Implementation of novel Efficient Secure Query Processing Algorithm on Encrypted Databases using Data Compression

Surekha Pinnapati[1], Dr. Prakasha S [2]

[a] Research Scholar, Department of Computer Science Engineering, R N S I T, Bangalore, Visvesvaraya Technological University, Belagavi, Karnataka.

[b] Associate Professor, Department of Computer Science Engineering, R N S I T, Bangalore-Visvesvaraya Technological University, Belagavi, Karnataka.

Abstract

Distributed computing involves storing data using external storage and accessing it from anywhere, anytime. With the evolution of distributed computing and databases, crucial data finds its place within databases. Yet, as this data resides in outsourced services like Database as a Service (DaaS), security concerns emerge from both server and client perspectives. Furthermore, the query processing on databases by multiple clients in a shared resource environment can lead to inefficiencies in data processing and retrieval, owing to time-consuming methods. Achieving secure and efficient data retrieval is possible through an effective data processing algorithm employed by various clients. This approach suggests employing a streamlined method via Regressive Probabilistic Key Encryption (RPKE) to enhance query processing efficiency. It involves implementing data compression techniques before transmitting encrypted results from the server to clients. Our strategy for addressing security concerns involves encrypting data at the server-side using CryptDB. Recent advancements in encryption techniques aim to ensure client confidentiality in cloud storage settings. This approach enables query processing using encrypted data without requiring decryption. Evaluating the performance of RPKE involves comparing it with the existing query processing algorithm in CryptDB. The findings indicate a significant improvement in storage efficiency, with space savings of up to 61%.

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

1. Introduction

The evolution of digital and technological advancements in fields such as education, healthcare, and business has underscored the demand for innovative approaches to data processing encompassing storage, analysis, and presentation. Leveraging both cloud computing and the benefits of distributed database concepts emerges as the most effective and optimal solution for managing and processing data from diverse locations [1,2,3]. As information security now stands as a vital prerequisite for upholding data confidentiality, integrity, and availability, it has become imperative in today's landscape. As of late, there's been a noticeable trend towards housing encrypted data within databases as opposed to storing it in plain text format [2,4,5]. To uphold confidentiality, to establish a data encryption method. Encryption is defined as the process of transforming original information through a series of mathematical basic operations to generate various representations of data formats. Encryption serves as an effective means to obscure information from unauthorized access, ensuring that only authorized individuals can view the encrypted data, known as ciphertext. The process of transforming ciphertext back into its original plaintext is termed decryption [6], as depicted in Fig.1. Within the current algorithm, encryption encompasses two primary techniques, including key-based encryption novel schemes and compression encryption.

1.1 Symmetric encryption

Symmetric encryption employs a single key for both encrypting and decrypting the message. Initially, the data is encrypted, and then the key is provided to the recipient for decryption. This novel encryption method is commonly utilized when the sender encrypts the data and subsequently sends the key and ciphertext separately to the recipient, enabling them to retrieve the original plaintext. Upon receiving the key, the information can be decrypted. While this scheme is relatively simple and quick to implement, it does come with drawbacks; for instance, if a hacker intercepts the key, decrypting messages becomes easier. Single key encryption is often more susceptible to hacking attempts. This implies that the algorithm used for encrypting the information is straightforward for hackers to grasp, facilitating their ability to decipher the message effectively.

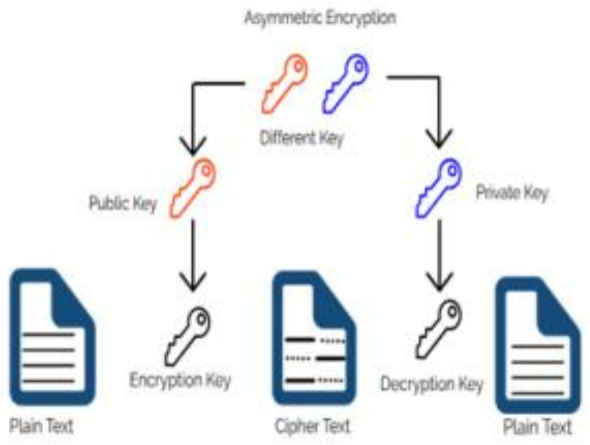


Figure 1: Two Types of Encryptions [7]

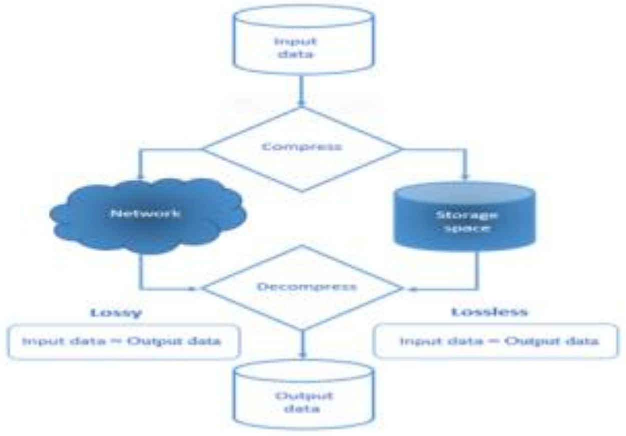


Figure. 2: Data Compression Techniques

1.2 Asymmetric encryption

In this encryption method, a pair of distinct keys is utilized: a public key and a private key. The public key is employed for encryption, while the private key is used for decryption. The public key can be readily distributed to facilitate communication with others, as recipients must possess the corresponding private key to decipher the message. To ensure secure communication between clients, the sender utilizes the recipient's public key to encrypt the data, which can then only be decrypted using the recipient's private key. In this scenario, it's crucial to secure the private key since decryption occurs exclusively at the destination point. Additionally, several data compression techniques are implemented to optimize bandwidth usage over a network and conserve storage space by reducing data size, as illustrated in Fig.2 and [8]. Compression techniques generally fall into two broad categories:

1.3 Lossless Compression Technique

This method proves highly effective in transmitting data across constrained bandwidth channels while mitigating data loss. Compression is achieved by representing the file with a reduced number of bits without sacrificing information integrity. Various numerical and statistical tools, such as entropy coding, are employed for this purpose, facilitating the conversion of compressed data back to its original uncompressed form, as depicted in Fig. 2: Data Compression Techniques. Traditional lossless compression algorithms utilized in earlier approaches include Huffman Coding, Prediction by Partial Matching (PPM), Deflate, and Abraham Lempel and Jacob Ziv (LZ77).

