar database servers, CryptDB and MongoDB, a document-based NoSQL database are compared to the suggested ESQPA_U data server. On an Intel R Core (TM) i7-3770 3.4 GHz CPU with 32 GB of RAM, the C++ programming language is used to implement ESQPA_U for the experiments. ESQPA_U is evaluated on three real large-scale datasets in Table I.

Table I: Datasets Information

Dataset	Number of keys	Number of documents
Reddit	5	48.2 thousand
Brazilian	2	3 million
Arxiv	14	3.3 million

A. Reddit Posts – Ukraine and Economy (UE) Dataset Description [23]

Reddit posts about Ukraine and Russia war and how it affects the economy. This data is about Ukraine war news from Reddit and how the war in Ukraine affects the global economy. This dataset was collected based on some Reddit pages, such as 'Ukraine', 'world news', 'Ukrainian Conflict', 'RussiaUkraineWar2022', and 'UkraineWarVideoReport'. The dataset contains 320 documents and five primary keys, defined as follows:

1. Id: An identification number that distinguishes each object from the other.
2. Subreddit: Text showing the name of the country supported by the user.
3. Title: Text explaining the user's economic view of the Russian-Ukrainian war.
4. Selftext: Text of encouragement or criticism of war.
5. Score: Evaluation number for the order of writings.

B. Brazilian Legal Proceedings (BLP) Dataset Description [24]

Each legal proceeding in Brazil is certainly one of three feasible categories of status: (one) archived proceedings, (two) active proceedings, and (three) suspended proceedings. The three possible categories are given in a particular on spot in time, which can be temporary or permanent. Furthermore, they have decided by using the courts to arrange their workflow, which in Brazil may additionally attain lots of simultaneous instances according to the judge. Constructing machine Learning models to organize legal cases according to their status can help public and commercial entities manage large collections of legal cases, providing improvements in size and efficiency.

In this dataset, each hearing is composed of a number of concise documents called "motions" that are drafted in Portuguese by the court's management team. Although not usually, the motions deal with the proceedings and their legal standing. This data consists of two datasets: one with 6449 court cases, each with a person and a number of different motions that have been identified by attorneys, and the other

with over three million unlabeled motions. Among the classified information, 47.14 percent is archived (class 1), 45.23 percent is active (class 2), and 7.63 percent is suspended (class 3). The datasets they utilize are typical samples from the two biggest state courts, the primary (Sao Paulo) and the secondary (Rio de Janeiro). State courts manage the most variable kinds of instances throughout Brazil and are accountable for 80% of the total amount of proceedings. Consequently, these datasets are excellent representations of a totally significant portion of using language and expressions in Brazilian legal vocabulary. Regarding the labeled dataset, the key "-1" denotes the latest text while "-2" is the second latest, and so on.

C. Arxiv-Metadata-Oai-Snapshot Dataset Description [25]

For nearly 30 years, ArXiv has served the public and research groups by offering open access to scholarly articles, from the large branches of physics to the various subdisciplines of computer science to the entirety in between. Examples include math, statistics, electrical engineering, quantitative biology, and economics. This rich corpus of information offers significant, but once in a while overwhelming depth.

Effective information extraction is essential in these times of particular international problems. They provide a free, open pipeline on Kaggle to the system-readable ArXiv dataset—a collection of 1.7 million articles—that will help make the ArXiv more accessible. This dataset contains important data such as full-text PDFs, summaries, paper titles, writers, classifications, and more. They seek to empower new use instances that may enable the exploration of richer system learning strategies that incorporate multi-modal capabilities toward programs like data analysis, document recommendation systems, category estimation, co-citation connections, knowledge graph generation, and keyword extraction interfaces. A JSON-formatted description file is provided with this dataset. Each paper has a record in this file, containing:

1. Id: The paper can be accessed using an ArXiv ID.
2. Submitter: The person who submitted the essay.
3. Authors: The people who wrote the paper.
4. Title: The paper's title.
5. Comments: Extra information, such as the number of pages and figures.
6. Journal-ref: Details on the journal in which the paper was published.
7. Doi: Digital Object Identifier
8. Report-no: A unique number assigned to each reviewer report.
9. Categories: The ArXiv system's categories and tags.
10. License: A list of licenses for paper publications.
11. Abstract: The paper's abstract.
12. Versions: A version history.
13. Update_date: The date of the paper last modified.
14. Authors_parsed: Abbreviation of authors' names.

