MULTIDISCIPLINARY INNOVATIONS IN TECHNOLOGY AND SCIENCE JOURNAL
Vol. No. 01, Issue No. 01, April 2024, pp. 01 - 13

Optimizing Format-Preserving Encryption for Secure and Efficient Database Management

Jayashre K

Department of Computer Science and Engineering
Shiv Nadar University, Chennai, Tamil Nadu, India
jaya2004kra@gmail.com

Abstract: Format-Preserving Encryption (FPE) is a cryptographic method that converts plain text into cipher text while maintaining its original format. FPE plays a crucial role in databases by ensuring that encrypted data retains the struc-ture and attributes of the original information. CryptDB is a database manage- ment system designed to provide secure solutions for encrypted databases. By employing various encryption techniques, including FPE, CryptDB addresses se-curity concerns while preserving functionality. This study focuses on enhancing storage optimization for SQL-aware encrypted databases within CryptDB, utiliz- ing the FNR encryption scheme. The goal is to improve storage efficiency with-out compromising data integrity or format, thereby advancing secure and practi-cal database management in encrypted environments.

Keywords: SQL-Aware Encrypted Databases, FNR Encryption Scheme, FPE,CryptDB, Storage Efficiency

1 Introduction

Migrating application stacks, legacy systems, and databases to the cloud provides numerous advantages. Nonetheless, protecting the privacy of sensitive fields while re- taining their computational functionality poses a significant challenge. To maintain pri- vacy in these scenarios, it is essential to identify sensitive fields within the system and then re-engineer them to enable encryption. Despite the potential advantages, many companies hesitate to move their critical applications to the cloud due to concerns about the privacy implications of handing over sensitive data to third-party cloud providers.

For example, let's consider a data field meant for storing a 32-bit IPv4 address. Throughout the application and/or database, there are several checks in place to ensurethat the input data adheres to IPv4 standards. Protecting the privacy of this field using a typical AES-128 encryption method would require modifying its capacity to accom-modate 128 bits, departing from its original 32-bit design. Additionally, any validationsreliant on identifying the data type, such as recognizing the dotted notation of the IPv4address, would need to be removed to make way for a randomized cipher string that represents the encrypted IPv4 address.

Increasing the length of the field leads to higher demands on storage and/or band- width. However, removing validations can expose security vulnerabilities such as in- jection attacks. This concern isn't confined to IPv4 addresses; it also pertains to protect- ing the privacy of sensitive data fields like MAC

addresses, email addresses, usernames, account numbers, serial numbers, personal identifiable numbers, and credit card numbers.

Contemporary encrypted database systems with SQL awareness, as showcased by CryptDB [14], represent significant progress in tackling confidentiality challenges. CryptDB is an advanced solution crafted to systematically address privacy concerns in SQL database-dependent applications. Importantly, it enables the execution of typical SQL queries on encrypted data without requiring decryption at any stage of the opera-tion.

CryptDB utilizes traditional Symmetric Encryption algorithms like AES and Blow-Fish to safeguard data stored in the database. However, these cryptographic methods, including AES and BlowFish, do not maintain the original plaintext format after en- cryption. As a result, there is a change in both the length and type of fields in the data-base. For example, encrypting a 16-digit credit card number (originally an Integer datatype) using AES yields a 128-bit ciphertext. A significant downside of conventional encryption schemes like AES and BlowFish is their contribution to increased storage space requirements for storing encrypted data.

This article introduces the application of Format Preserving Encryption (FPE) to en-crypt data fields within CryptDB. FPE guarantees the retention of the length and for- mats of the original plaintext during encryption. Consequently, the version imple- mented is labeled as FPE-CDB. Experimental results showcased in the paper illustrateimprovements in storage efficiency, albeit accompanied by a trade-off in performancedegradation.