1.4 Lossy Compression Technique

Compression involves reducing data size by eliminating redundant bits from the file. The resulting output data obtained after compression is not identical to the original data but serves as an approximation upon decompression due to the removal of unnecessary bits. Lossy compression techniques typically offer higher compression ratios compared to lossless compression methods. In lossy techniques, groups of pixels are approximated into a single value, often utilized in image and video compression schemes, employing techniques such as transform encoding or differential encoding. Common examples of lossy compression strategies include the Joint Photographic Experts Group (JPEG) and MPEG Audio Layer-3 (MP3).

To introduced a novel method, RPKE, for secure query processing on encrypted compressed databases, ensuring both data confidentiality and efficiency through encryption and compression techniques. Our approach employs the LZ77 lossless compression algorithm within the CryptDB server, effectively reducing storage space. This technique is straightforward and does not rely on prior knowledge of the source, nor does it make assumptions about source characteristics, as it capitalizes on the repetition of words and phrases within text files. In instances of repetition, to encode them by referencing a previous occurrence using a pointer, followed by the number of characters to be matched [9]. The Regressive Probabilistic Key Encryption (RPKE) algorithm is deployed within the CryptDB system. Our approach is compared with the existing CryptDB server, demonstrating a reduction in storage space of up to 63%. Experimental results validate the efficiency of our method in terms of space complexity. This paper is structured as follows: Section 2 discusses related work, followed by a brief explanation of the proposed ESQPA algorithm in Section 3. Section 4 details the implementation and evaluation of the algorithm, while Section 5 concludes our work and outlines future directions.

2 Related Works

Over the past few years, numerous algorithms and model systems have emerged for conducting query processing on encrypted databases. Among these, CryptDB, developed by Popa [10, 11], stands out as one of the most widely adopted practical systems. The system comprises three interconnected components: a database server, a proxy server, and clients.

CryptDB employs multiple encryption algorithms, known as the onion layers encryption technique, to encrypt different table columns. Positioned between the clients and the primary database server, the proxy server encodes the plaintext query from the client to secure the data, transmitting the encrypted text to the database server. Upon receiving the encoded response from the database, the server packages the information and sends it back to the intermediary server. Finally, the decoded structure is transmitted to the clients, as depicted in Figure 3.

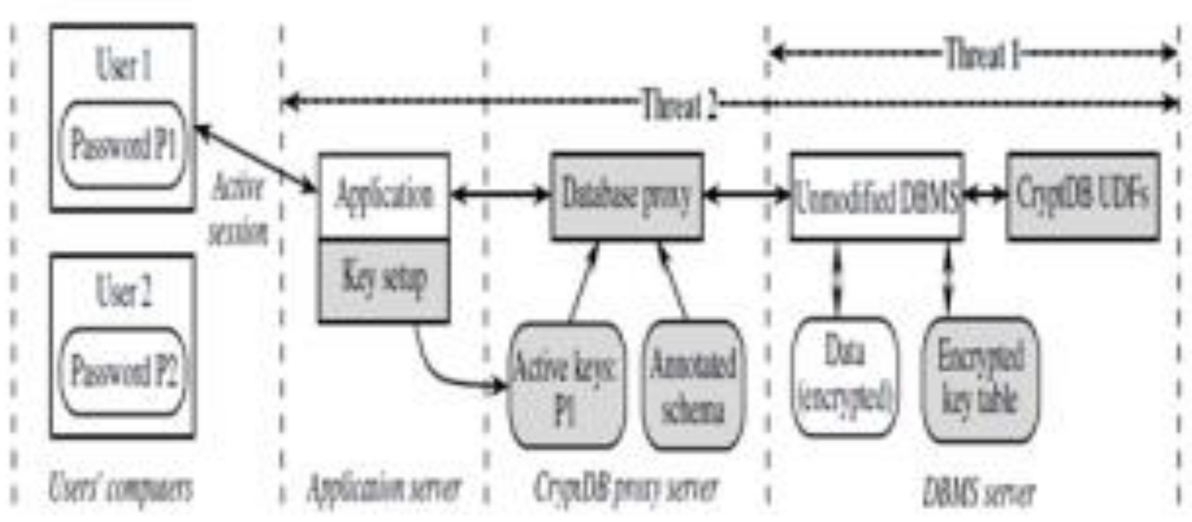


Figure. 4: CryptDB System Uses Onion Layers Encryption Techniques [10]

In his work [12], Ryan Su addresses secure database methods and encrypted database systems. CryptDB serves as an implementation enabling query processing over encrypted databases, ensuring confidentiality for applications on these systems by mitigating two primary threats: DBMS Server Compromise and adversaries gaining complete control of application and DBMS servers.

Furthermore, in their study presented in [13], Ihsan H. Ak?n and B. Sunar illustrate that data integrity alone is insufficient for safeguarding databases, particularly when considering attacks on query integrity and frequency. They propose a range of practical countermeasures to address and mitigate attacks aimed at compromising the integrity of the CryptDB database.

