Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 8s (2025)
https://www.jneonatalsurg.com

Implementation and Evaluation of Data Protection in Databases Using Symmetric Encryption Algorithms

Venkatesh B[1], L Swathi[2], Naresh Tangudu[3], Satishkumar V E[4]

[1]Associate Professor, Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Hyderabad, Telangana, India, 500090.

[2]Department of Information Technology, GMR Institute of Technology, Rajam, Andhra Pradesh,532127.

[3]Department of Computer Science Engineering (Data Science), Aditya Institute of Technology and Management, Tekkali, Srikakulam, Andhra Pradesh 532201.

[4]School of Engineering and Technology, Sunway University, No. 5, Jalan University, Bandar Sunway, 47500, Selangor Darul Ehsan, Malaysia.

***Corresponding Email:**

Email ID: sathishv@sunway.edu.my

.Cite this paper as: Venkatesh B, L Swathi, Naresh Tangudu, Satishkumar V E, (2025) Implementation and Evaluation of Data Protection in Databases Using Symmetric Encryption Algorithms. *Journal of Neonatal Surgery*, 14 (8s), 440-459.

ABSTRACT

In an era marked by rising cyber threats, ensuring database security and protecting sensitive information have become critical challenges. This study presents an experimental framework to evaluate the performance of various symmetric encryption algorithms—AES, DES, Blowfish, RC4, and ChaCha20—focusing on encryption and decryption efficiency and their impact on Data Manipulation Language (DML) operations, including insertion, selection, and update processes. A payment transaction system was simulated to replicate real-world conditions for assessing DML performance. Performance metrics such as encryption and decryption time were measured under different scenarios involving various database types, encryption key sizes, text lengths, and formats. Custom-developed Python scripts were used to implement the tests, and the results were analyzed using a Microsoft Power BI dashboard for detailed visualization. The findings highlight that RC4 demonstrated the fastest performance across all tested metrics, whereas DES exhibited the longest execution time. AES showed moderate performance compared to the other algorithms. These insights provide valuable guidance for researchers and practitioners on selecting encryption algorithms based on performance requirements, contributing to improved database security and operational efficiency.

Keywords: *Symmetric encryption algorithm, database security, data protection, performance analysis, AES, DES, Blowfish, RC4, ChaCha20*

1. INTRODUCTION

A. Background of Study

In the modern digital era, data security has become a vital concern across different kinds of industry and organizations, increasingly relying on databases to store sensitive information. Encryption, a process of encoding data to become messy code that cannot be directly understood by user is widely used in databases to prevent unauthorized access. Databases are developed and provide services to many industries including finance, healthcare, and e-commerce. As these systems are frequently facing cyberattacks leading to severe data breaches and financial losses, an iterative improvement towards the encryption techniques is essential. Their implementation challenges are usually related to performance, compatibility, and usability. An encryption technique works efficiently with databases to ensure sensitive information is accessible only to authorized users, to guarantee that data is not altered or tampered during storage or transmission, and to help organizations to meet legal and regulatory requirements for data protection. It should have a strong and non-vulnerable rampart to shield databases from cyberattacks. Shifting focus to the database, it simplifies the management of large datasets and makes the information easier to be accessed by enabling the storage centralization. It also preserves the data to maintain their consistency, reduce redundancy and ensure accuracy across systems. Databases nowadays supports scalability to fit the growing demand for data, facilitate business growth. The function of automated backups and ability of data recovery are important to ensure minimal data loss during the event of failures.

B. Problem Statement

Databases serve crucial functions in storing sensitive information, making encryption become an essential tool to maintain database protection. However, encryption could reduce the efficiency of database operation, or the encryption techniques sometimes are hard to cope with large or distributed databases. Thus, organizations must use encryption methods that not only meet data protection laws but also remain practical and user friendly to guarantee smoothness in normal workflow without causing disruption. Although this may be an important issue, it is noticeable that there is a lack of research exploring how different symmetric encryption algorithms perform across different database systems.

C. Aim and Objectives (RQs)

This paper aims to evaluate the performance of 5 well-known symmetric encryption algorithms – AES, DES, Blowfish, RC4 and ChaCha20 under varying configurations within different databases. By carrying out this analysis, this study able to provide useful information and insights in contributing to the field of database security. This study is guided by clearly defined research objectives (ROs) and research questions (RQs) as follows to systematically explore and analyze these algorithms thoroughly.

Research Objectives (ROs):

1. To analyze the encryption and decryption performance of AES, DES, Blowfish, RC4 and ChaCha20 under varying conditions (database types, key sizes, text lengths and text types)
2. To evaluate the impact of studied encryption algorithms on the efficiency of Data Manipulation Language (DML) operations (insertion, selection and update)
3. To visualize and compare the performance metrics using interactive platforms
4. To identify the most efficient algorithm(s) that provides the best performance

Research Questions (RQs)

1. How do different symmetric encryption algorithms perform in terms of encryption and decryption time across varying database configurations?
2. What is the impact of these algorithms in terms of DML operations' performance?
3. How do various factors such as encryption key size, text length and text formats affect algorithm performance?
4. Which algorithm(s) offer the best performance efficiency?

D. Scope of Study

This research focuses on evaluating and analyzing algorithms under various conditions which include diverse databases (MySQL, SQLite, MongoDB and Apache Cassandra), varying key sizes ranging from 40 bits to 448 bits, text lengths (short, medium and long characters) and text formats (alphabetic, numeric and alphabetical data). A series of experiments were conducted using custom Python scripts with the results compiled and visualized in Microsoft Power BI dashboard for detailed analysis.

E. Significance of Study

The findings obtained from this research able to contribute to the research field particularly in the field of database security. By carrying out this research, this may help bridge the gap between data security and efficiency which may help researchers and practitioners in designing or implementing secure and performant database systems.

F. Overall Report Structure

The report is structured into 5 sections, with the following section presenting a detailed literature review which delves into previous studies on encryption algorithms and their impact on database performance. Chapter 3 describes the methodology employed in evaluating algorithms' performance with various performance metrics. Whereas chapter 4 describes the key findings of the performance evaluation based on various metrics. Finally, the conclusion chapter summarizes the key findings and suggests future research directions in the field of database encryption.

2. LITERATURE REVIEW

A. Algorithms Used

1. AES (Advanced Encryption Standard)

The original name of AES algorithm was Rijndael algorithm [1]. It was authenticated by NIST (National Institute of Standards and Technology) in 2000 after a five-year standardization process to replace DES [2], [3]. AES is a symmetric block cipher which encrypts data in fixed-size blocks such as 64 bits and 128 bits [1]. The key size is the main determination for the number of encryptions rounds and the overall security. This might be determination for number of encryptions rounds and the overall security. The 128-bit, 192-bit and 256-bit keys are widely recognized with 10, 12 and 14 rounds respectively [2]. The state in AES is 4x4 matrix of bytes, which is used to represent the encrypted or decrypted data block during the process [1]. AES is well-known by academic workers and related industry for its strong security and itself is a standard for encrypting sensitive data [3], [4]. However, it requires intensive calculations which may not be optimal for situations with very tight resources [3].

2. Blowfish

Blowfish is a symmetric block cipher designed by Bruce Schneier as a general-purpose algorithm to replace DES in 1993 [1]. Blowfish is commonly used at a place where the key does not often change [4]. Specifically, its key size has a flexible length which could be in between 32 to 448 bits [4]. This flexibility of key length acts as one of the strengths of Blowfish as the users are allowed to choose a key size that fits the security level they need. They can select short keys for faster processing or long keys to enhance security, absolutely depending on the situation they are in. In contrast to DES, a longer key able to provide stronger security and this makes it more difficult to destroy by brute-force attacks. However, the key changing process is very slow in Blowfish as a pre-processing equivalent is required for every new key [4]. As Blowfish works in a 64-bit block, it is defenseless towards birthday attacks [4].

3. DES (Data Encryption Standard)

DES is an algorithm founded on the fundamental idea of Feistel Structure [5], which was standardized in January 1977. It is a symmetric block cipher which encrypts and decrypts in a fixed 64-bit block size with 16 rounds. Padding will be added to complete the block size with 16 rounds. It will be added if the input data is not a multiple of 64 bits. The key is 64 bits but only 56 bits are used in DES for encryption and decryption as the other 8 bits are used for error detection or as parity bits [5]. DES was a standard before as it was simple and easy for implementation, but it still not secures enough as its short key length is considered weak when facing modern brute-force attacks [5].

4. RC4 (Rivest Cipher 4)

RC4, also known as ARC4 (Alleged RC4), is a symmetric stream cipher designed by Ron Rivest in 1987. As stream cipher processes data bit-by-bit or byte-by-byte, it does not require an input of a certain amount of data before the encryption process [6]. Key setup phase and pseudorandom key stream generator phase are the two phases that need to be undergone in RC4 for encryption and decryption [4], [7]. Key setup phase is the first step of the process. This phase needs a flexible length key which length from 1 to 256 bytes to build a 256-byte array. This array is essential as it prompts the creation of pseudorandom bytes which must be done whenever a new key is taken into use to ensure a unique pseudorandom stream for every session [7]. After that, the pseudorandom key stream generator phase generates a pseudorandom stream of bytes to produce ciphertext for encryption, or vice versa for decryption. Like the key setup phase, this phase needs to be carried out if any new key is generated to ensure the security and integrity of the data is being encrypted or decrypted [7]. However, RC4 has pivotal security vulnerabilities. It will become very insecure if the start of the output keystream is not removed, or if the keystream is used twice [4].

5. ChaCha20

ChaCha20 is designed by Daniel J.Bernstein as a modification of Salsa20 cipher [2],[8]. ChaCha20 is lightweight as Daniel initially wanted to maximize the diffusion by increasing the amount of diffusion for each round, but he found that the minimum number of secure rounds for ChaCha20 algorithm is smaller than that for Salsa20. This clearly showed that if ChaCha20 has the same minimum number of secure rounds as Salsa20, ChaCha20 will have a better overall throughput than Salsa20 for the same level of security [9]. ChaCha20 is a symmetric stream cipher with 256-bit keys and 64-bit nonces [2]. A nonce, which is a value that should never be reused for the same key, is used to ensure that each encryption operation with the same key produces a unique ciphertext [4]. ChaCha20 is known for its excellent throughput of encryption and decryption, performing better AES on CPUs without hardware acceleration [3]. However, while ChaCha20 is highly efficient, it is not as mature as AES, and this might limit its implementation in some older system, software or hardware [3].