Table II and Fig.5 display an illustration of the ESQPA_U algorithm in action. The suggested algorithm's performance is assessed and contrasted with those currently used in MongoDB and CryptDB servers, whose capacity is expressed in gigabytes.

Table II: On Real Datasets, Numerical Findings for the Storage Space used by MongoDB, CryptDB, and our Algorithm ESQPA_U

Dataset	MongoDB server	CryptDB server	ESQPA_U server	Com. Ratio
Reddit	0.12 G	0.46 G	0.2 G	56%
Brazilian	1.62 G	4.3 G	2.1 G	51%
Arxiv	3.17 G	9.7 G	4.2 G	57%

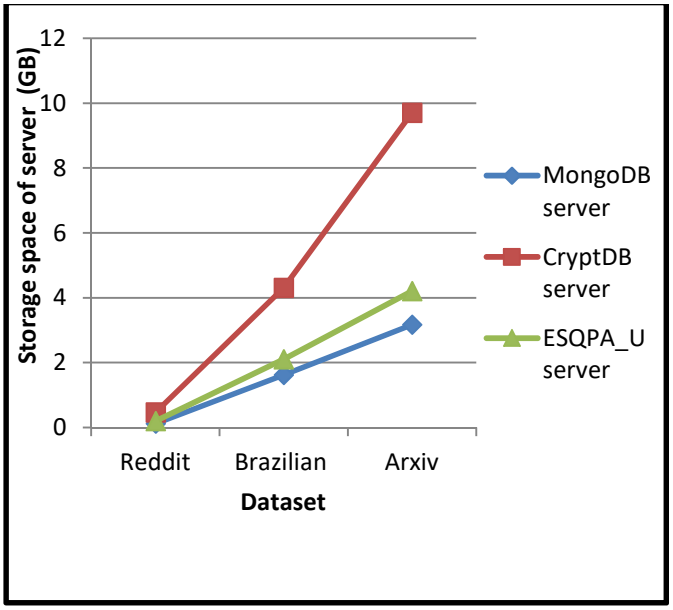

Fig.5. Relationship Between Storage and Real Datasets for Three Database Servers

Table III and Fig.6 demonstrate the ESQPA_U algorithm's illustrative execution time in seconds. A comparison is made between the suggested algorithm's execution time and that of the servers used by MongoDB and CryptDB.

Table III: Execution Time for Real Datasets in the ESQPA_U, CryptDB, and MongoDB Servers

Dataset	MongoDB server	CryptDB server	ESQPA_U server
Reddit	174 s	1465 s	1784 s
Brazilian	1035 s	18430 s	23746 s
Arxiv	2176 s	21725 s	25320 s

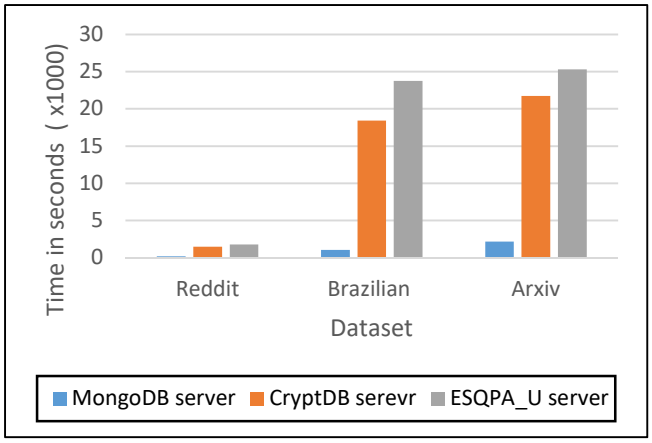

Fig.6. Relationship Between Time and Real Datasets for These Three Database Servers.

As shown in Fig.5, the capacity of ESQPA_U is 51% smaller than that of the CryptDB server. As seen in Fig. 6, Execution at ESQPA_U takes a tiny bit longer than it does at CryptDB. This is because after the data has been encrypted, data compression and decompression operations are performed. The order of the processes is mostly responsible for a relatively small rise in run time. According to [26,27], the complexity of LZ77 compression is divided by the order-preserving encryption difficulty O (n log n). The performance analysis of the ESQPA_U is built using O (m). As a result, the ESQPA_U's time complexity is O (m*n log n). However, the advantage of saving server storage space makes it possible to accept a brief increase in time. Lastly, the test results reveal that our strategy is superior and perfect for space conservation. Additionally, to being incredibly space-efficient when used with encrypted database management systems, this feature has the advantage of being intuitive.