2 Literature Review

Boldyreva et al. [1] explore cryptographic analysis by introducing order-preserving symmetric encryption (OPE) with a new security concept (IND-OCPA), significantly enhancing efficient search capabilities on encrypted data. Bose and Bai [2] propose Homomorphic Encrypted Federated Learning, which securely integrates Big Data, Metaverse, and Large Language Models, addressing privacy and security concerns. Maryam et al. [3] implement and assess Zero Update (ZU) Encryption Adjustment for querying encrypted databases, emphasizing its secure and efficient client-side utiliza- tion. Gentry [4] presents a fully homomorphic encryption scheme, enhancing decryp- tion efficiency through ideal lattices and advancing practical applications. These studies collectively contribute to cryptographic advancements, covering OPE, federated learn- ing, encrypted querying, and fully homomorphic encryption, providing varied perspec- tives on evolving security landscapes.

Song et al. [5] introduce cryptographic schemes designed for searching encrypted data, ensuring simplicity and efficiency while maintaining provable security. In a sim-ilar vein, Kim et al. [6] propose an secure database outsourcing, leveraging homomorphic cryptosystems to bolster secu- rity and surpass existing methods in query processing speed. Rautham and Vaisla [10] tackle security and

privacy concerns within cloud database frameworks by presenting the Verifiable Reliable Secure-DataBase (VRS-DB) framework. This framework en- hances security through secret share distribution and encrypted operators operating within the same encrypted space. Furthermore, Munjal and Bhatia [11] delve into ad- vanced cryptographic techniques, comparing RSA with the Paillier algorithm to fortify data privacy in cloud services. Their work offers practical solutions for ensuring secure database operations. Collectively, these papers contribute diverse cryptographic per- spectives, presenting innovative frameworks for secure querying, outsourcing, and da- tabase operations within cloud environments.

Bellare et al. [12] introduced format-preserving encryption (FPE), a method that en-crypts plaintext into ciphertext while maintaining an identical format. They explore the"rank-then-encipher" approach using unbalanced Feistel networks (FE1 and FE2), providing security analyses and discussing applications in finance, networking, and de-terministic encryption. Popa et al. [14] presented CryptDB, which addresses the confi-dentiality of SQL database applications through SQL-aware encryption and key linkageto user passwords. This ensures efficient queries while limiting information exposure. Implementations on MySQL and Postgres demonstrate minimal overhead, making CryptDB a versatile choice for various applications. Dara and Fluhrer [15] introduced the FNR encryption scheme, a practical small-domain block cipher that prioritizes inputlength preservation, relevant for ensuring data privacy in Software-as-a-Service appli-cations. Souza and Puttini [16] addressed concerns regarding entrusting sensitive data to cloud services by leveraging client-side encryption for health and financial records. They introduced homomorphic and order-preserving encryption systems for regular searches over encrypted records, thus maintaining confidentiality and end-users' pri- vacy in cloud environments. These papers contribute to the evolution of encryption solutions, considering practical applications and trade-offs in diverse contexts such as cloud computing scenarios.

Ren et al. [18] delve into the application of fully homomorphic encryption (FHE) for unbounded aggregation queries in databases. They introduce two FHE schemes tailored for both numerical and binary values, alongside a novel mechanism for transforming ciphertext. Despite the slower processing compared to plaintext, their research show- cases the potential of FHE in OLAP queries within encrypted databases, with a strongfocus on maintaining data privacy in aggregation scenarios. On a similar note, Miao etal. [19] present an Efficient Privacy-Preserving Spatial Range Query (PSRQ) scheme designed to bolster the security of outsourcing spatial data for Location-Based Services(LBS). By integrating the Geohash algorithm with the Circular Shift and Coalesce Bloom Filter framework, they enhance the confidentiality of spatial data. The extension of their work, the PSRQ+ scheme, incorporates a Confused Bloom Filter and spatial location conversion to achieve adaptive security and improved query efficiency, partic-ularly with large datasets reaching million-levels. These contributions collectively en-rich the landscape of privacy-preserving techniques in encrypted databases, addressing

both numerical and spatial domains and offering practical solutions for secure and ef- ficient query processing.