Kevin Foltz and William R. Simpson aimed to assess the feasibility of implementing a full-scale deployment on an actual Oracle Enterprise Resource Planning (ERP) framework [14]. This endeavor necessitated accommodating additional features such as stored procedures, views, and multi-client access controls. Their work demonstrates that these supplementary functionalities can be effectively implemented using encrypted data, and they can be integrated in a manner that necessitates no modifications to the ERP application code.

Similarly, Hebah Nasereddin and Ali Darwesh in [15] introduced object-oriented programming functionalities for operations like inserting, updating, and deleting objects within the encrypted database system CryptDB. These operations are carried out using the Java language. With this approach, developers can invoke the object directly without the need to write SQL queries each time, as the functionality is integrated into the object itself.

In [16], K. Sood proposed a structural model that combines various procedures to address information security in the cloud. This model consists of two phases: the first phase focuses on securely transmitting and storing data in the cloud, while the second phase deals with retrieving data from the cloud and generating requests for information access. Additionally, E. Saleh, A. Alsa?deh, and A. Kayed in [17] provide an overview of existing systems and approaches for handling encrypted data. They explore the business applications of such frameworks and examine current advancements in the field.

Xingbang Tian and colleagues [18, 19, 20, 21] leverage NoSQL databases for their high scalability, availability, and efficient storage and access capabilities. To mitigate security risks inherent in these databases, they employ transparent middleware solutions. For instance, MongoDB Enterprise Advanced edition utilizes an open SSL library to encrypt pages, thereby enhancing performance. Furthermore, it incorporates two types of encryption—Order revealing encryption and homomorphic encryption. JSON is utilized for the security schema. Additionally, high-performance NoSQL Cassandra database finds application in healthcare services, ensuring data security during transmission. Nonetheless, ensuring confidentiality may lead to a potential decrease in performance.

Manoj Kumar and Ankita Vaish [22] introduced an Encryption-then-Compression approach, utilizing singular value decomposition for encrypting images. They addressed the loss in the compressed bit stream by implementing Huffman coding. This technique not only enhances the image classification but also exhibits superior compression performance compared to the CTE method. It ensures secure storage, searching, and retrieval of encrypted data from the cloud.

The central concept presented by I. Demertzis, R. Talapatra, and C. Papamanthou [23] involves employing compression techniques to decrease the size of plaintext indexes before generating encrypted searchable indices. Their solution allows the use of any existing Searchable Encryption (SE) scheme as a black-box and any combination of lossless compression algorithms without compromising security.

Yiwen Shao, Sa Wang, and Yungang Bao introduce a backup and recovery system aimed at significantly reducing the storage cost of encrypted databases. The core concept outlined in [24] involves leveraging the metadata information of encryption schemes to selectively back up one or more columns among semantically redundant columns.

Sultan Almakdi and Brajendra Panda [25] introduced a model termed BVM (bit vector-based model), which employs QM (Query Manager) for retrieval, encryption, and decryption processes. This approach aims to reduce the amount of data recovery for encrypted data, resulting in a reduction of up to 35% of the overall encrypted data volume.

W. Zheng, F. Li, Raluca Ada Popa, Ion Stoica, and R. Agarwal [26] presented MiniCrypt, the initial big data key-value store integrating encryption and compression. MiniCrypt offers empirical insights into data compression trends and offers a suite of distributed systems techniques for managing encrypted packets, including recovery, updating, merging, and splitting, all while ensuring consistency and performance.

In a parallel vein, Meng Zhang and colleagues [27] introduced a novel encrypted key-value storage structure incorporating compression, implemented on a NoSQL Cassandra database. Like the previous technique, this approach enhances system performance and additionally boosts throughput. It supports both key-value queries and range queries.

3. Proposed Work

Secure query processing for diverse application development systems involves employing various techniques to retrieve relevant data from the database based on matching features and attribute ranges with encrypted data. To enhance accuracy and speed, this process can be further optimized through cloud-based data storage, parallel processing, or other techniques aimed at reducing time and space complexity. However, the selection of larger keys and the resultant encrypted data often necessitate increased storage space. In machine learning-based query processing, enhancing the classification system of ML algorithms requires augmenting the number of samples and updating other parameters. This augmentation contributes to increased space complexity, particularly as the training data for updating feature models grows. Consequently, this expansion also amplifies the time complexity involved in searching. To mitigate this issue, compression techniques are employed to store data features in the database with improved compression rates. However, it's essential to ensure that the system doesn't compromise data security by reducing the key size for encryption. The overarching goal is to achieve secure data processing while minimizing storage space. This can be accomplished by encrypting the data using lightweight cryptographic systems that allow for reduced key sizes without sacrificing data security. Additionally, compression techniques can further reduce storage size.

Within the compression framework of the secure query processing system, the cryptographic texture extraction model generates a structured grid of information across the data, enabling the identification and retrieval of matching data attributes from the database. This system incorporates the Regressive Probabilistic Key Encryption (RPKE) system, a lightweight cryptographic model that effectively reduces key size while ensuring robust data encryption. In this proposed approach, the database storage system can be efficiently managed by employing the compression model to analyze data patterns after the encryption process, facilitating storage optimization within the database. This was accomplished through the application of a lossless compression technique that combines LZ77 with the Huffman encoding model. The integrated system effectively optimizes both data size and key size, thereby reducing space and time complexities. The proposed method can be implemented using Python scripting and evaluated for performance using parameters such as Compression Ratio, Reconstruction Rate, database storage size, time complexity, security level, key size, and other relevant metrics, referencing the Ground Truth of the database.