V. CONCLUSION

The major goal of this work was to keep secure and effective data retrieval while minimizing space consumption. The ESQPA_U algorithm, which lessens space complexity, was created to demonstrate this. The ESQPA_U approach uses less storage space than existing servers like CryptDB and MongoDB, a document-based NoSQL database, and can save up to 57% of their combined capacity. Because of the time required for compression and decompression, the execution time was a little bit longer than with the previous techniques. However, our suggested approach did a good job of saving space. By reducing the total processing time with the help of cutting-edge algorithms and novel techniques, In our upcoming work, we want to improve how quickly queries are processed on encrypted databases.

REFERENCES

[1] Meier, Andreas, and Michael Kaufmann. SQL & NoSQL databases. Berlin/Heidelberg, Germany: Springer Fachmedien Wiesbaden, (2019).

[2] RAMZAN, Shabana, et al. Intelligent data engineering for migration to NoSQL based secure environments. IEEE Access, 7: pp. 69042-69057, (2019).

[3] Matallah, Houcine, Ghalem Belalem, and Karim Bouamrane, Comparative study between the MySQL relational database and the MongoDB NoSQL database, International Journal of Software Science and Computational Intelligence (IJSSCI), pp. 38-63, (2021).

[4] Maté, Alejandro, et al. Improving security in NoSQL document databases through model-driven modernization, Knowledge and Information Systems 63.8, pp. 2209-2230, (2021).

[5] https://oracle-patches.com/en/databases/nosql-case-studies

[6] https://www.complexsql.com/difference-between-sql-and-nosql

[7] Sabrina Sicari, Alessandra Rizzardi, Alberto Coen-Porisini, Security and privacy issues and challenges in NoSQL databases, Computer Networks, Volume 206,108828, ISSN 1389-1286, (2022)

[8] 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).

[9] X. Tian,B. Huang,M. Wu, A Transparent Middleware for Encrypting Data in MongoDB. IEEE Workshop on Electronics, Computer and Applications, (2014).

[10] M.W. Grim,A.T. Wiersma, F. Turkmen, Security and Performance Analysis of Encrypted NoSQL Databases. February 12, (2017).

[11] 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).

[12] S. Saha,T. Parbat,S. Neogy, Designing a Secure Data Retrieval Strategy Using NoSQL Database. Springer International Publishing, ICDCIT 2017, LNCS 10109, pp. 235–238, (2017).

[13] ZAHID, Anam; MASOOD, Rahat; SHIBLI, Muhammad Awais. Security of sharded NoSQL databases: A comparative analysis. In: 2014 conference on information assurance and cyber security (CIACS). IEEE, p. 1-8, (2014).

[14] 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).

[15] 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).

[16] Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H, CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, pp. 85–100. ACM, New York. (2011).

[17] A. Kumar, M. Hussain, Secure Query Processing Over Encrypted Database Through CryptDB. Springer Nature Singapore Pte Ltd. (2018).

[18] A.I. Taloba, M.A. Fouly, T. Soliman, Developing an Efficient Secure Query Processing Algorithm on Encrypted Databases using Data Compression, Information Sciences Letters, 12 (1) (2023), pp. 1-8.

[19] Hebah H. O. Nasereddin and Ali Jawdat Darwesh , An Object Oriented Programming on Encrypted Database System (CryptDB) . Talent Development & Excellence, Vol.12, No.1, pp. 5140 - 5146, (2020).

[20] I. Demertzis, R. Talapatra and Ch. Papamanthou, Efficient Searchable Encryption Through Compression. Proceedings of the VLDB Endowment, Vol. 11, No. 11, (2018).

[21] M. Kumar, A. Vaish, An efficient encryption-then-compression technique for encrypted images using SVD. Digital SignalProcessing60, pp. 81–89, (2016).

[22] 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).

[23] https://www.kaggle.com/datasets/dsxavier/reddit-posts-ukraine-and-economy/metadata

[24] Felipe Maia Polo, Itamar Ciochetti and Emerson Bertolo, Predicting Legal Proceedings Status: Approaches Based on Sequential Text Data. 2003.11561, (2021).

[25] https://www.kaggle.com/datasets/Cornell-University/arxiv

[26] T. Bell, Better OPM/L Text Compression. In IEEE Transactions on Communications, vol. 34, no. 12, pp. 1176-1182, December 1986, doi: 10.1109/TCOM.1986.1096485.

[27] F. Kerschbaum, A. Schröpfer, Optimal Average-Complexity Ideal-Security Order-Preserving Encryption. CCS'14, November 3–7, Scottsdale, Arizona, USA, (2014).