B. Previous Studies on Database Encryption

From the critical analysis of Symmetric Key Cryptographic of Nema and Rizvi [10], Blowfish is considered excellent among other well-known algorithms such as AES, DES and 3DES in terms of security, flexibility, memory usage, and encryption performance. Besides, Tyagi and Fanpati [10] also reviewed the performance of symmetric key encryption algorithms and concluded that Blowfish outperformed those algorithms mentioned above relating to encryption time, decryption time, and throughput. Rohit and Aman [1] conducted a study to determine the most secure algorithm with high performance and discovered that AES showed the best performance as it has the smallest total of encryption and decryption time. Avalanche Effect is a phenomenon even if a small alteration in plaintext, it will change significantly in the ciphertext, with multiple bits will be affected [1], [5]. It was declared that if the Avalanche Effect of an algorithm was over 50%, then it is a good algorithm [1]. The comparative study undertaken by Youssouf, Siti and Maheyzah [5] was focused on the strength of each algorithm towards Avalanche Effect. The results obtained show that AES has the best security level among DES, CAST-128 and

Blowfish. Blowfish is the second best, but DES is considered insecure as its short key length is insufficient against modern brute force attack. The previous study on evaluating various algorithm performance was presented and concluded thoroughly as shown in Table 1.

TABLE I. Summary of Related Works

Reference	Aims	Gap/Problem	Methodology	Findings	Future Work
[1]	This paper aims to examine the algorithm offering the highest security and performance among AES, DES, TDES, RC6 and Blowfish.	Lack of comprehensive analysis considering both performance and security aspects across various algorithms, insufficient information on the trade-offs between different algorithms for specific use cases	Java 9.0 was used for implementing algorithms. Avalanche Effect was calculated to compare the security of algorithms. The higher the Avalanche Effect, the more secure of the encrypted data.	AES shows the best performance as it has the smallest sum of encryption and decryption time. RC6 has the highest Avalanche effect in percentage, indicating that RC6 is the most secure algorithm for encryption.	Future work may consider using different programming languages, as the difference of languages handling the byte ordering may affect how the algorithm processes data. Other factors affecting algorithm performance should also be examined.
[5]	This paper focuses on comparing symmetric key algorithms based on Avalanche Effect and integrity checking.	The previous research mostly focused on individual performance such as speed and security but did not mainly evaluate the algorithms across parameters such as Avalanche Effect, integrity, flexibility, and scalability.	Crypto tool was used by changing bits in encryption key and generating new cipher text using the modified key. The original and modified cipher texts were compared by calculating number of changing bits with respective percentage.	DES showed the strongest Avalanche Effect in both ECB and CBC modes compared to others. While AES has the strongest integrity in ECB mode, Cast-128 has the strongest integrity in CBC mode.	Future research could investigate the performance of application of algorithms in real-world applications and exploring the algorithms' suitability towards cloud environments.
[10]	This paper aims to evaluate and compare the performance in terms of time and throughput of three symmetric encryption algorithm – AES, Blowfish, and Twofish.	There exist inconsistencies in the efficacy of these encryption algorithms.	Twofish, Blowfish and AES with 128 bits were evaluated using Python 3.10. Various file formats are used for comparison and the experiment was carried out 3 rounds and mean values were computed for each algorithm.	The process time is compared by calculating mean of encryption and decryption time. Experiment shows that AES has the best performance in between all encryption algorithms in terms of speed and processing time.	Comparative study with a larger dataset is needed to ensure the scalability of those algorithms.
[11]	This paper conducts a performance evaluation of symmetric data encryption algorithms between AES and Blowfish.	Existing studies often focus on limited data types, specific parameters, or comparisons among multiple algorithms without addressing the diverse data types or considering the efficiency of the algorithms across them comprehensively.	The study was conducted using the JDK 7.1 Java development kit. The data block sizes used were 128 bits for both algorithms, and the throughput was calculated. A prototype interface was set up to allow the user to interact with the application through a GUI.	For images, videos, audio and files, AES has higher encryption average time and Blowfish has higher throughput. Therefore, it was concluded that Blowfish is more efficient as compared to AES.	Future work can focus on decryption time, memory usage, CPU utilization, power consumption and resistance to various cryptanalytic attacks. More algorithms should also be covered in future studies.
[12]	This paper compares the execution times for asymmetric keys algorithms – RSA, ELGAMAL and ECIES using different sizes of encryption	The increasing complexity of those algorithms involves higher execution times, leading to an application performance decrease.	Java was used by deploying BouncyCastle API. Encryption and decryption were tested by strings 10, 20, and 30 characters. 192, 256 and 1024 bits were used, and results were obtained by	For encryption, ECIES has the longest encryption time whereas ElGAMAL has 7 times longer encryption time than RSA. For decryption, ECIES has taken the	Future work could focus on exploring the hybrid encryption schemes combining symmetric and asymmetric.

	and decryption keys.		performing 10 runs for each algorithm.	longest time whereas RSA has the shortest decryption time.	
[13]	This paper analyses recent works in encrypted database techniques to clarify the pros and cons points.	Lack of numerical analysis for readers to deeply understand the difference among different techniques.	Data was identified and collected based on strengths, effectiveness, limitations and specific use cases of each technique. A table was made to compare their pros and cons in different methodology used.	REA is secure but unsuitable for distributed systems, symmetric methods are fast but limited, and hybrid approaches boost security with high overhead. Column encryption improves queries but has vulnerabilities.	A novel proposal must be suggested to achieve high security and performance in encryption and time of query on encrypted database techniques.
[14]	This paper extends the TSFS algorithm's dataset to special characters and improves substitution and shifting.	TSFS face challenges in handling special characters, avoiding errors during decryption, and maintaining efficiency in terms of storage and query processing time.	Extending of TSFS to support special characters and correcting its substitution and shifting processes by introducing multiple modulo factors and four 16-arrays for different data types.	Enhanced TSFS has better encryption performance without increasing data size or affecting query time. The proposed ETSFS used the smallest space and encryption time.	ETSFS algorithm has not been tested for scalability with extremely large datasets or databases under heavy concurrent user access.
[15]	This paper deals with the search cost problem of the tag-based SSE in constructing index generation suitable for the tag-based searchable symmetric encryption SSE and DBMSs.	Tag-based SSE is difficult to construct any index from the tags, therefore sequential checking is needed which is the search cost depending on the number of tags stored in the server.	IG-TSSE was constructed by using a short bit string containing output value of deterministic encryption to allow construction of index such as B-tree which supported by many DBMSs from deterministic tags.	IG-TSSE lowers the search costs and at the same time the security is not degraded after disclosed search patterns.	The application of well-known attacks against searchable encryption and study their effectiveness.
[16]	This paper aims to explore and analyze Searchable Symmetric Encryption (SSE)	As previous studies on SSE have been broken by the research community, some risks when deploying SSE should be considered.	Functionality and comparison between PEKS and SSE, advantages, opportunities and challenges of implementing SSE in Switzerland were analyzed	SSE advanced in functionality, provides efficient search with strong privacy guarantees.	Data security needs to be carefully assessed as the secure usage of SSE approaches is very challenging. This can be enhanced in the future.
[17]	This paper evaluates homomorphic encryption (HE) techniques, compares all database encryption problems and developments, examined benefits and drawbacks of homomorphic approaches.	There needs further research on the applicability of FHE in real applications.	Review of encryption mechanisms such as TDE, CLE, PHE, SHE and FHE in terms of security, computational requirements, and practical applications, and mainly focuses on FHE advancements and challenges.	HE allows businesses to share private data with third parties to get computational service securely as whoever has access only to the encrypted data to perform computations.	Improvement of FHE systems in terms of security, simplicity, and especially speed. As they are too expensive in real-world applications, FHE needs to be greatly improved to be more practical on all platforms.

C. Factors Influencing Encryption Performance

1. Database-Specific Factors

Database related factors include indexing and query optimization. Efficient indexing can heavily affect the encryption processes by scanning and filtering out the data that no need to be encrypted or decrypted, which might save computational resources. Indexing has a great advantage in terms of retrieval performance enhancement by maintaining sequence characteristics and enhancing query processing speed [18]. When the columns containing encrypted data are indexed, the encryption method may make huge differences in how indexes are utilized. For instance, indexing is allowed in the case of deterministic encryption as the same plaintext always produces the same ciphertext. However, for the randomized encryption, as it produces different ciphertexts for the same plaintext, it makes efficient queries impossible. The indexing closely related to query optimization. Query optimization is for the purpose of minimizing unnecessary data scanning, joining or computations. A user can select if he/she intends to choose between a full table or using an index.

2. Algorithm-Specific Factors

Algorithm related factors are focused on the key size of the algorithm used for encryption or decryption, block size the algorithm process in one operation, and the computational complexity of the particular algorithm. The key size of the encryption algorithm mostly affected the security and integrity of the encrypted data. From Ariel and Ruji [19], the dependence of encryption algorithms integrity on the key size was mentioned. A longer key is harder to break than a shorter key, therefore the intruder could decrypt the data easily if a shorter key is used. In [5], DES was stated as an encryption algorithm which does not preserve the security measures. It has the lowest security level compared to other algorithms that had been used in that paper, which is caused by its short key length of 56 bits. However, different algorithms may take different key sizes to reach similar security levels. From analysis of Nigel and Emmanuel [20], it can be concluded that 128-bit key has different security levels when facing attacks in different algorithms. The 128-bit key of AES is suitable for it to be against quantum attacks, but it is weak in RSA as it can be broken easily [20]. The block size of an encryption algorithm might affect the throughput and security. Equation (1) and (2) show the results of the number of blocks with different block sizes for 1MB (8,000,000 bits) of plaintext.

$$\text{Number of blocks} = \frac{8{,}000{,}000}{64} = 125{,}000 \text{ blocks} \quad (1)$$

$$\text{Number of blocks} = \frac{8{,}000{,}000}{128} = 62{,}500 \text{ blocks} \quad (2)$$

It can be easily observed that a larger block size can reduce the number of blocks that need to be processed [21]. However, the decision needs to be made carefully when deciding the block size, as inappropriate sizes may affect decryption processes [21]. The computational complexity which involves the time complexity of an algorithm, resource consumption of CPU time, memory, battery power and the scalability of the algorithms to work when the data size keeps expanding. Cryptographic algorithms with high computational complexity will need more time to complete the process and resources. According to [22], AES-128 has a time complexity of $O(n)$, which implies the performance increases linearly with input size, efficient for both encryption and decryption processes. However, more complicated algorithms used in fully homomorphic encryption could be slower.