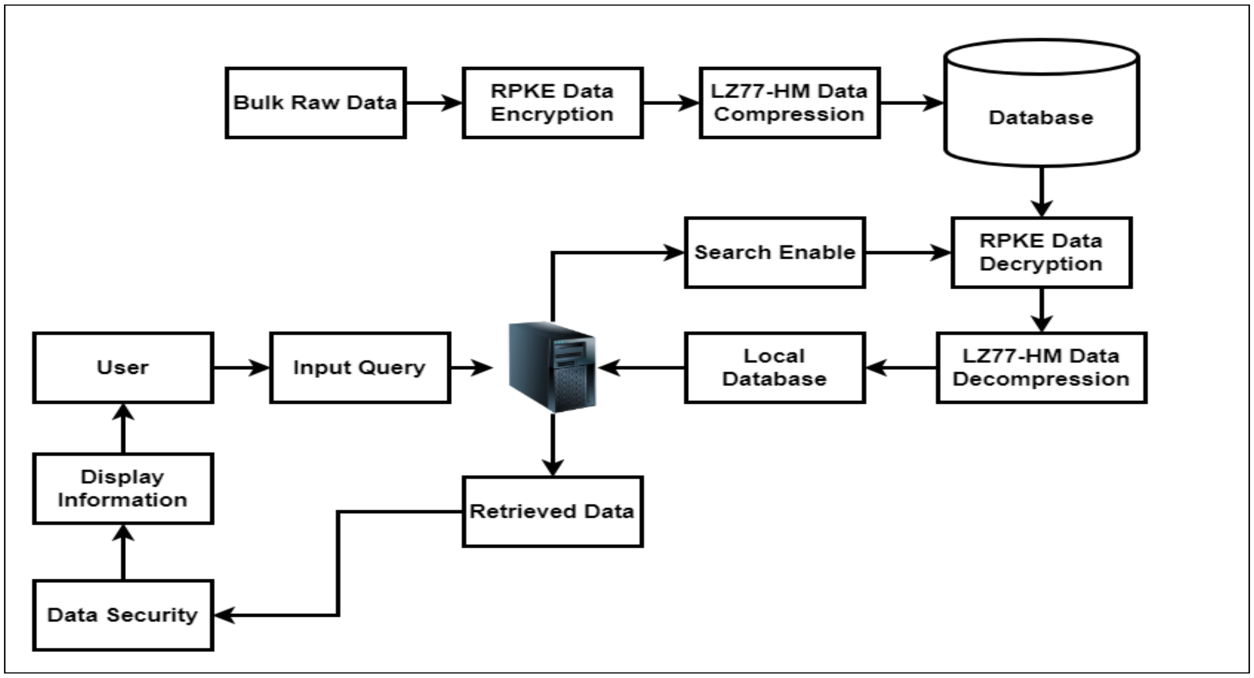


Figure 5: Flow Diagram

Compression Algorithm LZ77 with Huffman coding

```
• Start
• Input data
• for i to number of blocks
        Compress one block with one of the compression modes
        Shortest mode=min {(Mode 1 Compression), (Mode 2 Compression), (Mode 3 Compression)
        Compress with LZ77 coding
• End
• Compress data
```

4. Implementation and Evaluation

In this section, the proposed RPKE data server is compared with two analogous database servers: MySQL and CryptDB server. The experiments involved implementing RPKE using the Python programming language on an Intel Core i7-3770 3.4 GHz processor with 16 GB of memory. ESQPA was evaluated using various random large-scale datasets and three real datasets, as detailed in tables [1, 2].

Encryption Algorithm

- Start
- Input text data

```
• Generate permutation based key
• for i to number of words/character
        Convert to ASCII code
• end
• for i to number of ASCII code
        cipher text= ASCII ⊕ key
• end
• for i to number of ASCII code
        Convert to character form
• end
• encrypted text
```

4.1 Random Dataset Description

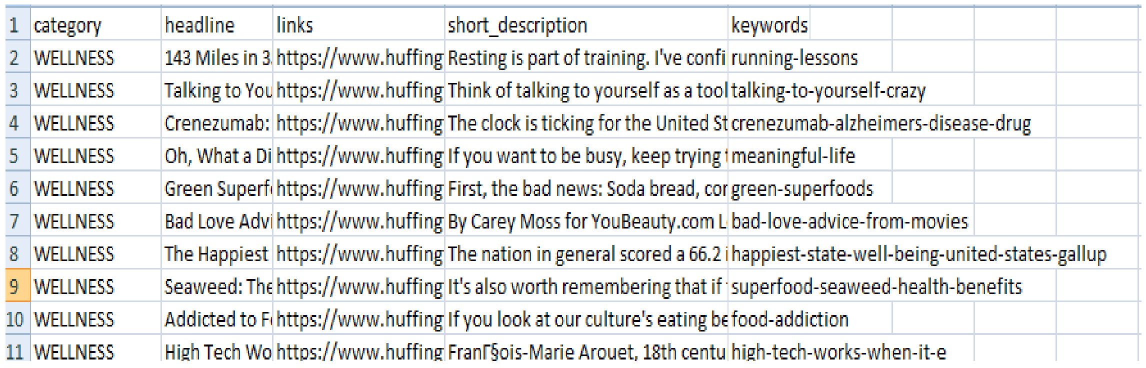
I have used the [28] tool to generate random data, as illustrated in Fig.5. It does support many field data types like integer, float, double, varchar, date, text and binary long object. In addition to supporting foreign keys constraints.