3 Foundational Concepts

3.1 Format Preserving Encryption.

Format Preserving Encryption (FPE) ensures that the original format of input do- mains is preserved during encryption. While conventional encryption methods like AES result in a random string when encrypting an IPv4 address represented in dotted notation, FPE encryption schemes maintain the IPv4 address format in the resulting ciphertext.

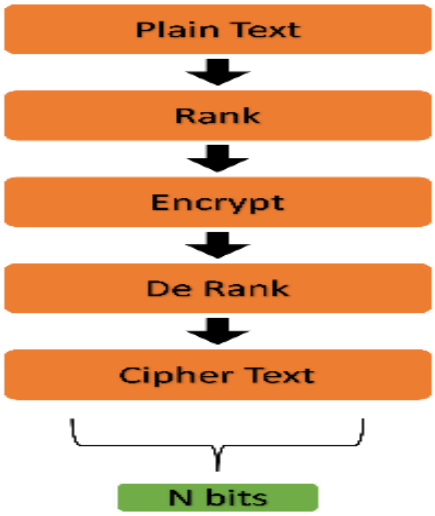

Figure 1 Format Preserving Encryption

We utilize simplified definitions of FPE schemes while maintaining their generality intact. Detailed definitions and proofs are available in [12]. FPE algorithms incorporate additional parameters known as tweak (t) and domain (d). The tweak (t) operates simi- larly to a cryptographic salt, adding extra randomness. On the other hand, the domain

specifies the context of the plaintext, such as IPv4 addresses or credit card numbers, among other possibilities [12].

The fundamental algorithm of an FPE encryption scheme are outlined as follows:

KeyGen(σ): Generates a key (k) and tweak (t) based on a security parameter (σ).

Encryption(p, k, t, d): The algorithm produces ciphertext (c) from a given plaintext (p), key (k), and tweak (t), ensuring that both the plaintext (p) and ci- phertext (c) fall within the same domain (d). This process involves utilizing rank and de-rank methods alongside a length-preserving block cipher (enc‘). Addi- tional details on ranking functions are discussed in following sections.

Decryption(c, k, t, d): For a given ciphertext (c), key (k), and tweak (t), the al- gorithm retrieves the corresponding plaintext (p). In this process, both the ci- phertext (c) and plaintext (p) belong to the same domain.

FNR, a block cipher introduced in [15], is tailored for safeguarding the format and length of sensitive data within limited domains like MAC addresses, IPV4 (32), IPV6 (32), and similar instances due to its arbitrary length and small domain. Combining Pairwise Independent Permutations with classic Feistel Networks, FNR offers en- hanced security over alternative methods.

3.2 SQL Aware Encryption

SQL-Aware Encryption refers to a specialized approach in encryption that custom- izes the encryption process to suit specific predefined SQL operations. These operations include tasks like equality checks, joins, aggregates, and inequality checks, among oth- ers. With this methodology, each data item undergoes encryption in a manner that fa- cilitates computations on the resulting ciphertext, as detailed in [14].

4 Technical Architecture

4.1 FPE-CDB

The underlying framework of our modified edition closely resembles that of CryptDB. Nevertheless, variances emerge in the specific encryption methods utilized, as discussed further in the following sections.

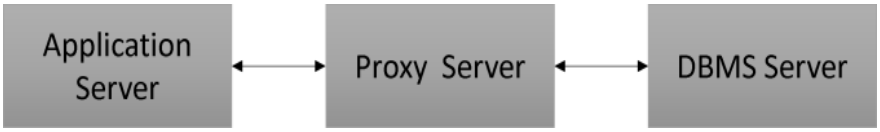

Figure 2 CryptDB Architecture

CryptDB comprises three essential components, as depicted in Figure 2: the Appli- cation Server, Proxy Server, and DBMS Server. The Application Server acts as the primary server, hosting CryptDB's database proxy and managing the DBMS Server. The Proxy Server stores crucial elements such as a confidential Master Key, the data- base Schema, and the layered encryption for all database columns. In contrast, the DBMS server holds access to the anonymized database schema, encrypted database data, and certain cryptographic User-Defined Functions (UDFs). These cryptographic UDFs are employed by the DBMS server to execute specific operations on the en- crypted data (ciphertext).