3. Hardware and Environment Factors

Hardware and environment factors such as hardware accelerators and energy efficiency are the key considerations to determine the suitability of encryption schemes from high-performance systems to embedded devices. A hardware accelerator can also be described as a special-purpose processor as it executes certain tasks much faster than a general-purpose CPU would. It performs encryption and decryption faster than the CPU as it executes multiple operations at the same time. The implementation of cryptographic hardware accelerators on dynamically reconfigurable platforms can enhance the encryption significantly in terms of encryption and decryption of data, hashing, and key generation [23]. Besides the enhanced utilization of resources, accelerators such as ACA-SDS can be used to improve energy efficiency [24]. Energy efficiency serves as another important factor that makes a strong impact on encryption performance, especially in wireless sensor networks (WSNs) and mobile devices whose operations depend on the resource-constrained environments. For example, encryption algorithms such as AES and RSA required significant computational resources for a complete encryption operation. Efficient hardware reduces the energy required to enhance both performance and energy savings.

D. Symmetric Encryption Applications in Real-World Databases

Symmetric encryption is vital for securing data during the transaction or while the data are stored in Fintech applications. It is commonly implemented to encrypt sensitive financial documents such as information about payment, user credentials and transaction details, whether sent across networks or stored on servers [25]. Their advantages include efficiency in encryption and excellent throughput that make them find their ideal application in securing data storage and real-time transaction processing in mobile payment apps and digital banking systems [25]. As AES possesses strong security, high throughput and low latency, it is widely accepted by many fintech applications [25]. Other than that, Searchable Symmetric Encryption is trusted to be used in the medical industry. In medical cloud computing, the Searchable Symmetric Encryption is used to protect sensitive and private information such as medical case and diagnostic report, while at the same time the search function can be used when the data is encrypted [26]. Besides, symmetric encryption safeguards the transmission of physiological signals in telemedicine. These encryptions are achieved in enhancing security, ensuring data confidentiality and integrity in remote medical services by integrating chaotic maps and machine learning networks [27]. Symmetric encryption is also broadly used in the e-commerce industry where AES and DES form the integral part that ensures data confidentiality. It is usually part of a more comprehensive security framework which involves digital envelopes and message authentication to maintain data integrity and authenticity [28], [29].

3. RESEARCH METHODOLOGY

The methodology conducted in this research is experimental and comparative-based study, and it serves as a quantitative study which analyzes the algorithm performance in numerical measurement. This study is clearly divided into 2 main phases, with the first phase of the study aims to evaluate both encryption and decryption performance of 5 symmetric encryption algorithms (AES, DES, Blowfish, RC4 and ChaCha20). The performance evaluation has been conducted by evaluating various criteria, whereas the second phase of the research involves selecting the key sizes yielding the best performance from each algorithm in Phase 1 to simulate a real-world payment and transaction system which implements field-level encryption to visualize its performance across both unencrypted and encrypted data. The evaluation of performance for both Phases 1 and 2 were presented via interactive Microsoft Power BI dashboard function for further analysis and detailed visualization. Figure 1 shows the overall flowchart involved in this study to provide a clear workflow carried out in this study.

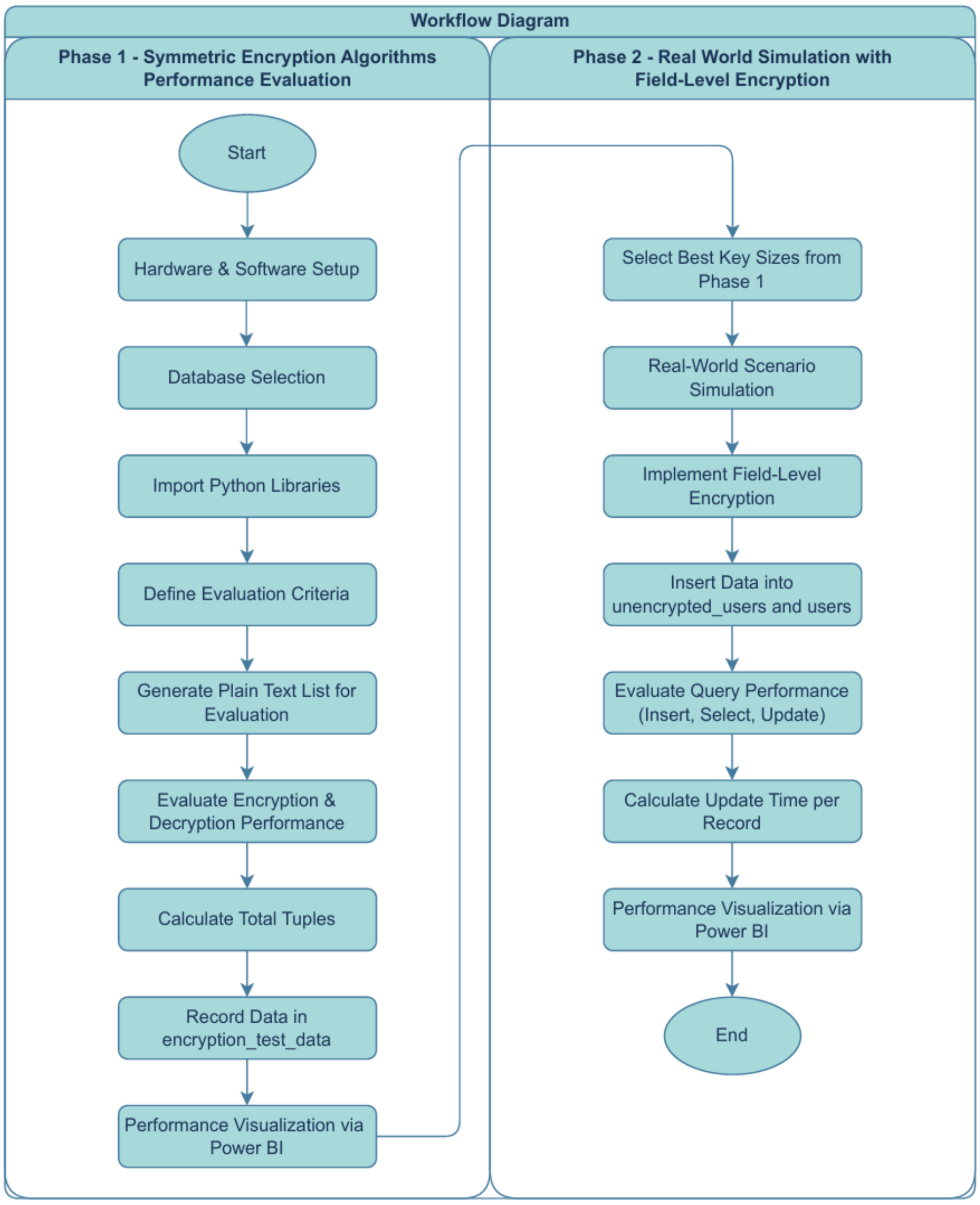

Workflow Diagram
Phase 1 - Symmetric Encryption Algorithms Performance Evaluation
Phase 2 - Real World Simulation with Field-Level Encryption
Start
Hardware & Software Setup
Database Selection
Import Python Libraries
Define Evaluation Criteria
Generate Plain Text List for Evaluation
Evaluate Encryption & Decryption Performance
Calculate Total Tuples
Record Data in encryption_test_data
Performance Visualization via Power BI
Select Best Key Sizes from Phase 1
Real-World Scenario Simulation
Implement Field-Level Encryption
Insert Data into unencrypted_users and users
Evaluate Query Performance (Insert, Select, Update)
Calculate Update Time per Record
Performance Visualization via Power BI
End

Fig 1. Workflow Diagram

All phases were conducted with the following hardware configurations: AMD Ryzen 7 5800U with Radeon Graphics running at 1.90 GHz and 16GB DDR4 RAM with the speed of 3200 MHz. The processor belongs to Ryzen 5000 series based on Zen 3 architecture, and it is considered as Generation 5 AMD's Ryzen processor. The Solid-State Drive (SSD) model used for conducting this research was SK Hynix HFS512GD9TNI-L2A0B. The study was also conducted on a Microsoft Windows 11 Pro operating system with version 23H2.

The software and technologies employed in this research are presented as follows: PyCharm Community Edition 2023.1 as Integrated Development Environment (IDE) for encryption algorithm implementation using Python 3.13.1. Besides, database engines such as MySQL version 8.0.36, SQLite3 version 3.47.0, MongoDB v8.0.3 and finally Apache Cassandra with Cassandra Query Language (CQL) version 3.4.7 were also deployed in carrying out this research.

A. Phase 1 – Performance Evaluation of Symmetric Encryption Algorithms

The first phase of the study started off with evaluating the capability and performance of various encryption algorithms based on various performance metrices. Symmetric encryption algorithms such as AES, DES, Blowfish, RC4 and ChaCha20 were selected for conducting the study. These algorithms were then tested on various databases, with relational database management systems (RDBMS) – MySQL and SQLite3, as well as non-relational databases (NoSQL) – MongoDB and Apache Cassandra.

To establish connections between Python and respective studied databases, specific libraries were employed for each database by importing Python libraries and database client modules as shown in Table 2. In this study, the algorithms' performance was measured based on the following criteria.