4.2 India News Headlines (INH) Dataset

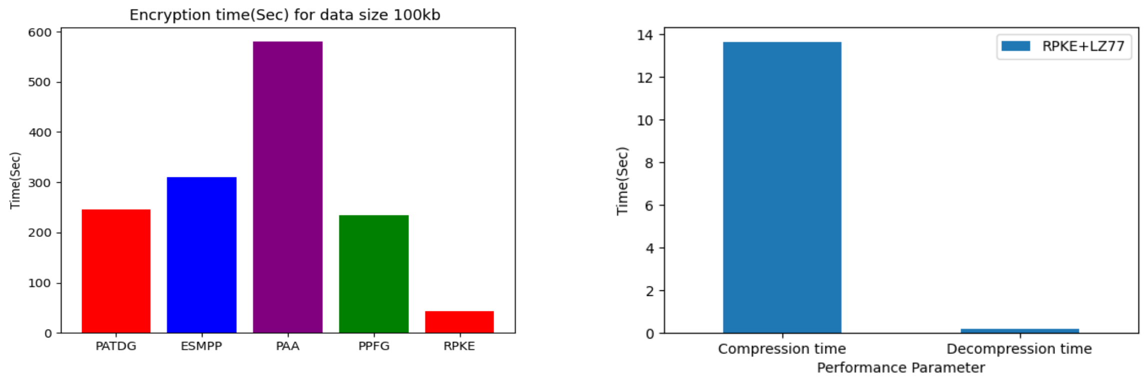
Dataset Description

This news dataset is real constant historical file of notable events in the Indian subcontinent from start-2003 to end-2021, recorded progressively by the columnists of India [29]. It contains roughly 1.2 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 1.2 million records and three columns defined as follow:

1. Article Publish Date: The date when the article is published online, formatted as yyyy-MM-dd.
2. Headline Category: The category of the headline, represented in ASCII, dot-delimited, lowercase values.
3. Headline Text: The text of the headline written in English, limited to ASCII characters



1	category	headline	links	short_description	keywords
2	WELLNESS	143 Miles in 3	https://www.huffing	Resting is part of training. I've confi	running-lessons
3	WELLNESS	Talking to You	https://www.huffing	Think of talking to yourself as a tool	talking-to-yourself-crazy
4	WELLNESS	Crenezumab:	https://www.huffing	The clock is ticking for the United St	crenezumab-alzheimers-disease-drug
5	WELLNESS	Oh, What a Di	https://www.huffing	If you want to be busy, keep trying t	meaningful-life
6	WELLNESS	Green Superf	https://www.huffing	First, the bad news: Soda bread, cor	green-superfoods
7	WELLNESS	Bad Love Advi	https://www.huffing	By Carey Moss for YouBeauty.com L	bad-love-advice-from-movies
8	WELLNESS	The Happiest	https://www.huffing	The nation in general scored a 66.2 i	happiest-state-well-being-united-states-gallup
9	WELLNESS	Seaweed: The	https://www.huffing	It's also worth remembering that if	superfood-seaweed-health-benefits
10	WELLNESS	Addicted to F	https://www.huffing	If you look at our culture's eating be	food-addiction
11	WELLNESS	High Tech Wo	https://www.huffing	FranΓ§ois-Marie Arouet, 18th centu	high-tech-works-when-it-e



.

Figure 6: Encryption and performance parameter for 100kb

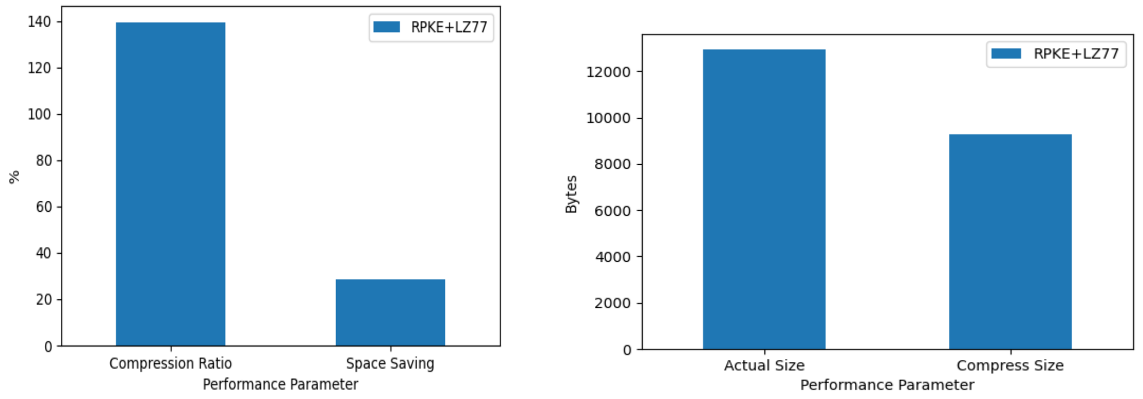


Figure 7: Encryption and performance parameter for 100kb

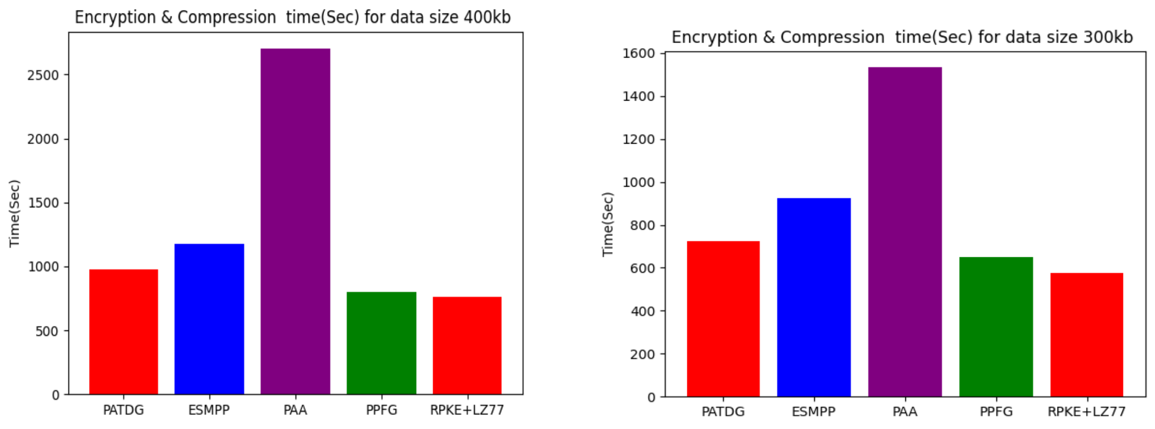
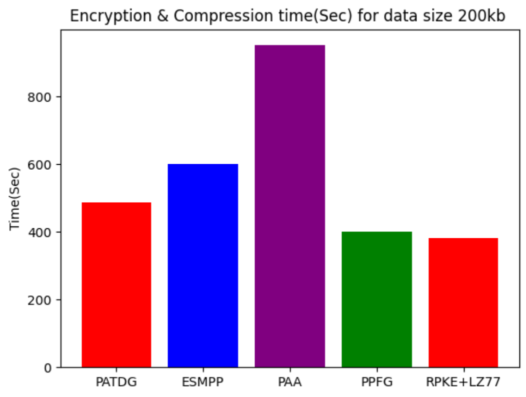