Below, we outline the steps entailed in processing a query within CryptDB.

In the first stage, the application server initializes a query. Next, the query is received and handled by the proxy server, which anonymizes both the table name and indi- vidual column names.

Additionally, all constants within the query are encrypted us- ing a securely stored master key. Moreover, the encryption layers, also known as onion layers, are adjusted depending on the specific operation required by the query. For example, if the query involves equality checks, the proxy server employs the deterministic encryption scheme (DET) to encrypt all values within the designated column where the equality check is to be conducted.

After encryption, the user's query is sent to the DBMS server. At this point, the DBMS server executes the queries using standard SQL and calls upon User-Defined Functions (UDFs) for performing particular tasks like token search and aggregation. Importantly, these tasks are conducted on the encrypted data within the database.

The operations on the encrypted data are performed by the DBMS server, after which the encrypted results are transmitted back to the proxy server.

The proxy server decrypts the encrypted query result it obtains, then returns it to the application server.

4.2 Encryption Techniques

CryptDB utilizes a variety of encryption techniques tailored to the operations speci- fied in a given query. For instance, if a query doesn't include any equality or inequality checks, it employs the random encryption scheme (RND layer). When the query in- volves SUM aggregate calculations, CryptDB utilizes homomorphic encryption (HOM layer). For executing equality checks, it applies the deterministic encryption scheme (DET layer). Notably, CryptDB employs different encryption schemes for managing inequality checks and joins.

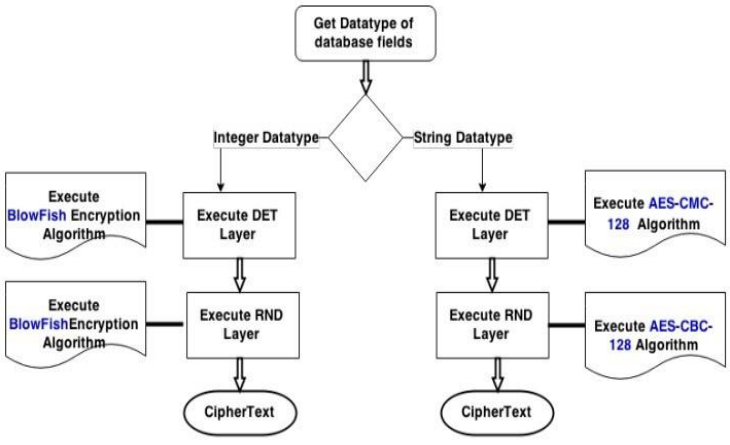

Figure 3 Onion Layers of CryptDB for an SQL INSERT query.

This paper highlights the importance of employing two encryption layers: Random (RND) and Deterministic (DET).

Random (RND)

This encryption method functions probabilistically, guaranteeing that two identi- cal plaintexts produce unique ciphertexts. This built-in probabilistic nature boosts security, making it IND-CPA secure. In CryptDB, the random encryption layer em-ploys AES-CBC-128 for strings and Blowfish (CBC mode) for integers. Both ap- proaches integrate a random initialization vector (IV) into their encryption process.

Deterministic (DET)

This encryption method functions in a deterministic manner, meaning that iden- tical plaintexts produce identical ciphertexts. This deterministic aspect makes it lesssecure when contrasted with the random (RND) layer. In CryptDB, the deterministicencryption layer is implemented using AES-CMC-128 and Blowfish, with a zero- initialized vector. This layer is primarily used by the server for executing select que-ries that involve equality checks, equality joins, COUNT operations, and DISTINCTclauses.