1. Database

Table 2 shows a comprehensive breakdown on the database-specific libraries imported for each database with respective connector details.

TABLE II. Python Database Connection Libraries

	Database	**Connector**	**Installation**	**Connection**
RDBMS	MySQL	mysql-connector-python	pip install mysql-connector-python	mysql.connector.connect()
	SQLite	Built-in sqlite3	No installation required	sqlite3.connect()
NoSQL	MongoDB	pymongo	pip install pymongo	MongoClient
	Apache Cassandra	cassandra-driver	pip install cassandra-driver	cassandra.Cluster

2. Key Size

Table 3 presents the breakdown of key sizes used by each algorithm in this study. There were 11 key sizes of the algorithms to be tested out in this study to thoroughly examine the performance of encryption and decryption time under different key sizes.

TABLE III. Key Sizes Used for Each Algorithm

Algorithms	**Studied Key Size (bits)**
AES	128, 192, 256
DES	192
Blowfish	128, 256, 448
RC4	40, 128, 256
ChaCha20	256

3. Text Length

Table 4 shows the benchmark of text length used in this study which divides the text length into short, medium and long to evaluate the capability of algorithms in encrypting and decrypting varying length of texts.

TABLE IV. Text Length Benchmark

Text Length	Number of Characters
Short	10 – 19
Medium	20 – 45
Long	46 – 50

4. Text Types

Table 5 depicts the types of plain text to be encrypted and decrypted with respective descriptions. This study examines the ability of each algorithm in encrypting numerical, alphabetical and alphanumerical data.

TABLE V. Text Types with Description

Text Types	Description
Numeric	Consists of only numbers
Alphabetic	Consists of only alphabets
Alphanumeric	Combination of numbers, alphabets and special symbols

Based on the criteria mentioned above, a plain text list that meets Criteria 3 and 4 was developed as shown in Table 6 for evaluating encryption and decryption performance across each algorithm for ensuring consistency of the study. There were 18 plain texts to be encrypted and decrypted for each algorithm across 4 databases.

TABLE VI. Constructed Plain Text Used in This Study

	Numeric	Alphabetic	Alphanumeric
Short	•1234567890 •9876543210	•HelloWorld •Encryption	•P@ssw0rd!1 •A1B2C3D4E5
Medium	•12345678901234567890 •09876543210987654321	•DataEncryptionTest •SecureDataTesting	•P@ssw0rd2023!Secure •Test123!Encryption
Long	•12345678901234567890123456789012345678901234567890 •98765432109876543210987654321098765432109876543210	•ThisIsALongPlaintextForEncryptionTestingPurposes •SecureYourDatabaseWithProperEncryptionMethods	•LongP@ssw0rd123!SecureDataEncryptionTest2023!# •Encryption$Mix123!DataTestForAnalysis2023!AB

In accordance with the criteria mentioned above, a table named *encryption_test_data* was constructed for each database to specifically document the time taken for encryption and decryption in milliseconds (ms) together with respective attributes as presented in Table 7.

TABLE VII. Attributes in encryption_test_data Table

Attributes	**Description**
id	Unique identifier
plain_text	Input data to be encrypted
encrypted_data	Resulting encrypted output
encryption_algorithm	Encryption algorithms used
key_size	Size of the encryption key
encryption_time	Time taken for encryption
decryption_time	Time taken for decryption
encryption_key	Key used for encryption

Equation (3) shows the calculation of the number of rows (tuples) yielded from each database by multiplying the number of key sizes and number of plain texts included in this study.

$$T = K \times P \qquad (3)$$

where T represents the number of tuples for each database, K represents the number of key sizes included, and P represents the number of plain texts included.

Whereas (4) presents the calculation of acquiring the total number of tuples across all databases in this study.

$$\text{Total Tuples} = \sum_{i=1}^{D} T_i \qquad (4)$$

where D represents the total number of databases, and T_i represents the number of tuples per database.

Based on the equations above, given that there are 11 key sizes as depicted in Table 3 and 18 plain texts as depicted in Table 6 to be employed in this study, the number of tuples included for each database table was computed as shown in (5) and (6).

$$T = 11 \times 18 = 198 \text{ tuples per database} \qquad (5)$$

$$\text{Total Tuples} = 4 \times 198 = 792 \text{ tuples} \qquad (6)$$

Referring to (6), the algorithm performance analysis was based on 792 tuples across both RDBMS and NoSQL. A Microsoft Power BI dashboard was built for visualizing the performance based on the criteria mentioned with the following details being presented in Table 8. In this study, Microsoft Power BI was utilized in analyzing and visualizing algorithm performance due to its Graphical User Interface (GUI) that allows users to easily manipulate the data and gain useful insights via interactive dashboards.

To establish a connection link from each database to Microsoft Power BI, Open Database Connectivity (ODBC) drivers were utilized for MySQL, SQLite and Apache Cassandra to enable seamless data retrieval for performance visualization. Additionally, a Comma-Separated Values (CSV) import method was employed to fetch and retrieve data from MongoDB to Microsoft Power BI for further analysis.

TABLE VIII. Microsoft Power BI Dashboards in Phase 1

Microsoft Power BI	**Description**
Dashboard 1	Presents overall performance of both encryption and decryption for each algorithm
Dashboard 2	Presents encryption and decryption performance across different database
Dashboard 3	Presents encryption and decryption performance specifically based on various

	key sizes
Dashboard 4	Presents encryption and decryption performance specifically based on various text length
Dashboard 5	Presents encryption and decryption performance specifically with different types of plain texts

B. Phase 2 - Real-World Simulation with Field Level Encryption

To proceed to the second phase of this study, a real-world scenario simulation was developed to test out the query performance for each encryption algorithm across various databases. A payment or transaction system was simulated by creating a table named *users* and *unencrypted_users* with the following attributes as shown in Table 9.

TABLE IX. Attributes Included in users and unencrypted_test_data Tables

Attributes	**Description**
id	Unique identifier
name	User’s name
email	User’s email for transaction
creditcard	Credit card for payment

For the table *users*, given the sensitive nature of data stored in a payment system such as the attributes *email* and *creditcard*, field-level encryption was implemented in these attributes for ensuring data security from any unauthorized access. The field-level encryption was applied using both 5 encryption algorithms (AES, DES, Blowfish, RC4 and ChaCha20) with the key sizes selected for each algorithm based on its best performance evaluated in Phase 1. Whereas the table un*encrypted_users* stores all user records without implementing any encryption method for showing the comparison.

1000 random tuples were then constructed using Python scripts and inserted into each database for both encrypted table (*users* table) and unencrypted table (*unencrypted_users* table). This process was repeated for each encryption algorithm to ensure a consistent sample size across all databases for better comparison and performance evaluation. After that, the query performance was evaluated via the following DML operations as shown in Table 10. The performance metrices were evaluated towards both encrypted data (*users* table) and unencrypted data (*unencrypted_users* table).

TABLE X. DML Operations for Performance Evaluation

Operations	**Description**	**Performance Metric**
SELECT	To retrieve all records from database	Time taken to select all records in database
INSERT	To insert 1000 records into database	Time taken to insert 1000 records
UPDATE	To update the *name* attribute for records where the *email* attribute ends with *@example.com*	Time taken to update all matching records

Additionally, to ensure a concise and precise evaluation of query performance, the update time per record was computed as well using (7) to account for the varying number of update statements to the 1000 random tuples.

$$\text{Update Time per Tuple (s)} = \frac{\text{Update Time}}{\text{Tuples Updated}} \qquad (7)$$

Similarly, the performance of selection time, insertion time and update time taken by each encryption algorithm across all studied databases were documented in Microsoft Power BI dashboard as well to provide comprehensive visualization and comparison. The dashboard details involved in this phase are outlined as in Table 11.

TABLE XI. Microsoft Power BI Dashboards in Phase 2

Microsoft Power BI	**Description**
Dashboard 6	Presents the DML operation performance across encrypted and unencrypted data in terms of query selection, insertion and update time
Dashboard 7	Presents the DML operation performance and analysis specifically for encrypted data

4. RESULTS AND DISCUSSIONS

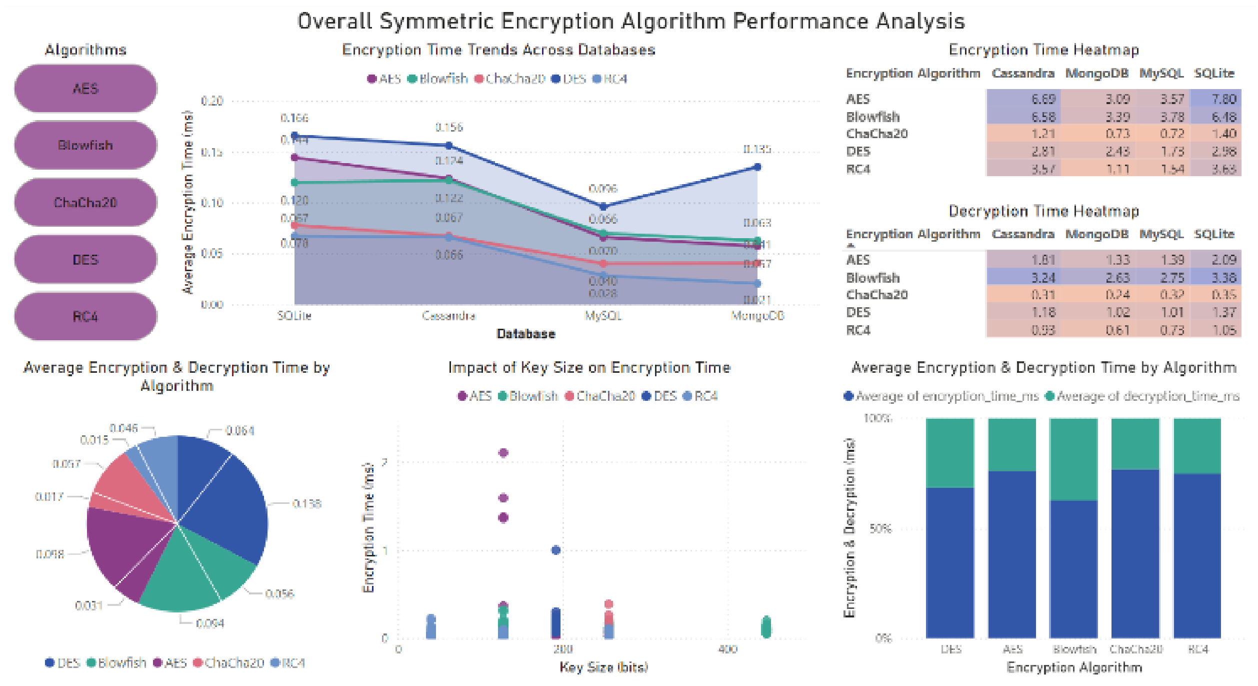
The results obtained from this study for both Phases 1 and Phase 2 were represented in Microsoft Power BI Dashboard function as attached in the provided URL (Appendix A.1). The algorithm performance analysis was derived based on the output obtained from running the Python scripts (Appendix A.2), and the results were compiled in both Microsoft Excel Open XML Format Spreadsheet (XLSX) and CSV formats (Appendix A.3) for maximizing transparency and clarity of the results. The dashboard provides a clear overview focusing on the performance metrics of the encryption algorithms across different databases, encryption key sizes, text length and plain text types. The key findings were categorized into 2 phases similar to the methodology section – algorithm performance analysis and real-world scenario simulation.