Figure 8: Encryption and Compression parameter for 400kb



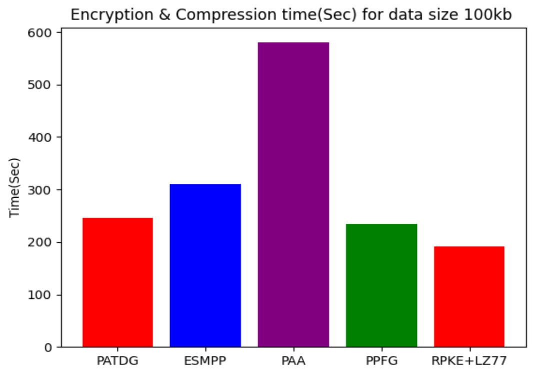
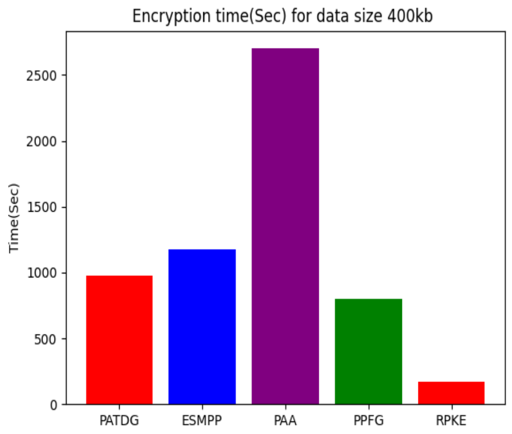


Figure 9: Encryption time and Compression time for 100kb and 200kb



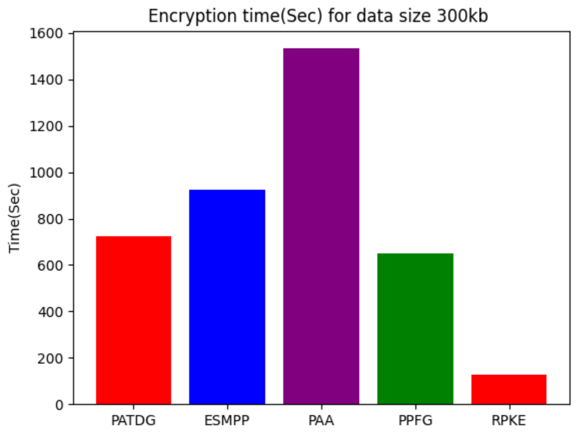


Figure 10: Encryption time and Compression time for 300kb & 400kb

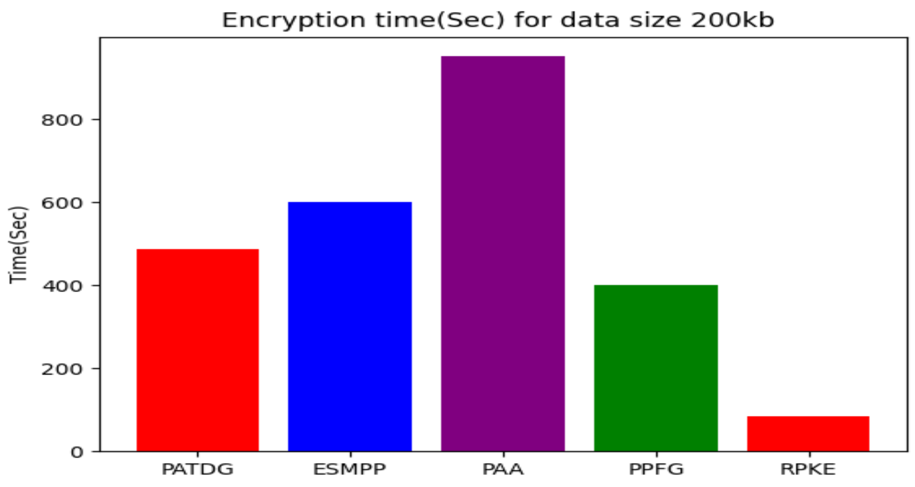
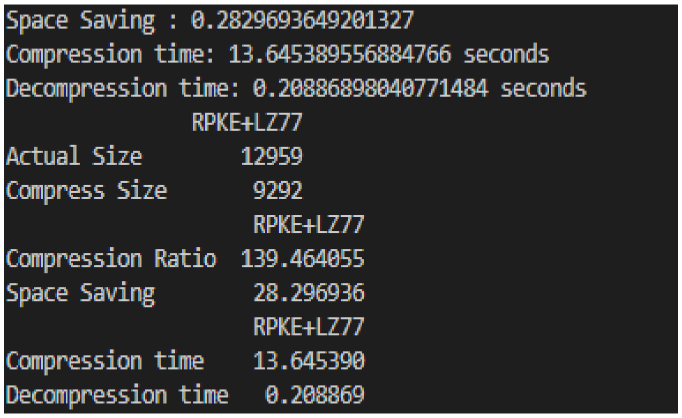
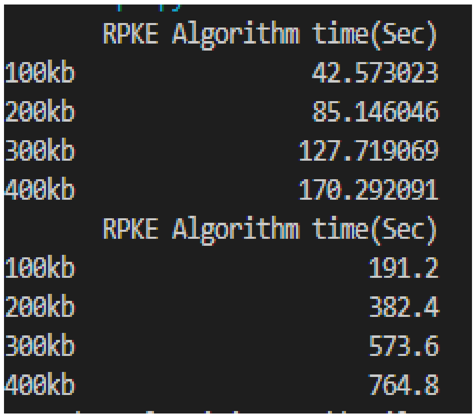