Figure 3 demonstrates the varied encryption layers employed by CryptDB when executing a fundamental INSERT query. Similar to Figure 2, the columns in the query undergo a two-step encryption process. Initially, they are encrypted using the DET layer, followed by encryption with the RND layer. Subsequently, the resulting cipher- text, generated by RND_int (for columns with Integer Input type) and RND_str (for columns with varchar input type), is stored within the database.

FPE-CDB effectively achieves the two characteristics specified for Format Preserv-ing Encryption (FPE) schemes mentioned earlier. Figure 4 illustrates the various layersof security utilized by FPE-CDB.

In addition to the initial two layers described earlier, CryptDB [1] integrates severalsupplementary layers akin to onion layers. These encompass the Order-Preserving En-cryption (OPE) layer, devised to streamline range queries; the Homomorphic Encryp- tion (HOM) layer [4] [13], aimed at enabling aggregation and counts; and the SEARCHlayer, facilitating full word keyword searches on encrypted data [5].

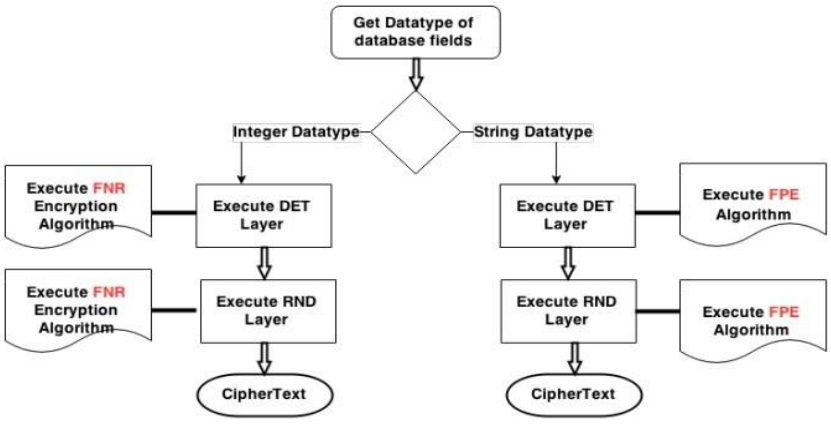

Figure 4. Onion Layers of FPE-CDB for an SQL INSERT Query

The presence of these extra layers facilitates the performance of diverse SQL oper- ations on encrypted data without necessitating decryption. These layers remain intact within the adapted version, FPE-CDB.

4.3 Toy Application

Our study focuses on a network monitoring application that serves as the benchmark. This application utilizes a dataset commonly used by security analysts to monitor and analyze network traffic. The dataset includes crucial attributes such as Source IP ad- dress, Destination IP address, Protocol, Number of packets, Number of Records, No of Bytes, Start date, End date, and Sensor. These fields are essential for various network monitoring tasks like traffic analysis, intrusion detection, and packet filtering. For our analysis, we have chosen five specific fields from this dataset, which are outlined in Table 1.

Table 1 Database Schema of FPE-CDB

Column ID	Field	Plaintext (char)	AES (char)	FNR (char)
1	Source IP	15	48	15
2	Destination IP	15	48	15
3	Start-Date	19	64	19
4	End-Date	19	64	19
5	Sensors	2	48	2

4.4 Encryption of Datatypes

Preserving the original formats and lengths of input strings is paramount in the mod- ified FPE-CDB. To achieve this, the Flexible Naor and Reingold (FNR) encryption scheme has been chosen. FNR is acknowledged for its capability as a length-preserving block cipher, accommodating input sizes from 32 to 128 bits [15], [7]. This choice guarantees that both the lengths and formats remain intact throughout the encryption process, ensuring the preservation of data types such as IPv4 (referenced in Table 2) and Time Stamps (referenced in Table 3), as further discussed below.