Additionally, this section provides an analysis of the overall performance of the symmetric encryption algorithm using the default settings for each dashboard without applying any additional filter. For a more detailed and dynamic visualization of various performance metrics, readers are highly encouraged to interact with the dashboards in Microsoft Power BI by utilizing the available settings to explore the performance across various metrics.

A. Phase 1 - Symmetric Encryption Algorithm Performance Analysis

For analyzing the algorithm performance thoroughly, the results and discussion were presented based on the criteria examined as shown in methodology section, including the analysis across various databases, key sizes, text lengths and text types.

1. Overall Performance



Encryption Time Heatmap

Encryption Algorithm	Cassandra	MongoDB	MySQL	SQLite
AES	6.69	3.09	3.57	7.80
Blowfish	6.58	3.39	3.78	6.48
ChaCha20	1.21	0.73	0.72	1.40
DES	2.81	2.43	1.73	2.98
RC4	3.57	1.11	1.54	3.63

Decryption Time Heatmap

Encryption Algorithm	Cassandra	MongoDB	MySQL	SQLite
AES	1.81	1.33	1.39	2.09
Blowfish	3.24	2.63	2.75	3.38
ChaCha20	0.31	0.24	0.32	0.35
DES	1.18	1.02	1.01	1.37
RC4	0.93	0.61	0.73	1.05

Fig 2. Overall Performance of Algorithms

The first dashboard presented in Microsoft Power BI is as shown in Figure 2. It provides an overall view of the performance of each algorithm conducted in the study. Based on the pie chart named ‘Average Encryption & Decryption Time by Algorithm’ and area chart titled ‘Encryption Time Trends Across Databases’, the distribution of **DES** takes up the largest portion of the diagrams, which indicates that it has the highest processing time among the remaining algorithms. In contrast, **RC4** occupies the least portion on both diagrams, reflecting that it has the fastest speed in encrypting and decrypting data. Additionally, as observed from 100% stacked column chart named ‘Average Encryption & Decryption Time by Algorithm’, the total encryption time is greater than the decryption time. This difference exists due to the extra computational steps performed such as key expansion during the encryption steps. While the decryption time is shorter as decryption typically involves reversing the encryption operations which is easier and less complex. The overall ranking of algorithms on average is as shown in Table 12.

TABLE XII. Ranking of Algotihms on Average

	Performance (Fastest to Slowest)
Encryption	RC4 < ChaCha20 < AES < Blowfish < DES
Decryption	RC4 < ChaCha20 < AES < Blowfish < DES

2. Algorithm Performance by Database

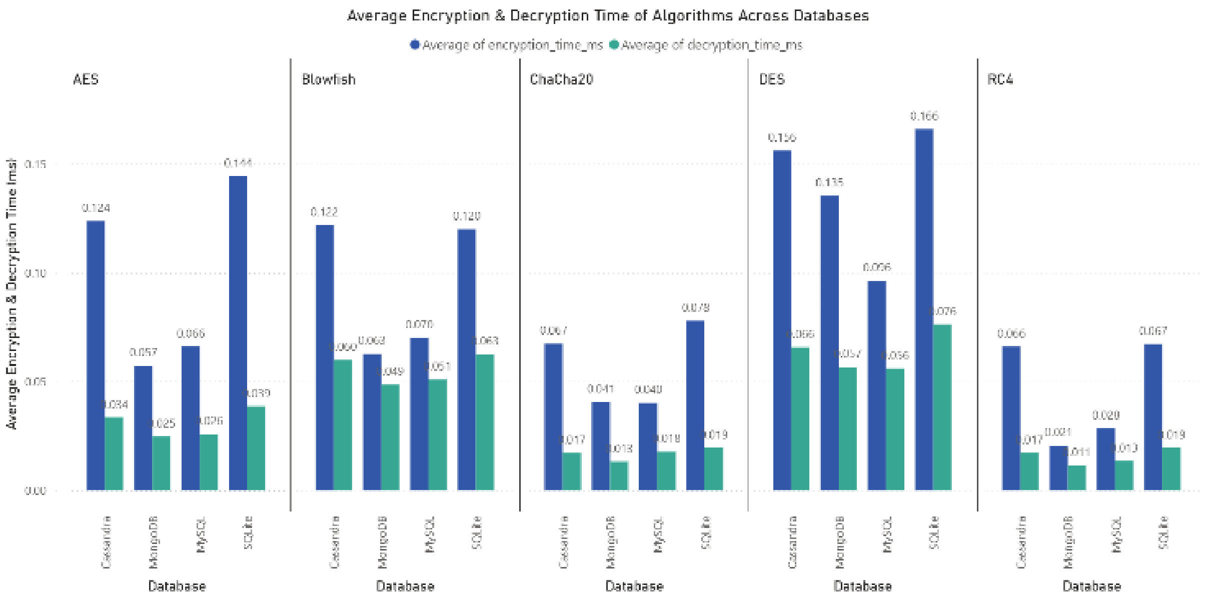


Fig 3. Algorithm Performance Across Databases

TABLE XIII. Time Taken by Algorithm in Best Performing Databases

Algorithm	**Best Performing Database**	**Encryption Time (ms)**	**Decryption Time (ms)**
AES	MongoDB	0.057	0.025
DES	MySQL	0.096	0.056
Blowfish	MongoDB	0.063	0.049
RC4	MongoDB	0.021	0.011
ChaCha20	MongoDB	0.040	0.018

The clustered column chart depicting the average encryption and decryption time taken in milliseconds (ms) by AES, DES, Blowfish, RC4 and ChaCha20 was presented in Figure 3. Based on the diagram, the encryption and decryption time taken appears to be the highest among Apache Cassandra and SQLite as compared to the other databases. On the whole, **RC4** algorithm stands out to be the fastest algorithm across all databases with the encryption and decryption time being nearly identical. Its best performance can be observed in MongoDB (0.021 ms encryption and 0.011 ms decryption). Apart from that, **ChaCha20** algorithm stands out as an efficient algorithm as well by having a close and low encryption and decryption times across all databases and the encryption time is only 0.078 ms even in the slowest database (SQLite). In this context, ChaCha20 appears to be one of the fastest and most consistent algorithms tested. Whereas **AES** and **Blowfish** algorithms perform similarly in terms of the encryption time, however Blowfish has higher decryption time as compared to AES. Conversely, **DES** algorithm shows relatively higher encryption and decryption time in comparison with the other algorithms, with SQLite and Apache Cassandra having higher encryption and decryption time, which aligns with the older architecture and computationally intensive design. Based on Table 13, it is clear that MongoDB turns out to be the best performing database across algorithms such as AES, Blowfish, RC4 and ChaCha20, and SQLite performs the best for DES algorithm. The ranking of algorithms categorized by database is presented in Table 14.

TABLE XIV. Ranking of Algorithms by Databases

	Performance (Fastest to Slowest)
Encryption	RC4 < ChaCha20 < Blowfish < AES < DES
Decryption	RC4 < ChaCha20 < AES < Blowfish < DES

3. Algorithm Performance by Key Size

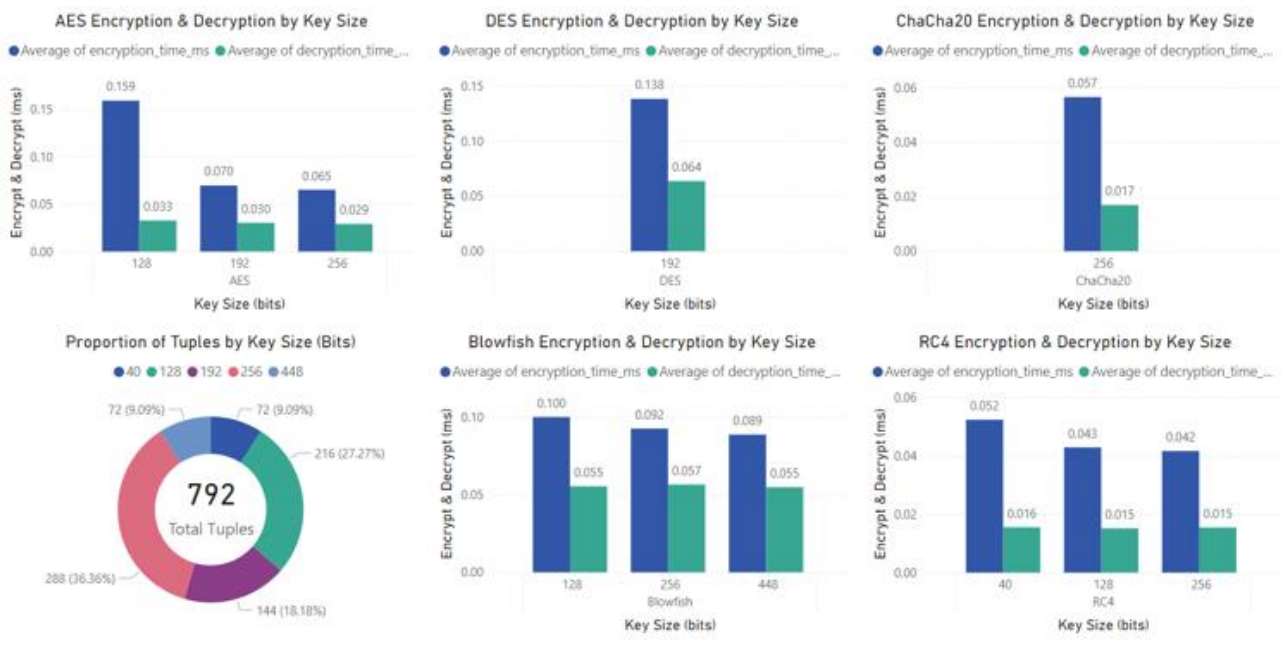


Fig 4. Algorithm Performance by Key Sizes

TABLE XV. Time Taken Across Varying Key Sizes

Algorithm	**Best Performing Bits**	**Encryption Time (ms)**	**Decryption Time (ms)**
AES	256	0.065	0.029
DES	192	0.138	0.064