Figure 11: Encryption data size with existing algorithms 200kb

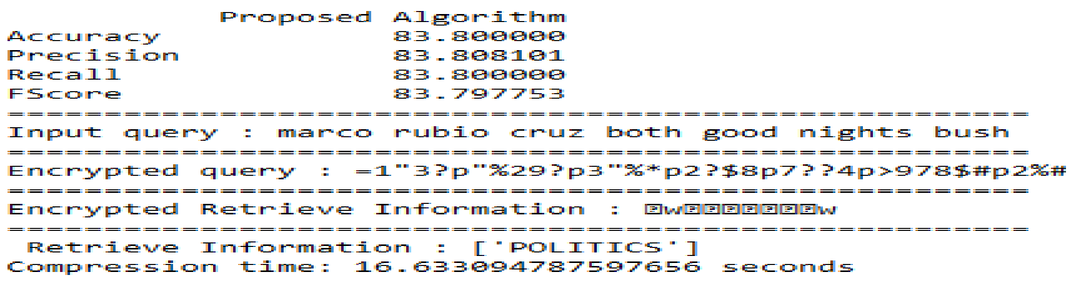


```
Space Saving : 0.2829693649201327
Compression time: 13.645389556884766 seconds
Decompression time: 0.20886898040771484 seconds
               RPKE+LZ77
Actual Size        12959
Compress Size       9292
                    RPKE+LZ77
Compression Ratio  139.464055
Space Saving        28.296936
                    RPKE+LZ77
Compression time    13.645390
Decompression time   0.208869
```

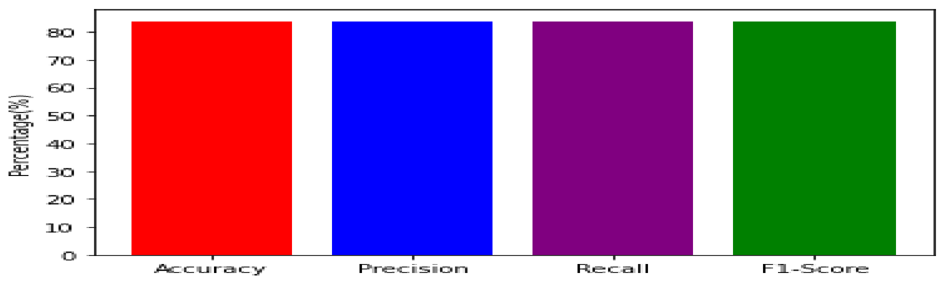


```
        RPKE Algorithm time(Sec)
100kb                 42.573023
200kb                 85.146046
300kb                127.719069
400kb                170.292091
        RPKE Algorithm time(Sec)
100kb                     191.2
200kb                     382.4
300kb                     573.6
400kb                     764.8
```

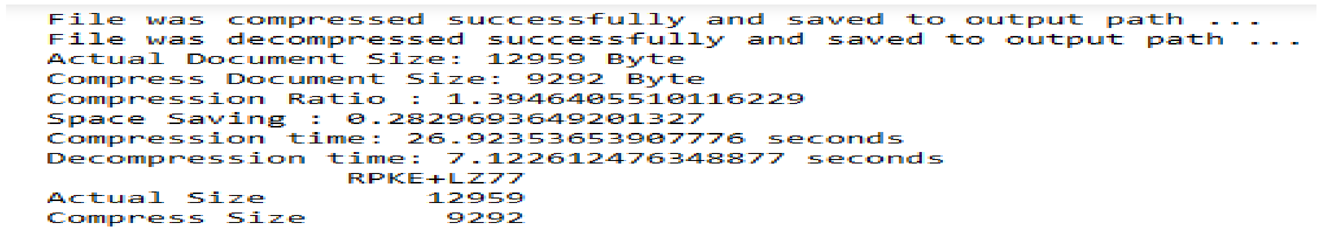
Proposed Algorithm
Accuracy 83.800000
Precision 83.808101
Recall 83.800000
FScore 83.797753
===
Input query :marco rubio cruz both good nights bush
===
Encrypted query : =1"3?p"%29?p3"%*p2?$8p7??4p>978$#p2%#8
===
Encrypted Retrieve Information :
w——
——————————————————————w
===
 Retrieve Information: ['POLITICS']
Compression time: 16.633094787597656 seconds



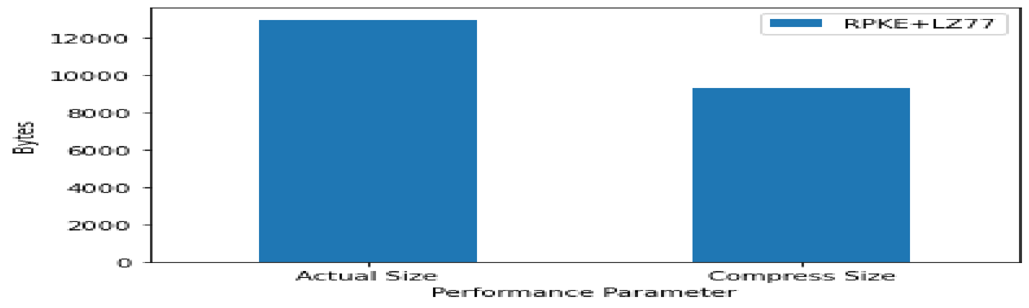
```
                 Proposed Algorithm
Accuracy                  83.800000
Precision                 83.808101
Recall                    83.800000
FScore                    83.797753
=====================================================
Input query : marco rubio cruz both good nights bush
=====================================================
Encrypted query : =1"3?p"%29?p3"%*p2?$8p7??4p>978$#p2%#
=====================================================
Encrypted Retrieve Information : □w□□□□□□□□□w
=====================================================
 Retrieve Information : ['POLITICS']
Compression time: 16.633094787597656 seconds
```