Table 2 Samples for IPv4 Addresses

Plain Text		Cipher Text	
Raw (Dotted)	Ranked (In- teger)	Raw (Inte-ger)	De-ranked(Doted)
64.243.129.86	941079480	1226870871	73.32.144.87
56.23.187.184	2213763856	1067498731	63.160.188.235

131.243.91.16	4026531837	2739475379	163.73.19.179
239.255.255.253	905584639	2223369266	132.133.236.50
53.250.31.255	3222780570	2000079960	119.54.204.88

IPv4 Address

Ranking an IPv4 address involves interpreting it as a 32-bit integer. This integer is then encrypted using a block cipher like FNR, yielding another 32-bit integer. Tomaintain the original format, the resulting ciphertext is converted back (de-ranked) to the dotted notation of the IPv4 address. The algorithm's process is depicted in Figure 1.

Time Stamp

The process starts by converting the input timestamp into an epoch value, whichrepresents a specific date and time used as a reference for computer clocks and timestamps. This epoch value is then encrypted using the FNR encryption scheme. After encryption, the encrypted epoch value is reversed to obtain a date string, main-taining the original format of the plaintext. This resulting date string serves as the ciphertext.

Table 3 Samples for Timestamps

Plain Text		**Cipher Text**	
Raw (String)	**Ranked(Integer)**	**Raw (Inte-ger)**	**De-Ranked(String)**
Date String	*Unix Timestamp*	*Unixtimestamp*	*Date String*
16/11/2023T21:04:05	1100639045	1531152620	09/07/2021T16:10:20
15/11/2023T15:44:38	1100533478	627185476	16/11/2018T02:11:16
14/12/2022T09:27:05	569150825	3645016902	03/07/2085T16:41:42
7/10/2023T41:33:07	1105205587	2685279782	03/02/2085T15:03:02
6/11/2023T22:07:06	1105049226	3275728681	20/10/2073T12:38:01

Integers

In the application, there are certain fields, like Port Number and Packets Trans- ferred, which are of integer type. When encrypting these integers using the FNR scheme, the result is an integer ciphertext. These particular fields do not necessitatedistinct ranking functions. In this implementation, we utilize an unsigned 64-bit in- teger data type to store the ciphertexts.

5 Experiments

The investigation utilized a setup comprising Ubuntu 23.10 on VMware, equipped with 7GB of RAM and powered by a 1.70GHz 12th Gen Intel(R) Core (TM) i5-1240P Pro- cessor. The reference data

originated from anonymized enterprise packet header traces obtained from Lawrence Berkeley National Laboratory and ICSI [8][9]. This dataset encompasses network traffic from late 2004 to early 2005, presented in SILK flow rec- ord format, and was restricted to specific hours and dates. The encryption process in- volved encrypting a maximum of 1 million SILK flow records [17] using the FNR en- cryption scheme, followed by a subsequent performance analysis.

6 Results

Storage: The results of our experiments show a roughly 50% increase in storage efficiency with FPE-CDB. In Figure 5, we visually compare the estimated storage space needed for data encrypted in CryptDB (using AES) versus FPE-CDB (using FNR). The degree of storage improvement depends on the specific application data.

For our dataset, we observed an approximate 50% enhancement in storage effi- ciency. The X-axis of the graph represents the number of records inserted into the database, while the Y-axis shows the Database Size in MB. Additionally, Table 1 provides details on the database schema of FPE-CDB, including the length (numberof characters) of AES ciphertexts (used in CryptDB) and FNR ciphertexts (used in FPE-CDB).