Blowfish	448	0.089	0.055
RC4	256	0.042	0.015
ChaCha20	256	0.057	0.017

Figure 4 presents the combination of clustered column chart and pie chart for visualizing the symmetric encryption algorithms performance in terms of varying encryption key sizes. The average encryption and decryption time taken for each key size was documented as seen in Table 15. For **AES**, **Blowfish** and **RC4** which employ multiple key sizes, it was observed that the higher the key size, the lower the encryption and decryption time taken to encrypt and decrypt the data. However, this contradicts with the typical expectation that larger encryption key sizes result in higher processing times due to the increased complexity of encryption operations. The unexpected result could be due to the impact of the hardware environment used for the tests. Whereas **ChaCha20** and **DES** algorithm are also considered as efficient algorithms in terms of their speed. Additionally, it was observed that the AES with 128 bits took the longest time for encryption, and this resulted in a huge difference between the encryption and decryption time. The ranking of performance for each algorithm based on varying key sizes is presented in Table 16.

TABLE XVI. Ranking of Algorithms by Key Sizes

	Performance (Fastest to Slowest)
Encryption	RC4 < ChaCha20 < AES < Blowfish < DES
Decryption	RC4 < ChaCha20 < AES < Blowfish < DES

4. Algorithm Performance by Plain Text Length

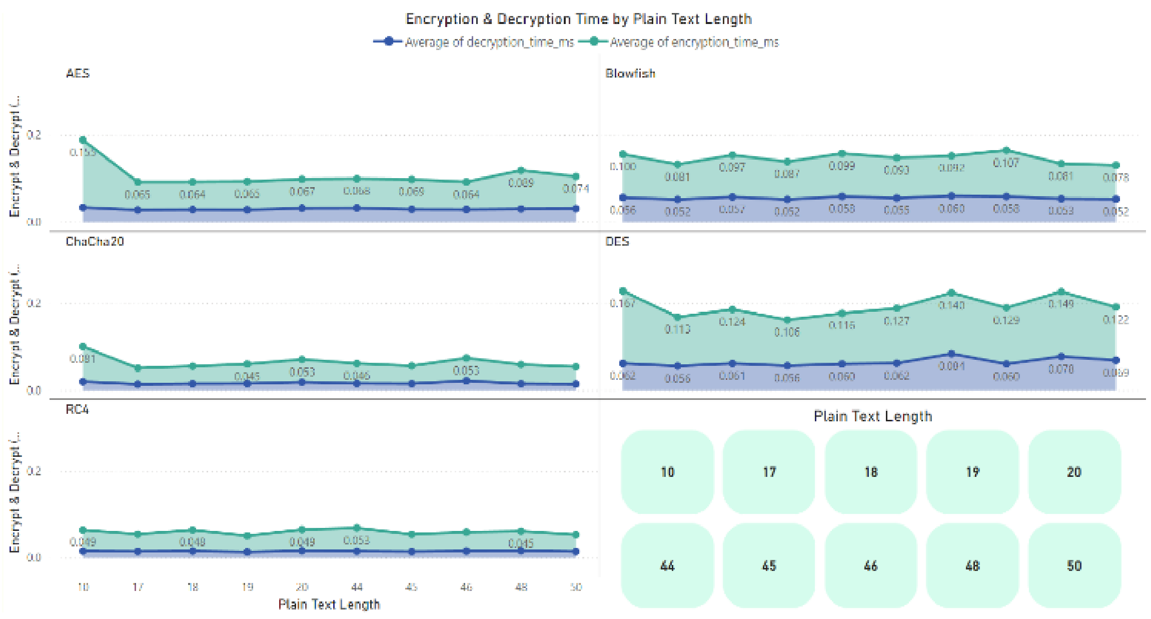


Fig 5. Algorithm Performance by Plain Text Length

Based on Figure 5 which shows the algorithm performance based on different plain text length, it was observed that **DES** is still considered to have the highest encryption and decryption time taken as compared to the remaining. Whereas **RC4** appears to be the fastest algorithm as usual in terms of the encryption and decryption time, followed by **ChaCha20**. **Blowfish** was observed to be placed after DES as it takes longer time for processing as well. While **AES** performed ideally for encryption with having consistent performance across various categories of text length, but it turned out to be slightly unstable for decryption time by having longest time for the length of 10 characters, and Blowfish performed averagely as compared to the remaining algorithm. Ranking of performance based on different text length is depicted in Table 17.

TABLE XVII. Ranking of Algorithms by Text Length

	Performance (Fastest to Slowest)
Encryption	RC4 < ChaCha20 < AES < Blowfish < DES
Decryption	RC4 < ChaCha20 < AES < Blowfish < DES

5. Algorithm Performance by Plain Text Types

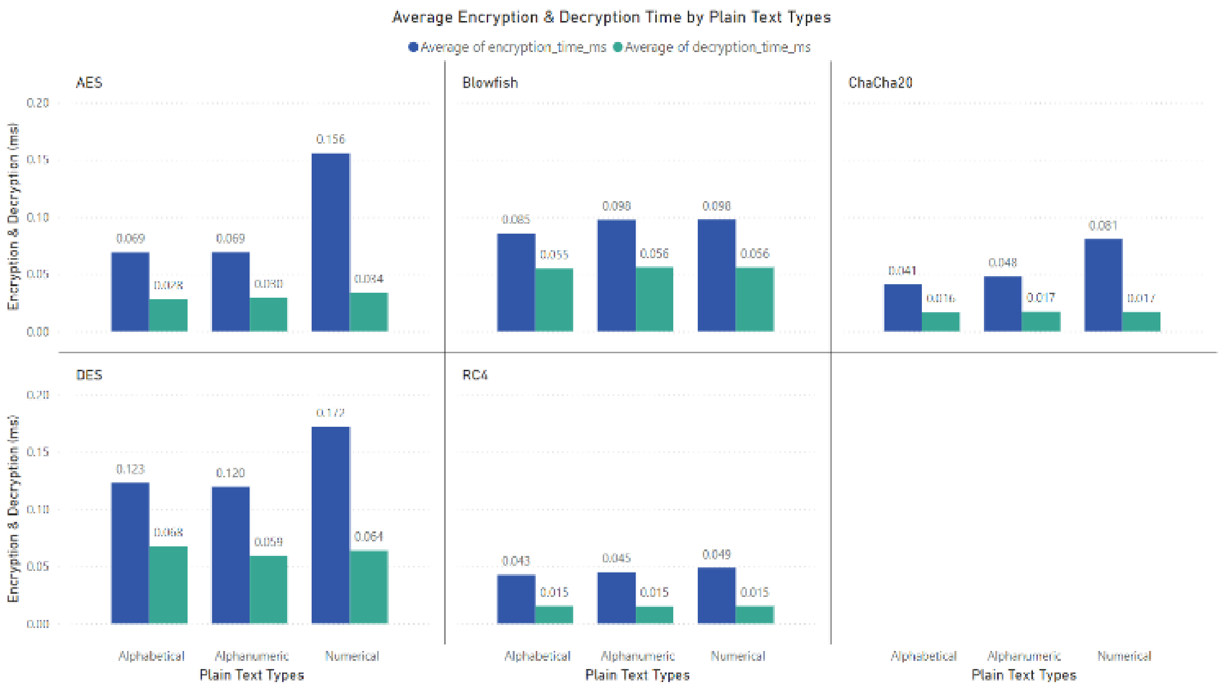


Fig 6. Algorithm Performance by Plain Text Types

TABLE XVIII. Time Taken by Algorithm in Best Performing Text Types

Algorithm	Best Performing Text Types	Encryption Time (ms)	Decryption Time (ms)
AES	Alphabetic	0.069	0.028
DES	Alphanumeric	0.120	0.059
Blowfish	Alphabetic	0.085	0.055
RC4	Alphabetic	0.043	0.015
ChaCha20	Alphabetic	0.041	0.016

In accordance with Fig 6. *Algorithm Performance by Plain Text Types* which presents the performance evaluation of algorithms based on different plain text types – numeric, alphabetic and alphanumeric, with Table 18 shows a detailed analysis of the best performing text types for each algorithm. Based on the visualization, algorithms such as **AES**, **Blowfish**, **RC4** and **ChaCha20** were performing better and faster for alphabetic values – data consisting of only alphabets. This is because alphabetic systems often use straightforward and less complex mapping architecture, hence it processes faster compared to numerical and alphabetic values. Whereas **DES** was best performed for encrypting and decrypting alphanumeric values. RC4 in this case is still outperforming the rest of the algorithms with having the shortest processing time. Based on observation, alphanumeric values took the longest time in processing due to its complex nature, thus it provides a better security in keeping the information more secure. Table 19 shows the algorithm rankings categorized by various text types.