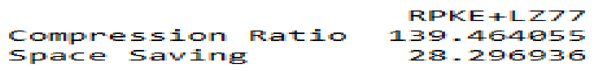


Vol. 21, No. 1, (2024)
ISSN: 1005-0930

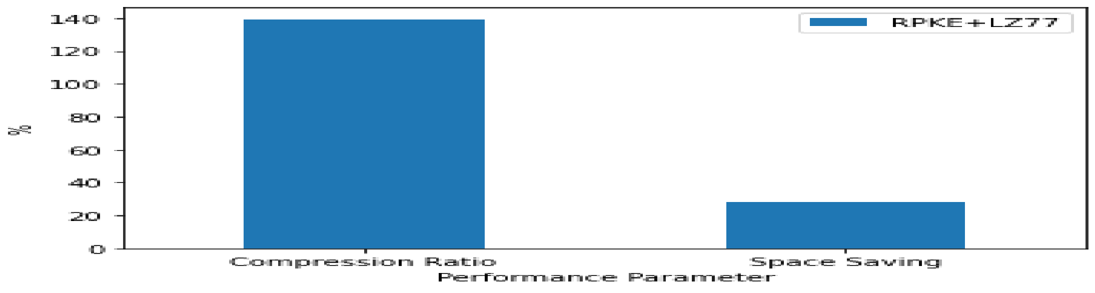


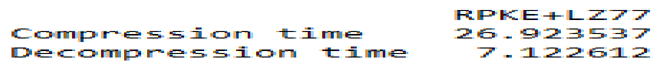
```
File was compressed successfully and saved to output path ...
File was decompressed successfully and saved to output path ...
Actual Document Size: 12959 Byte
Compress Document Size: 9292 Byte
Compression Ratio : 1.3946405510116229
Space Saving : 0.2829693649201327
Compression time: 26.92353653907776 seconds
Decompression time: 7.122612476348877 seconds
                          RPKE+LZ77
Actual Size                   12959
Compress Size                  9292
```



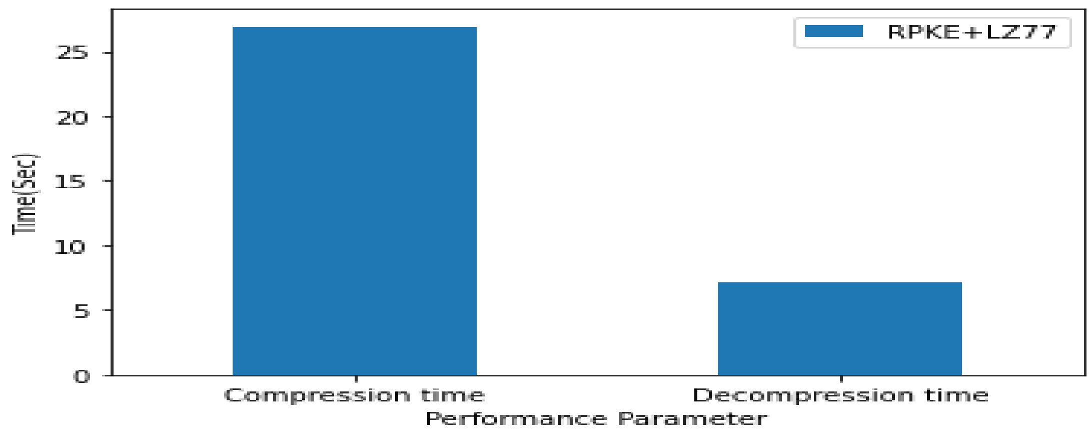


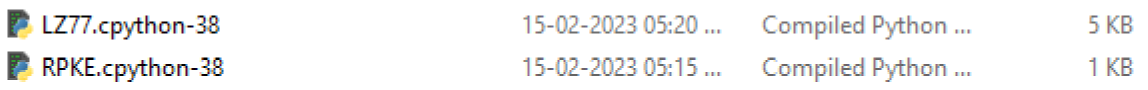
```
                          RPKE+LZ77
Compression Ratio     139.464055
Space Saving           28.296936
```





```
                        RPKE+LZ77
Compression time        26.923537
Decompression time       7.122612
```





```
LZ77.cpython-38      15-02-2023 05:20 ...   Compiled Python ...   5 KB
RPKE.cpython-38      15-02-2023 05:15 ...   Compiled Python ...   1 KB
```

5. Conclusion

The primary objective of this paper is to minimize space consumption while maximizing the secure and efficient retrieval of data. To achieve this goal, the RPKE algorithm is introduced to reduce both time and space complexity. When compared to the existing CryptDB server, the RPKE method demonstrates a reduction in storage space of up to 72%. Although the execution time is slightly higher than previous methods due to compression and decompression processes, our proposed method excels in space savings. As part of future work, to aim and enhance query processing efficiency on encrypted databases by reducing overall processing time through the utilization of advanced algorithms and unconventional methods.

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.ston efly.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). c
 2023 NSP Natural Sciences Publishing Cor.8 A. I. Taloba et al.: Developing an Efficient Secure Query Processing Algorithm...
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. Simpso n, 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. Schr¨opfer, Optimal Average-Complexity Ideal-Security Order-Preserving Encryption, CCS?14, November 3?7, Scottsdale, Arizona, USA, (2014).