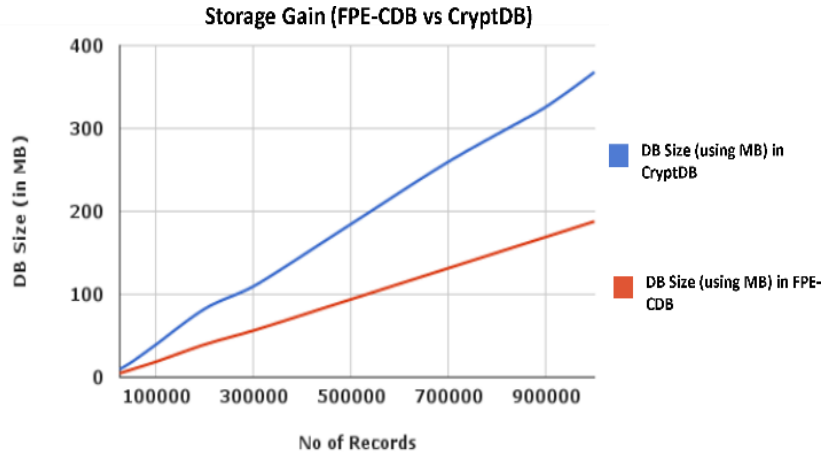

Figure 5 Storage for CryptDB versus FPE-CDB (in MB)

Calculating the average row length in SQL databases involves considering various parameters such as the fixed data size (size of fixed-length SQL fields), variable data size (size of variable-length SQL datatypes like VARCHAR, number, etc.), null bit- map size, and row header size. Consequently, the total database size can be expressed as data length = Total Number of Rows × Average Row Length. In this particular setup, the database comprises five VARCHAR fields as illustrated in Table 1. Addi- tionally, the encrypted database stores salt values used for encrypting queries and onion layers in

CryptDB. Thus, the estimated average row length includes the sum of the variable data length (size of stored ciphertext) and the total size required to store the salt value for each field. Based on this calculation, the average row length of the encrypted table in FPE-CDB is approximately 197 bytes, considering some overhead.

Performance: Figure 6 depicts a performance comparison between FNR and AES-128. The horizontal axis indicates the number of records inserted, while the vertical axis represents the time (measured in milliseconds) needed to execute a specific number of SQL INSERT queries. In the case of FPE-CDB, there's a decline in per- formance compared to CryptDB. Notably, the observed decrease in performance is approximately sevenfold ($y \approx 7x$). This is consistent with the anticipated theoretical outcome, as the FNR scheme inherently involves seven rounds of AES as part of its design [15].

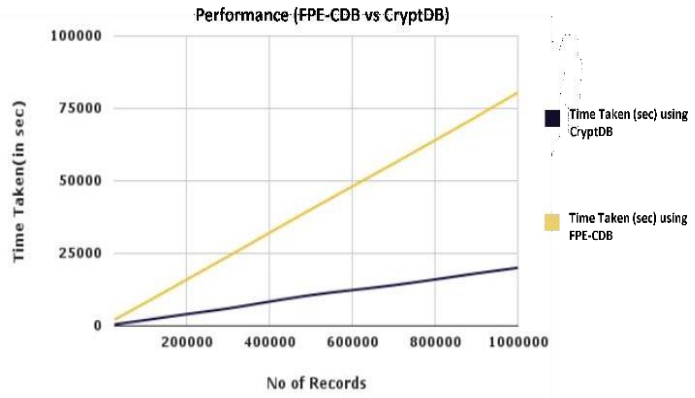

Figure 6 Performance of DPE-CDB versus CryptDB

7 Conclusion

In summary, this paper introduces FPE-CDB, an Encrypted Database that preservesformat, with a focus on network monitoring as its primary application. Through exper-iments, the study demonstrates the storage benefits achievable through Format Preserv-ing Encryption (FPE) methods, albeit with a performance trade-off. Moving forward, enhancing security could involve integrating authentication and data integrity featuresinto both CryptDB and FPE-CDB.