TABLE XIX. Ranking of Algorithms by Plain Text Types

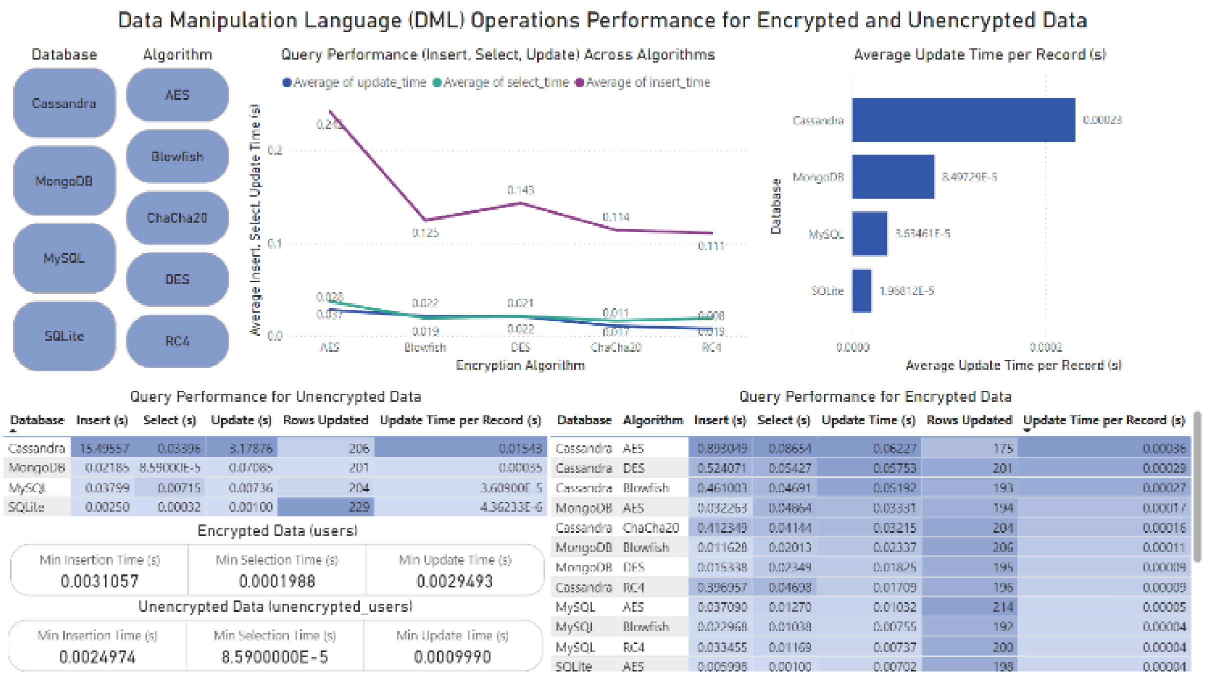
	Performance (Fastest to Slowest)
Encryption	RC4 < ChaCha20 < AES< Blowfish < DES
Decryption	RC4 < ChaCha20 < AES < Blowfish < DES

B. Phase 2 - Real-World Scenario Simulation with Field-Level Encryption

Prior to analyzing the real-world scenario simulation with a payment or transaction system, specific key size was selected from each encryption algorithm respectively based on the algorithm performance categorized by key size obtained in Phase 1 result. The final decision on the key size to be employed in Phase 2 is outlined as follows in Table 20. These key sizes were chosen as they appear to be outperforming as compared to the remaining key sizes in Phase 1.

TABLE XX. Final Selection for Scenario Simulation

Algorithms	**Key Size (bits)**
AES	256
DES	192
Blowfish	448
RC4	256
ChaCha20	256


Data Manipulation Language (DML) Operations Performance for Encrypted and Unencrypted Data
Database
Cassandra
MongoDB
MySQL
SQLite
Algorithm
AES
Blowfish
ChaCha20
DES
RC4
Query Performance (Insert, Select, Update) Across Algorithms
● Average of update_time ● Average of select_time ● Average of insert_time
Average Insert, Select, Update Time (s)
0.2
0.1
0.0
0.24
0.125
0.143
0.114
0.111
0.028
0.037
0.022
0.019
0.021
0.022
0.011
0.017
0.008
0.019
AES
Blowfish
DES
ChaCha20
RC4
Encryption Algorithm
Average Update Time per Record (s)
Database
Cassandra
0.00023
MongoDB
8.49729E-5
MySQL
3.63461E-5
SQLite
1.96812E-5
0.0000
0.0002
Average Update Time per Record (s)

Query Performance for Unencrypted Data

Database	Insert (s)	Select (s)	Update (s)	Rows Updated	Update Time per Record (s)
Cassandra	15.49557	0.03396	3.17876	206	0.01543
MongoDB	0.02185	8.59000E-5	0.07085	201	0.00035
MySQL	0.03799	0.00715	0.00736	204	3.60900E-5
SQLite	0.00250	0.00032	0.00100	229	4.36233E-6

Encrypted Data (users)

Min Insertion Time (s)	Min Selection Time (s)	Min Update Time (s)
0.0031057	0.0001988	0.0029493

Unencrypted Data (unencrypted_users)

Min Insertion Time (s)	Min Selection Time (s)	Min Update Time (s)
0.0024974	8.5900000E-5	0.0009990

Query Performance for Encrypted Data

Database	Algorithm	Insert (s)	Select (s)	Update Time (s)	Rows Updated	Update Time per Record (s)
Cassandra	AES	0.893049	0.08654	0.06227	175	0.00036
Cassandra	DES	0.524071	0.05427	0.05753	201	0.00029
Cassandra	Blowfish	0.461003	0.04691	0.05192	193	0.00027
MongoDB	AES	0.032263	0.04864	0.03331	194	0.00017
Cassandra	ChaCha20	0.412349	0.04144	0.03215	204	0.00016
MongoDB	Blowfish	0.011628	0.02013	0.02337	206	0.00011
MongoDB	DES	0.015338	0.02349	0.01825	195	0.00009
Cassandra	RC4	0.396957	0.04698	0.01709	196	0.00009
MySQL	AES	0.037090	0.01270	0.01032	214	0.00005
MySQL	Blowfish	0.022968	0.01038	0.00755	192	0.00004
MySQL	RC4	0.033455	0.01169	0.00737	200	0.00004
SQLite	AES	0.005998	0.00100	0.00702	198	0.00004

Fig 7. DML Operations Performance Overview

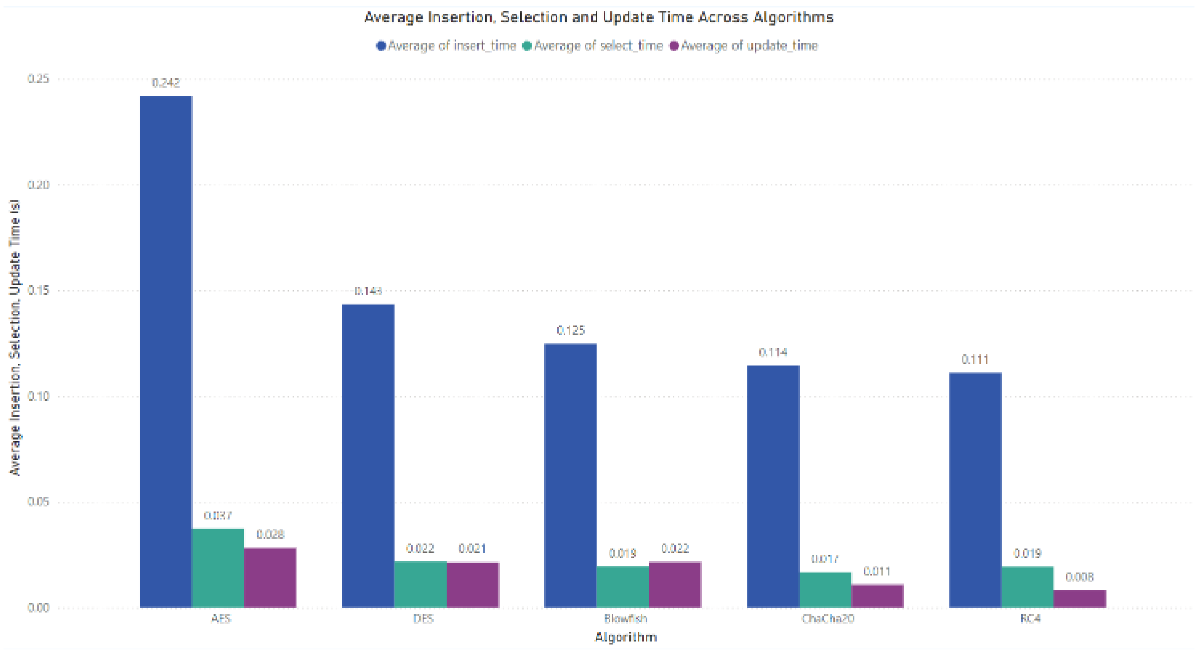


Fig 8. DML Operations Performance Analysis

Figure 7 presents the DML operations performance tested in all 5 algorithms using payment and transaction settings. The result shows that in the comparison between unencrypted and encrypted data, the minimum time taken for both insertion, selection and update time for encrypted data is significantly higher than those unencrypted. This is because the encrypted data are often complex in nature and requires more processing steps as compared to the unencrypted data. Referring to the line chart titled 'Query Performance (Insert, Select, Update) Across Algorithms' shown in Figure 7 and clustered column chart named 'Average Insertion, Selection and Update Time Across Algorithms' as shown in Figure 8, in terms of Insertion time, **AES-256** has the highest insertion time, while **RC4-256** has the lowest processing time. As for insertion and update time, both algorithms performed nearly identical but with AES taking slightly longer time for selecting and updating queries. Although **DES-192** appeared to be the slowest algorithm as tested in Phase 1, it performed faster than AES in terms of both insertion, selection and update time in Phase 2.