8 References

[1] A Boldyreva, N Chenette, Y Lee and A O'Neill, "Order-preserving symmetric encryption", *Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2009, vol. 5479, doi: https://doi.org/10.1007/978-3-642-01001-9_13

[2] A Bose and L Bai. “A Fully Decentralized Homomorphic Federated Learning Framework”, IEEE *20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, 2023, pp. 178-185, doi: https://doi.org/10.1109/MASS58611.2023.00029

[3] A Maryam & K Boris & L Alexei, “Zero Update Encryption Adjustment on Encrypted Da-tabase Queries”, 2023, doi: http://dx.doi.org/10.1007/978-3-031-37807-2_2

[4] C Gentry, “Fully Homomorphic Encryption using Ideal Lattices”, Proceedings *of the 41st Annual ACM Symposium on Theory of Computing, STOC,* 2009, vol. 9, pp. 169-178, doi: http://dx.doi.org/10.1145/1536414.1536440

[5]Rautham & K S Vaisla, “V RS-DB: preserve confidentiality of users’ data using encryp- tion approach”, *Digital Communications and Networks*, 2023, vol. 7, pp. 62-71, doi: https://doi.org/10.1016/j.dcan.2019.08.001

[6]K Munjal & R Bhatia, “Analysing RSA and PAILLIER homormorphic Property for security in Cloud”, *Procedia Computer Science,* 2022, vol. 215, pp. 240-246, doi: https://doi.org/10.1016/j.procs.2022.12.027

[7] D X Song, D Wagner and A Perrig, “Practical techniques for searches on encrypted data”,

[8] *Security and Privacy,* 2000, pp. 44-55, doi: http://dx.doi.org/10.1109/SECPRI.2000.848445

[9] H Kim, Y Kim, H Lee & J Chang, “Privacy – Preserving Top-k Query Processing Algo- rithms Using Efficient Secure Protocols over Encrypted Database in Cloud Computing En- vironment”, *Electronics*, 2022, vol. 11, pp. 2870, doi: http://dx.doi.org/10.3390/electron- ics11182870J S

[10] M Bellare, T Ristenpart, P Rogawat & T Stegers, “Format-preserving Encryption”, *SelectedAreas in Cryptography*, 2009, vol. 5867 pp. 295-312, doi: https://doi.org/10.1007/978-3- 642-05445-7_19

[11] P Pailler, Proceedings of the 18th Annual International Conference on the Theory and Ap- plications of Cryptographic Techniques, 1999, doi: https://doi.org/10.1007/b72231

[12] R A Popa, C Redfield, N Zeldovich and H Balakrishnan, “CryptDB: protecting confidenti- ality with encrypted query processing”, *Proceedings of the Twenty-Third ACM Symposiumon Operating Systems,* 2011, pp. 85-100, doi: https://doi.org/10.1145/2043556.2043566

[13] Dara and S Fluhrer, “FNR: Arbitrary length small domain block cipher proposal”, *Secu- rity, Privacy and Applied Cryptography Engineering*, 2014, vol. 8804, doi: https://doi.org/10.1007/978-3-319-12060-7_10

[14] Stefano M P C Souza & Ricardo S Puttini, “Client-side Encryption for Privacy-sensitive Applications on the Cloud”, *Procedia Computer Science*, 2016, vol. 97, pp. 126-130, doi: https://doi.org/10.1016/j.procs.2016.08.289

[15] T Shimeall, S Faber, M DeShon & A Kompanek, “Using SILK for network traffic analysis”*CERT Network Situational Awareness Group,* 2010.

[16] X Ren, L Su, Z Gu, S Wang, F Li, Y Xie, S Bian, C Li and F Zhang, “HEDA: Multi-Attribute Unbounded Aggregation over Homomorphically Encrypted Database”, *Proceedings of the VLDB Environment*, 2023, vol. 16, issue 4, pp. 601 – 614, doi: http://dx.doi.org/10.14778/3574245.3574248

[17] Y Miao, Y Yang, X Li, Z Liu, H Li, K Choo and R Deng, “Efficient Privacy-Preserving Spatial Range Query Over Outsourced Encrypted Data”, *IEEE Transactions on Information Forensics and Security*, 2023, vol. 18, pp. 99, doi: http://dx.doi.org/10.1109/TIFS.2023.3288453