On the whole, as observed in the results obtained from Phases 1 and 2, it can be concluded that RC4 is the most efficient symmetric encryption algorithm among all the rest of the algorithms, whereas DES turns out to be taking the most processing time in encryption, but it performed averagely in terms of DML operations. It was also concluded that AES performs averagely with having balance performance between both encryption, decryption and DML operations.

5. CONCLUSIONS

This paper has presented a detailed comparison and analysis of all the symmetric encryption algorithms examined in this study (AES, DES, Blowfish, RC4 and ChaCha20) towards various perspectives – databases, encryption key sizes, plain text sizes and plain text types by implementing Python programming language. The encryption time and decryption time taken by each algorithm based on each criterion were carefully documented and presented by building an interactive Microsoft Power BI dashboard function for detailed performance evaluation and analysis. The findings show that RC4 algorithm outperforms the remaining by establishing the highest performance in terms of encrypting and decrypting time in Phase 1, as well as the DML operations time in Phase 2 of the study. While DES consists of the slowest performance overall in comparison with the remaining algorithms.

This paper contributes to a better expansion of knowledge on how various degrees of factors may affect the performance of those studied symmetric encryption algorithms. It also serves as a useful resource for researchers and readers who are seeking insights into the trade-offs between encryption strength and computational efficiency.

Nonetheless, there exist several limitations within this study, such as the limitation persists in Apache Cassandra database in processing unencrypted data, as it experiences significantly longer connection time. Additionally, to achieve a higher accuracy of the results, this study also acknowledges the need for further testing by running additional stress tests to account for variability of the outputs and to strive for a more balanced set of results. Specifically, the statistical analysis methods including the calculation of variance and standard deviation should be conducted to ensure stability and the consistency of the results.

Apart from that, future research should also explore the impact of various hardware specifications in influencing the algorithms' performance to provide a more comprehensive understanding and increase the reliability of the result. Future research could also focus on examining the throughput of the symmetric encryption algorithms to further evaluate the efficiency of these algorithms, especially in real-world applications where vast amount of data need to be encrypted and decrypted promptly. All in all, this study provides useful insights into the symmetric encryption algorithm performance and the future work could focus on refining the testing methods and expanding the scope of the research to contribute further to the field of data protection and security.

REFERENCES

[1] R. Verma and A. Kumar Sharma, "SIMULATION-BASED COMPARATIVE ANALYSIS OF SYMMETRIC ALGORITHMS," *International Journal of Advanced Research in Computer Science*, vol. 11, no. 5, 2020, doi: 10.26483/ijarcs.v11i5.6655.

[2] R. K. Muhammed *et al.*, "Comparative Analysis of AES, Blowfish, Twofish, Salsa20, and ChaCha20 for Image Encryption," *Kurdistan Journal of Applied Research*, vol. 9, no. 1, pp. 52–65, May 2024, doi: 10.24017/science.2024.1.5.

[3] D. A. F. Saraiva, V. R. Q. Leithardt, D. de Paula, A. S. Mendes, G. V. González, and P. Crocker, "PRISEC: Comparison of symmetric key algorithms for IoT devices," *Sensors (Switzerland)*, vol. 19, no. 19, Oct. 2019, doi: 10.3390/s19194312.

[4] A. Salkanovic, S. Ljubic, L. Stankovic, and J. Lerga, "Analysis of Cryptography Algorithms Implemented in Android Mobile Application," *Information Technology and Control*, vol. 50, no. 4, pp. 786–807, Dec. 2021, doi: 10.5755/j01.itc.50.4.29464.

[5] Y. Mahamat, S. H. Othman, and S. H. Nkiama, "Comparative Study Of AES, Blowfish, CAST-128 And DES Encryption Algorithm," 2016. [Online]. Available: www.iosrjen.org

[6] I. Sumartono, A. Putera, U. Siahaan, and N. Mayasari, "An Overview of the RC4 Algorithm," *Article in IOSR Journal of Dental and Medical Sciences*, vol. 18, no. 6, pp. 67–73, 2016, doi: 10.9790/0661-1806046773.

[7] P. Nagar and N. B. Hulle, "Compact High Speed Reconfigurable Hardware Implementation of RC4 Stream Cipher," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP*, vol. 5, no. 1, pp. 2319–4197, 2015, doi: 10.9790/4200-05125456.

[8] S. Barbero, D. Bazzanella, and E. Bellini, "Rotational Cryptanalysis on ChaCha Stream Cipher," *Symmetry (Basel)*, vol. 14, no. 6, Jun. 2022, doi: 10.3390/sym14061087.

[9] M. H. Taha and J. M. Al-Tuwaijari, "Improvement of Chacha20 algorithm based on tent and Chebyshev chaotic maps," *Iraqi Journal of Science*, vol. 62, no. 6, pp. 2029–2039, Jul. 2021, doi: 10.24996/ijs.2021.62.6.29.

[10] K. Assa-Agyei and F. Olajide, "A Comparative Study of Twofish, Blowfish, and Advanced Encryption Standard for Secured Data Transmission," 2023. [Online]. Available: www.ijacsa.thesai.org

[11] B. A. Buhari, A. A. Obiniyi, K. Sunday, and S. Shehu, "Performance Evaluation of Symmetric Data Encryption Algorithms: AES and Blowfish," *Saudi Journal of Engineering and Technology*, vol. 04, no. 10, pp. 407–414, Oct. 2019, doi: 10.36348/sjeat.2019.v04i10.002.

[12] A. Boicea, F. Radulescu, C. O. Truica, and C. Costea, "Database Encryption Using Asymmetric Keys: A Case Study," in *Proceedings - 2017 21st International Conference on Control Systems and Computer, CSCS 2017*, Institute of Electrical and Electronics Engineers Inc., Jul. 2017, pp. 317–323. doi: 10.1109/CSCS.2017.50.

[13] A. M. Abbas, A. M. S. Rahma, and N. F. Hassan, "Comparative study on encrypted database techniques," *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 12, no. 3, Nov. 2020, doi: 10.29304/jqcm.2020.12.3.710.

[14] J. R. Sekhar and G. Sivaranjani, "Database Encryption Using TSFS Algorithm," 2018.

[15] T. Hirano, Y. Kawai, and Y. Koseki, "DBMS-Friendly Searchable Symmetric Encryption: Constructing Index Generation Suitable for Database Management Systems," in *2021 IEEE Conference on Dependable and Secure Computing, DSC 2021*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021. doi: 10.1109/DSC49826.2021.9346255.

[16] C. Krähenbühl and A. Perrig, "Searchable symmetric encryption," in *Trends in Data Protection and Encryption Technologies*, Springer Nature, 2023, pp. 71–75. doi: 10.1007/978-3-031-33386-6_14.

[17] T. S. Patel, S. Kolachina, D. P. Patel, and P. S. Shrivastav, "Comparative evaluation of different methods

of 'Homomorphic Encryption' and 'Traditional Encryption' on a dataset with current problems and developments."

[18] C. Ho, K. Pak, S. Pak, M. Pak, and C. Hwang, "A Study on Improving the Performance of Encrypted Database Retrieval Using External Indexing System of B+ Tree Structure," in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 706–714. doi: 10.1016/j.procs.2019.06.110.

[19] R. E. J. Paje, A. M. Sison, and R. P. Medina, "Multidimensional key RC6 algorithm," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jan. 2019, pp. 33–38. doi: 10.1145/3309074.3309095.

[20] J. W. . Bos and Martijn. Stam, *Computational cryptography: algorithmic aspects of cryptology*. Cambridge University Press, 2021.

[21] D. I. George Amalarethinam, J. Sai Geetha, and K. Mani, "Analysis and enhancement of speed in public key cryptography using Message Encoding Algorithm," *Indian J Sci Technol*, vol. 8, no. 16,

[22] A. Kumar, P. Pranav, S. Dutta, and S. Chakraborty, "Analysis of the empirical complexity of Advanced Encryption Standards-128 statistically," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 6, pp. 1887–1903, Sep. 2024, doi: 10.47974/JDMSC-1854.

[23] L. Tang, J.-N. Liu, D. Feng, and W. Tong, "An adaptive cryptographic accelerator for network storage security on dynamically reconfigurable platform," in *Eighth International Symposium on Optical Storage and 2008 International Workshop on Information Data Storage*, SPIE, Dec. 2008, p. 71251B. doi: 10.1117/12.823322.

[24] E. Ozturk, Y. Doroz, E. Savas, and B. Sunar, "A custom accelerator for homomorphic encryption applications," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 3–16, Jan. 2017, doi: 10.1109/TC.2016.2574340.

[25] Omolara Patricia Olaiya, Temitayo Oluwadamilola Adesoga, Azeez Adekunle Adebayo, Fehintola Moyosore Sotomi, Oluwaseun Aaron Adigun, and Paschal M Ezeliora, "Encryption techniques for financial data security in fintech applications," *International Journal of Science and Research Archive*, vol. 12, no. 1, pp. 2942–2949, Jun. 2024, doi: 10.30574/ijsra.2024.12.1.1210.

[26] H. Li, Y. Yang, Y. Dai, S. Yu, and Y. Xiang, "Achieving Secure and Efficient Dynamic Searchable Symmetric Encryption over Medical Cloud Data," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 484–494, Apr. 2020, doi: 10.1109/TCC.2017.2769645.

[27] C. H. Lin, J. X. Wu, P. Y. Chen, C. M. Li, N. S. Pai, and C. L. Kuo, "Symmetric Cryptography with a Chaotic Map and a Multilayer Machine Learning Network for Physiological Signal Infosecurity: Case Study in Electrocardiogram," *IEEE Access*, vol. 9, pp. 26451–26467, 2021, doi: 10.1109/ACCESS.2021.3057586.

[28] F. Gao, "Data encryption algorithm for e-commerce platform based on blockchain technology," *Discrete and Continuous Dynamical Systems - Series S*, vol. 12, no. 4–5, pp. 1457–1470, Aug. 2019, doi: 10.3934/dcdss.2019100.

[29] H. Zhang, "Application of Information Encryption Technology in Computer Network Communication Security," *Wirel Commun Mob Comput*, vol. 2022, 2022, doi: 10.1155/2022/9354441.